EdX Open Learning XML Guide - Alpha

Version

Release

Apr 12, 2017

Contents

This guide is intended for those interested in using OLX (open learning XML) to develop edX courses. Readers of this guide should be familiar with XML.

CHAPTER 1

General Information

Read Me

The *edX Open Learning XML Guide* provides the information you need to build an edX course through OLX (open learning XML) and supporting files, without using edX Studio.

This documentation is created using RST files and Sphinx. You, the user community, can help update and revise this documentation project on GitHub.

https://github.com/edx/edx-documentation/tree/master/en_us/olx/source

The edX documentation team welcomes contributions from Open edX community members. You can find guidelines for how to contribute to edX Documentation in the GitHub edx/edx-documentation repository.

Other edX Resources

Learners, course teams, researchers, developers: the edX community includes groups with a range of reasons for using the platform and objectives to accomplish. To help members of each group learn about what edX offers, reach goals, and solve problems, edX provides a variety of information resources.

To help you find what you need, browse the edX offerings in the following categories.

- *Resources for Developers*
- *Resources for Open edX*

All members of the edX community are encouraged to make use of the resources described in this preface. We welcome your feedback on these edX information resources. Contact the edX documentation team at docs@edx.org.

Resources for edx.org Learners

Documentation

The EdX Learner's Guide is available on the docs.edx.org web page. This guide is also available when you select **Help** while you are in a course, and from your dashboard of enrolled courses.

In a Course

All edX courses have a discussion page where you can ask questions and interact with other students and with the course team: select **Discussion**. Many courses also offer a wiki for additional resources and materials: select **Wiki**.

Other resources might also be available, such as a course-specific Facebook page or Twitter feed. Be sure to check the **Home** page for your course as well as the **Discussion** and **Wiki** pages.

Resources on the edx.org Website

To help you get started with the edX learning experience, edX offers a course (of course!). You can find the edX Demo course on the edx.org website.

When you are working in an edX course, you can select **Support** to access a help center with frequently asked questions and answers.

If you still have questions or suggestions, you can get help from the edX support team: select **Support**, select **Contact** at the bottom of any edX web page, or send an email message to info@edx.org.

For opportunities to meet others who are interested in edX courses, check the edX Global Community meetup group.

The edX Partner Portal

The edX Partner Portal is the destination for partners to learn, connect, and collaborate with one another. Partners can explore rich resources and share success stories and best practices while staying up-to-date with important news and updates.

To use the edX Partner Portal, you must register and request verification as an edX partner. If you are an edX partner and have not used the edX Partner Portal, follow these steps.

1. Visit partners.edx.org, and select **Create New Account**.
2. Select **Request Partner Access**, then fill in your personal details.
3. Select **Create New Account**. You will receive a confirmation email with your account access within 24 hours.

After you create an account, you can sign up to receive email updates about edX releases, news from the product team, and other announcements. For more information, see *Release Announcements by Email*.

Course Team Support in the edX Partner Portal

EdX partner course teams can get technical support in the edX Partner Portal. To access technical support, submit a support ticket, or review any support tickets you have created, go to partners.edx.org and select **Course Staff Support** at the top of the page. This option is available on every page in the Partner Portal.

The Open edX Portal

The Open edX Portal is the destination for learning about hosting an Open edX instance, extending the edX platform, and contributing to Open edX. In addition, the Open edX Portal provides product announcements and other community resources.

All users can view content on the Open edX Portal without creating an account and logging in.

To comment on blog posts or the edX roadmap, or subscribe to email updates, you must create an account and log in. If you do not have an account, follow these steps.

1. Visit open.edx.org/user/register.
2. Fill in your personal details.
3. Select **Create New Account**. You are then logged in to the Open edX Portal.

Release Announcements by Email

To receive and share product and release announcements by email, you can subscribe to announcements on one of the edX portal sites.

1. Create an account on the Open edX Portal or the edX Partner Portal as described above.
2. Select **Community** and then **Announcements**.
3. Under **Subscriptions**, select the different types of announcements that you want to receive through email. You might need to scroll down to see these options.
4. Select **Save**.

You will now receive email messages when new announcements of the types you selected are posted.

System Status

For system-related notifications from the edX operations team, including outages and the status of error reports. On Twitter, you can follow @edxstatus.

Current system status and the uptime percentages for edX servers, along with the Twitter feed, are published on the edX Status web page.

Resources for edx.org Course Teams

Course teams include faculty, instructional designers, course staff, discussion moderators, and others who contribute to the creation and delivery of courses on edx.org or edX Edge.

The edX Course Creator Series

The courses in the edX Course Creator Series provide foundational knowledge about using the edX platform to deliver educational experiences. These courses are available on edx.org.

- *edX101: Overview of Creating a Course*
- *StudioX: Creating a Course with edX Studio*
- *BlendedX: Blended Learning with edX*
- *VideoX: Creating Video for the edX Platform*

edX101: Overview of Creating a Course

The edX101 course is designed to provide a high-level overview of the course creation and delivery process using Studio and the edX LMS. It also highlights the extensive capabilities of the edX platform.

StudioX: Creating a Course with edX Studio

After you complete edX101, StudioX provides more detail about using Studio to create a course, add different types of content, and configure your course to provide an optimal online learning experience.

BlendedX: Blended Learning with edX

In BlendedX you explore ways to blend educational technology with traditional classroom learning to improve educational outcomes.

VideoX: Creating Video for the edX Platform

VideoX presents strategies for creating videos for course content and course marketing. The course provides step-by-step instructions for every stage of video creation, and includes links to exemplary sample videos created by edX partner institutions.

Documentation

Documentation for course teams is available on the docs.edx.org web page.

- Building and Running an edX Course is a comprehensive guide with concepts and procedures to help you build a course in Studio and then use the Learning Management System (LMS) to run a course.

  You can access this guide by selecting **Help** in Studio or from the instructor dashboard in the LMS.

- Using edX Insights describes the metrics, visualizations, and downloadable .csv files that course teams can use to gain information about student background and activity.
- The edX Release Notes summarize the changes in each new version of deployed software.

These guides open in your web browser. The left side of each page includes a **Search docs** field and links to the contents of that guide. To open or save a PDF version, select **v: latest** at the lower right of the page, then select **PDF**.

Note: If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

Email

To receive and share information by email, course team members can:

- Subscribe to announcements and other new topics in the edX Partner Portal or the Open edX Portal. For information about how to subscribe, see *Release Announcements through the Open edX Portal*.
- Join the openedx-studio Google group to ask questions and participate in discussions with peers at other edX partner organizations and edX staffers.

Wikis and Web Sites

The edX product team maintains public product roadmaps on *the Open edX Portal* and *the edX Partner Portal*.

The edX Partner Support site for edX partners hosts discussions that are monitored by edX staff.

Resources for Researchers

At each partner institution, the data czar is the primary point of contact for information about edX data. To set up a data czar for your institution, contact your edX partner manager.

Data for the courses on edx.org and edX Edge is available to the data czars at our partner institutions, and then used by database experts, statisticians, educational investigators, and others for educational research.

Resources are also available for members of the Open edX community who are collecting data about courses running on their sites and conducting research projects.

Documentation

The edX Research Guide is available on the docs.edx.org web page. Although it is written primarily for data czars and researchers at partner institutions, this guide can also be a useful reference for members of the Open edX community.

The *edX Research Guide* opens in your web browser, with a **Search docs** field and links to sections and topics on the left side of each page. To open or save a PDF version, select **v: latest** at the lower right of the page, and then select **PDF**.

Note: If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

Discussion Forums and Email

Researchers, edX data czars, and members of the global edX data and analytics community can post and discuss questions in our public research forum: the openedx-analytics Google group.

The edX partner portal also offers community forums, including a Research and Analytics topic, for discussions among edX partners.

Important: Please do not post sensitive data to public forums.

Data czars who have questions that involve sensitive data, or that are institution specific, can send them by email to data.support@edx.org with a copy to your edX partner manager.

Wikis

The edX Analytics team maintains the Open edX Analytics wiki, which includes links to periodic release notes and other resources for researchers.

The edx-tools wiki lists publicly shared tools for working with the edX platform, including scripts for data analysis and reporting.

Resources for Developers

Software engineers, system administrators, and translators work on extending and localizing the code for the edX platform.

Documentation

Documentation for developers is available on the docs.edx.org web page.

- The edX Platform Developer's Guide includes guidelines for contributing to the Open edX project, options for extending the Open edX platform, instrumenting analytics, and testing.
- Installing, Configuring, and Running the Open edX Platform provides procedures for getting an edX developer stack (devstack) and production stack (fullstack) operational.
- Open edX XBlock Tutorial guides developers through the process of creating an XBlock, and explains the concepts and anatomy of XBlocks.
- Open edX XBlock API Guide provides reference information about the XBlock API.
- edX Open Learning XML Guide provides guidelines for building edX courses with OLX (open learning XML). Note that this guide is currently an alpha version.
- edX Data Analytics API provides reference information for using the data analytics API to build applications to view and analyze learner activity in your course.
- edX Platform APIs provide reference information for building applications to view course information and videos and work with user and enrollment data.

Note: If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

GitHub

These are the main edX repositories on GitHub.

- The edx/edx-platform repo contains the code for the edX platform.
- The edx/edx-analytics-dashboard repo contains the code for edX Insights.

- The edx/configuration repo contains scripts to set up and operate the edX platform.

Additional repositories are used for other projects. Our contributor agreement, contributor guidelines and coding conventions, and other resources are available in these repositories.

Getting Help

The Getting Help page in the Open edX Portal lists different ways that you can ask, and get answers to, questions.

Wikis and Web Sites

The Open edX Portal is the entry point for new contributors.

The edX Engineering team maintains an open Confluence wiki, which provides insights into the plans, projects, and questions that the edX Open Source team is working on with the community.

The edx-tools wiki lists publicly shared tools for working with the edX platform, including scripts and helper utilities.

Resources for Open edX

Hosting providers, platform extenders, core contributors, and course staff all use Open edX. EdX provides release-specific documentation, as well as the latest version of all guides, for Open edX users. The following documentation is available.

- Open edX Release Notes provides information on the contents of Open edX releases.
- Building and Running an Open edX Course is a comprehensive guide with concepts and procedures to help you build a course in Studio, and then use the Learning Management System (LMS) to run a course.
- Open edX Learner's Guide helps students use the Open edX LMS to take courses. This guide is available on the docs.edx.org web page. Because learners are currently only guided to this resource through the course, we encourage course teams to provide learners with links to this guide as needed in course updates or discussions.
- Installing, Configuring, and Running the Open edX Platform provides information about installing and using devstack and fullstack.
- The edX Platform Developer's Guide includes guidelines for contributing to the Open edX project, options for extending the Open edX platform, instrumenting analytics, and testing.
- Open edX XBlock Tutorial guides developers through the process of creating an XBlock, and explains the concepts and anatomy of XBlocks.
- Open edX XBlock API Guide provides reference information on the XBlock API.
- EdX Open Learning XML Guide provides guidelines for building edX courses with Open Learning XML (OLX). Note that this guide is currently an alpha version.
- EdX Data Analytics API provides reference information for using the data analytics API to build applications to view and analyze learner activity in your course.
- EdX Platform APIs provide reference information for building applications to view course information and videos and work with user and enrollment data.

Note: If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

---

edX Browser Support

The edX platform runs on the following browsers.

- Chrome
- Safari
- Firefox
- Microsoft Edge and Microsoft Internet Explorer 11

The edX platform is routinely tested and verified on the current version and the previous version of each of these browsers. We generally encourage the use of, and fully support only, the latest version.

Note: If you use the Safari browser, be aware that it does not support the search feature for the guides on docs.edx.org. This is a known limitation.

CHAPTER 2

What is Open Learning XML?

OLX (open learning XML) is the XML-based standard used to build courses for the edX Platform.

With OLX, you can:

- Move content between instances of Open edX.
- Create course content outside of edX Studio, including by conversion from other content formats.
- Ensure content remains free of proprietary encoding and allow portability.

XML Resources

OLX is based on XML. XML, or Extensible Markup Language, is a set of rules for creating documents in a format that is both human-readable and machine- readable.

To work with OLX, you should have a strong understanding of XML. This document presumes you understand XML and can use tools to create and edit XML files.

For a primer on XML, see the Wikipedia XML entry .

CHAPTER 3

Getting Started with OLX

To develop your course in OLX (open learning XML), edX's XML markup format, you complete the following steps.

1. *Define course policies*.
2. *Add course assets*.
3. *Create the course About page*.
4. *Create tabs, or pages, in your course*.
5. *Organize Courseware*.
6. *Create course components*.
7. *Create problems and tools*.

CHAPTER 4

OLX Course Structure

This topic describes the structure of a generic OLX (open learning XML) course.

For more information about how a specific OLX course is structured, see *The Structure of edX-Insider*.

For more information about how a course exported from edX Studio is structured, see *Example of OLX for a Studio Course*.

OLX and Directory File Structures

All files and subdirectories that comprise your OLX course are stored within a single directory.

OLX provides for some flexibility in the directory and file structure you use to build your course.

Top-level Directory

Starting out, it is easiest to create your courseware structure in a single file, the `course.xml file`.

This file can contain your entire course, but in most cases, it is convenient to split out large chunks of content into individual files. This is typically done either at the level of large components, such as problems or homework assignments.

Currently, when Studio exports a course, it places each component in its own file.

For example, the edX Platform contains a directory called manual-testing-complete that contains a course with all component types for testing purposes.

Descriptions of the directories needed for a typical course follow. You should set up these directories in preparation for developing your course content.

Note: If you are using custom XBlocks, you can include additional directories that store the XML for XBlocks of that type.

XBlock Directories

EdX course components can be broken out of the main `course.xml` file into individual files. Those files go into directories of the name of the component type (XML tag). For example, components of type `html` can be placed as individual files in the `html` directory. If your course does not contain .html files, or if they are all embedded in their top-level components, you do not need to create an `html` directory.

For information about several examples of these directories, see the following topics.

- *HTML Components*
- *Exercises, Tools, and Problem Types*
- *Video Components*

As the set of XBlocks grows, so does the set of associated XML tags and directories.

edX Platform Directories

In addition to the course hierarchy, which is designed to be generic and cross-platform, OLX courses contain a set of JSON and HTML files that specify course policies and non-courseware content.

`about` Directory

The `about` directory contains the following files.

- `overview.html`, which contains the content for the course overview page that learners see in the the Learning Management System (LMS).
- `short_description.html`, which contains the content for the course in the course list.

For more information, see *The Course About Pages*.

`info` Directory

The `info` directory contains the following files.

- `handouts.html`, which contains the content for the **Course Handouts** page in the course.
- `updates.html`, which contains the course updates learners see when opening a course.

`policies` Directory

The `policies` directory contains the following files.

- `grading_policy.json`, which defines how work is graded in the course.
- `policy.json`, which defines various settings in the course.
- `assets.json`, which defines all files used in the course, such as images.

For more information, see *Course Policies*.

`static` Directory

The `static` directory contains the files used in your course, such as images or PDFs.

For more information, see *Course Assets*.

`tabs` Directory

The `tabs` directory contains an HTML file for each page you add to your course.

For more information, see *Course Tabs*.

CHAPTER 5

Policies

The topics in this section describe how to use OLX (open learning XML) to define policies for your course.

Course Policies

You create a course policy file to specify metadata about your course.

Create the Course Policy File

You define policies for your course in the `policy.json` file.

Save the `policy.json` file in the `policy/<course-name>` directory.

The `<course-name>` directory must match the value of the `url_name` attribute in the `course.xml` file.

Course Policy JSON Objects

`start`	The start date for the course. For example: `"2012-09-05T12:00"`.
`advertised_start`	The start date displayed in the course listing and course about pages. For example: `"2012-09-05T12:00`.
`disable_policy_graph`	Whether the policy graph should be disabled (`true`) or not (`false`).
`enrollment_start`, `enrollment_end`	The dates in which students can enroll in the course. For example, `"2012-09-05T12:00"`. If not specified, students can enroll any time.
`end`	The end date for the course. For example: `"2012-11-05T12:00"`.
`end_of_course_survey_url`	The URL for an end of course survey. The link is shown after the course is over, next to certificate download links.
`tabs`	Custom pages, or tabs, in the courseware. See below for details.
`discussion_blackouts`	An array of time intervals during which students cannot create or edit discussion posts. For example, you could specify blackout dates during exams. For example: `[[""2012-10-29T04:00", "2012-11-03T04:00"], ["2012-12-30T04:00", "2013-01-02T04:00"]]` Course team members with the Discussion Moderator, Community TAs, or Administrator role are not restricted during blackout periods.
`show_calculator`	Whether the calculator is shown in the course (`true`) or not (`false`).
`days_early_for_beta`	The number of days early that students in the beta-testers group can access the course.
`cohort_config`	• `cohorted` : Boolean. Set to `true` if this course uses student cohorts. If so, all inline discussions are automatically cohorted, and top-level discussion topics are configurable via the `cohorted_discussions` list. Default is `false`, not cohorted). • `cohorted_discussions`: list of discussion topics that should be cohorted. Any not specified in this list are not cohorted. • `auto_cohort_groups`: `["group name 1", "group name 2", ...]` If `cohorted` is `true`, each student is automatically assigned to a random group from this list, creating the group if needed.
`pdf_textbooks`	Have pdf-based textbooks on tabs in the courseware. See below for details.
`html_textbooks`	The addition of HTML-based textbooks on tabs in the courseware has been deprecated.

Example Course Policy File

An example with a few of the settings defined in the course policy file follows.

```
{
  "course/course": {
    "advanced_modules": [
        "poll",
        "survey",
    ],
    "discussion_blackouts": [],
    "discussion_topics": {
        "General": {
            "id": "course"
        }
    "show_calculator": true,
    "show_reset_button": true,
    "start": "2017-10-01T00:30:00Z",
    "tabs": [
        {
            "course_staff_only": false,
            "name": "Home",
            "type": "course_info"
        },
        {
            "course_staff_only": false,
            "name": "Course",
            "type": "courseware"
        },
        {
            "course_staff_only": false,
            "name": "Discussion",
            "type": "discussion"
        },
        {
            "course_staff_only": false,
            "is_hidden": true,
            "name": "Wiki",
            "type": "wiki"
        },
        {
            "course_staff_only": false,
            "name": "Progress",
            "type": "progress"
        },
        {
            "course_staff_only": true,
            "name": "Staff only (Alison)",
            "type": "static_tab",
            "url_slug": "7cf2fccec33541dc81ce5e0e34e2689c"
        }
    ],
    "user_partitions": [
        {
            "active": true,
            "description": "The groups in this configuration can be mapped to cohort␣
↪groups in the LMS.",
            "groups": [
```

```
                {
                    "id": 1124782865,
                    "name": "Group A",
                    "version": 1
                },
                {
                    "id": 254579781,
                    "name": "Group B",
                    "version": 1
                }
            }
        ]
}
```

Grading Policy

You create a grading policy file to specify how problems are graded in your course.

Create the Grading Policy File

You define policies for your course in the `grading_policy.json` file.

Save the `grading_policy.json` file in the `policy/<course-name>` directory.

The `<course-name>` directory must match the value of the `url_name` attribute in the `course.xml` file.

Course Policy JSON Objects

`GRADE_CUTOFFS`	The minimal grade for passing the course, and receiving a certificate if offered.
`GRADER`	For each assignment type: • `min_count`: TBD • **`weight`: The percentage of the total grade determined by** assignments of this type. The total value for all assignment types must equal 1.0. • `type`: The name of the assignment type. • `short_label`: The label for the assignment type shown on the student's Progress page. • `drop_count`: The number of assignments of this type that can be dropped when calculating the final grade.

Example Grading Policy File

```
{
    "GRADE_CUTOFFS": {"Pass": 0.6},
    "GRADER": [
                {
                    "min_count": 3,
                    "weight": 0.75,
                    "type": "Homework",
                    "drop_count": 1,
                    "short_label": "Ex"
                },
                {
                    "short_label": "",
                    "min_count": 1,
                    "type": "Exam",
                    "drop_count": 0,
                    "weight": 0.25
                }
            ]
}
```

Course Asset Policy

You create an asset policy file to provide details of the assets used in your course. Assets can include image files, textbooks, handouts, and supporting JavaScript files.

You must enter policy details for each asset you add to the `static` directory. For more information, see *Course Assets*.

Create the Asset Policy File

You define policies for your assets in the `assets.json` file.

Save the `assets.json` file in the `policy` directory. You use one `assets.json` file for all of the courses you might have in your directory structure.

Asset Policy JSON Objects

`contentType`	The MIME type of the file.
`displayname`	The file name.
`locked`	`true` if users can only access the file from within your course. `false` if users can access the file from outside of your course.
`content_son`	A collection that contains: * `category`: Equal to `asset`. * `name`: The file name. * `course`: The course number. * `tag`: * `org`: The organization that created the course. * `revision`
`filename`	The full path and name of the file in the edX Platform.
`import_path`	TBD
`thumbnail_location`	An array containing: * `c4x` * The organization. * The course number. * `thumbnail` * The filename for the thumbnail.

Example Asset Policy File

The following example shows the JSON policy for one image file.

```
{
    "dashboard.png":
        {
            "contentType": "image/png",
            "displayname": "dashboard.png",
            "locked": false,
            "content_son":
                {
                    "category": "asset",
                    "name": "dashboard.png",
                    "course": "Course number",
                    "tag": "c4x",
                    "org": "Organization",
                    "revision": null
                },
                "filename": "/c4x/Organization/Course-number/asset/dashboard.png",
                "import_path": null,
                "thumbnail_location":
                    [
                        "c4x",
                        "Organization",
                        "Course number",
                        "thumbnail",
                        "dashboard.jpg",
                        null
                    ]
        }
}
```

CHAPTER 6

Course Assets

You must put all assets, or files that support your course, in the `static` directory. Assets include any image files, textbooks, handouts, and supporting JavaScript files.

You must also define the asset in the assets policy file. For more information, see *Course Asset Policy*.

CHAPTER 7

The Course About Pages

The topics in this section describe how to use OLX (open learning XML) to create and edit the information that prospective learners see about your course.

Course Overview

Each course must have an overview page. Learners see the overview page when searching and registering for the course.

Create the Overview File

In the `overview` directory, you create an HTML file called `overview.html`.

Overview Sections

The `overview.html` must contain specific sections.

Each section is wrapped in `section` tags. The value of the `class` attribute specifies what the section is for and how it is displayed to learners. Within the `section` tags, you use valid HTML.

The overview must contain sections with the following names.

- `about`
- `prerequisites`
- `course-staff`
- `teacher`
- `faq`

A Template For Your Course Overview

Replace the placeholders in the following template with your information.

```
<section class="about">
  <h2>About This Course</h2>
  <p>Include your long course description here. The long course description
    should contain 150-400 words.</p>
  <p>This is paragraph 2 of the long course description. Add more paragraphs
    as needed. Make sure to enclose them in paragraph tags.</p>
<section>
<section class="prerequisites">
  <h2>Prerequisites</h2>
  <p>Add information about class prerequisites here.</p>
</section>
<section class="course-staff">
  <h2>Course Team</h2>
  <article class="teacher">
    <div class="teacher-image">
      <!-- Replace the path below with the path to your faculty image. -->
      <img src="/c4x/edX/edX101/asset/Placeholder_FacultyImage.jpg"
        align="left" style="margin:0 20 px 0"/>
    </div>
    <h3>Team Member</h3>
    <p>Biography of course team member</p>
  </article>
  <article class="teacher">
    <div class="teacher-image">
      <img src="/c4x/edX/edX101/asset/Placeholder_FalcutyImage.jpg"/>
    </div>
    <h3>Team Member Name</h3>
    <p>Biography of course team member</p>
  </article>
</section>
<section class="faq">
  <section class="responses">
    <h2>Frequently Asked Questions</h2>
    <article class="response">
      <h3>What web browser should I use?</h3>
      <p>The Open edX platform works best with the current versions of Chrome,␣
↪Firefox, Safari, and Microsoft Edge.</p>
      <p>See our <a href="http://edx.readthedocs.io/projects/open-edx-learner-guide/
↪en/latest/front_matter/browsers.html">list of supported browsers</a> for the most␣
↪up-to-date information.</p>
    </article>
    <article class="response">
      <h3>Question 2?</h3>
      <p>Answer 2.</p>
    </article>
  </section>
</section>
```

Short Description

Optionally, you can define a short description for your course.

Learners see the short description when they move their cursors over the course image in the catalog.

Create the Short Description File

You create an HTML file called `short_description.html` in the `overview` directory.

The short description is limited to 150 characters.

Within that limit, you can add any text and HTML markup to the short description file.

CHAPTER 8

Course Tabs

You can add tabs, or pages, to your course. Each page appears in your course's navigation bar.

Create the Tab File

For each page you want your course to offer, you create an HTML file in the `tabs` directory.

You can add any text and HTML markup to the page. Pages can also be links or other types of content. One design pattern is to link a tab to a chromeless XBlock in the courseware, which allows for top-level interactive course content.

CHAPTER 9

Organizing Courseware

The topics in this section describe how to use OLX (open learning XML) to organize your courseware.

OLX Course Building Blocks

Before you begin using OLX (open learning XML) to set up a course, you should understand the building blocks of an edX course.

Courseware

Courseware is the main content of your course and consists mainly of lessons and assessments. The following list describes how courseware is organized in OLX.

- Course chapters are at the top level of your course and typically represent a time period. In Studio, chapters are called *sections*.
- A chapter contains one or more children which correspond to top-level pages in the course. In Studio, these are called 'subsections' and are currently restricted to `sequential` elements at this level. OLX supports any XBlock at this level.
- Courses are composed of structural elements, such as `sequential` and `vertical`, and leaf-nodes or content elements, such as `html` or `problem`. Studio has a fixed hierarchy where children of `sequential` elements are `vertical` elements (called units), and children of `vertical` elements are leaf elements (called modules).
  - *Course Components (XBlocks)*

- *Exercises, Tools, and Problem Types*

For more information, see *The Courseware Structure*.

Supplemental Course Content

In addition to the courseware described above, you course can contain supplemental content, such as textbooks, custom pages, and files. The following list describes the types of supported content.

- Course about pages appear in the course list for prospective students and are used to market your course. For more information, see *The Course About Pages*.
- Course assets are any supplemental files you use in your course, such as a syllabus as a PDF file or an image that appears in an HTML component. For more information, see *Course Assets*.
- Course pages are custom pages that you can have appear in the top navigation menu of your course. For more information, see *Course Tabs*.

Course Policies

Course policies determine how your course functions. For example, policies control grading and content experiments. For more information, see *Policies*.

The Courseware Structure

You develop the courseware structure in the `course.xml` file, in the top- level directory.

For an example of a `course.xml` file, see *The edX-Insider course.xml File*.

The `course.xml` File

The root element of the `course.xml` file is `course`.

An example of the contents of a `course.xml` file follows.

```
<course advanced_modules="[&quot;concept&quot;, &quot;done&quot;,
    &quot;profile&quot;, &quot;recommender&quot;]" course="edX_Insider"
    course_image="code.png" display_name="edX Demo"
    enrollment_start="2014-03-01T04:00:00Z" org="edX"
    start="2014-03-03T00:00:00Z" url_name="Ongoing">
    . . .
    . . .
```

course Attributes

Attribute	Meaning
advanced_modules	The list of advanced modules, or custom XBlocks, used in your course.
url_name	The value in the course URL path directly after the domain, organization, and course name. The url_name must also be the name of the course outline XML file (without the .xml extension).
org	The organization sponsoring the course. This value is in the course URL path, following the domain and /courses/.
course	The name of the course. This value is in the course URL path, following the organization.
course_image	The filename of the image used on the course About page.
enrollment_start	The date and time that students can start enrolling in the course.

course Element Attributes and Course URLS

The attributes of the course element are used to construct URLs in the course. The following course URL shows where these values are used.

```
http://my-edx-server.org/courses/<@org value>/<@course value>/<@url_name value>/info
```

For example:

```
http://my-edx-server.org/courses/edX/DemoX/Demo_Course/info
```

Course Chapters

You create a course chapter with the chapter element, as a child of the root course element. Chapter elements are top-level pages in the course. The edX platform renders navigation chrome around them (tab-set on top and accordion on the left). It is possible to disable chrome for specific chapters using the chrome option. It is possible to associate chapters with different elements of the tabset with the default_tab option. It is possible to hide them from the navigation using the hide_from_toc option.

For example, the course outline is defined by elements in the following format.

```
<course>
  <chapter display_name="Exam Review" url_name="exam_review">
  . . .
</course>
```

chapter Attributes

Attribute	Meaning
display_name	The value that is displayed to students as the name of the chapter, or section.
start	The date and time, in UTC, that the chapter is released to students. Before this date and time, students do not see the chapter.

chapter Children

The chapter element contains one or more children. Studio uses sequential elements for all children of chapters, and calls these subsections.

The following example shows a chapter with two sequentials, or subsections.

```
<chapter display_name="Example Week 2: Get Interactive">
    <sequential display_name="Simulations" url_name="simulations">
        . . .
    <sequential display_name="Graded Simulations"
        url_name="graded_simulations">
        . . .
</chapter>
```

Course Sequentials

You create a course sequential with the `sequential` element, for each subsection in the chapter.

For example, the course can contain a sequential in this format.

```
<course>
    <chapter url_name="exam_review">
        <sequential display_name="Simulations" url_name="simulations">
            . . .
        </sequential>
    </chapter>
    . . .
</course>
```

`sequential` Attributes

Attribute	Meaning
`display_name`	The value that is displayed to students as the name of the sequential, or subsection.
`start`	The date and time, in UTC, that the sequential is released to students. Before this date and time, students do not see the sequential.
`graded`	Whether the sequential is a graded subsection; `true` or `false`.
`format`	If the sequential is graded, the assignment type.
`graceperiod`	If the sequential is graded, the number of seconds in the grace period.
`rerandomize`	TBD
`showanswer`	TBD
`xqa_key`	TBD

`sequential` Children

The `sequential` element contains one or more child `vertical` elements.

The `vertical` element references a vertical, or unit, in the course.

The following example shows a chapter with a sequential that has three verticals, or units.

```
<course>
    <chapter url_name="exam_review">
        <sequential display_name="Simulations" url_name="simulations">
            <vertical display_name="Unit 1" url_name="Lesson_1_Unit_1">
                . . . .
            <vertical display_name="Unit 2" url_name="Lesson_1_Unit_2">
                . . . .
        </sequential>
    </chapter>
```

```
    . . .
</course>
```

Course Verticals

In the course structure, a course vertical serves the following functions.

- Defines the display name for the vertical, or unit.
- Organizes components and other verticals in the vertical.

You create a course vertical with the `vertical` element, for each unit in the subsection.

For example, the course can contain a vertical in this format.

```
<course>
    <chapter url_name="exam_review">
        <sequential display_name="Simulations" url_name="simulations">
            <vertical display_name="Unit 1" url_name="Lesson_1_Unit_1"/>
                . . .
        </sequential>
    </chapter>
    . . .
</course>
```

`vertical` Attributes

Attribute	Meaning
`display_name`	The value that is displayed to students as the name of the sequential, or subsection.

`vertical` Children

The `vertical` element contains one or more child elements for each component in the vertical, or unit.

Note: You can embed the content of components in the `course.xml` file, as child elements of the `vertical` element. However, you might want to store components in separate files, to better enable content reuse across courses.

A vertical element can also contain a vertical element. You can nest verticals, or units, recursively.

Child elements of `vertical` refer to components in your course. The edX Platform supports a wide range of components, including custom XBlocks.

The following example shows a vertical with two components.

```
<vertical display_name="Lesson_1_Unit_1">
    <html url_name="Introduction"/>
    <video url_name="Unit_1_Video"/>
</vertical>
```

CHAPTER 10

Course Components (XBlocks)

The topics in this section describe how to use OLX (open learning XML) to create and edit course components.

HTML Components

Create the HTML Component

To add an HTML component to your course, you can embed the XML for it in the parent XML file, or split it up into either 1 or 2 additional files. You can break up the HTML configuration into an .xml file in the html directory and an additional .html file in the same directory.

Caution: If you are including HTML that is not valid HTML, you must break out HTML content in a separate file.

Example of an HTML Component Embedded in a Vertical

```
<vertical display_name="Lesson_1_Unit_1">
    ...
    <html>The above has an error. <b>x</b> should be <b>y</b> in the second equation.
↪</html>
</vertical>
```

Example of Separate HTML Files

You create an XML file in the `html` directory for the content that you choose to break out into separate HTML files.

The name of the XML file must match the value of the @url_name attribute of the `html` element in the vertical XML file.

For example, a vertical XML file contains the following url_name.

```
<vertical display_name="Lesson_1_Unit_1">
    <html url_name="Introduction"/>
    . . .
</vertical>
```

You create the file `html/Introduction.xml` to define the HTML component.

HTML Component XML File Elements

The root element of the XML file for the HTML component is file is `html`.

In this case, the `html` element contains no children.

`html` Element Attributes

Attribute	Meaning
`display_name`	Required. The value that is displayed to students as the name of the HTML component. If you do not supply a `display_name` value, "html" is supplied for you.
`file_name`	The name of the HTML file that contains the content for the HTML component, without the `.HTML` extension.

Example HTML Component XML File

The following example shows an XML file for an HTML component.

```
<html filename="Introduction" display_name="Unit Introduction"/>
```

Example HTML Component Content

In the component's HTML file, you add valid HTML to represent the content you want to be displayed to students. For example, the following is from an HTML file for the edX Demo course:

```
<h3>Lesson 2: Let's Get INTERACTIVE!</h3>
<p>
<img
src="/static/interactive_x250.png" alt="Interactive" width="250"
hspace="12" vspace="12" border="0" align="right" />Now that you know your
way around an edX course let's look at some of the exciting interactive
tools you may encounter. Use the unit navigation bar above to explore.
 </p>
<p>Once you have tried the interactive tools in this lesson,
make sure to check out the week 2 homework where we show you several of the
really cool interactive labs we&rsquo;ve created for past courses.
 They&rsquo;re fun to play with.  Many courses will have tools
and labs that you need to use to complete homework assignments.</p>
```

Discussion Components

You can add inline discussion components to any container in your course.

Create the XML File for a Discussion Component

You create an XML file in the `discussion` directory for each inline discussion component in your course.

The name of the XML file must match the value of the @url_name attribute of the `discussion` element in the vertical XML file.

An example of how you create a discussion component in a vertical XML file follows.

```
 <vertical display_name="Lesson_1_Unit_1">
    <discussion
        discussion_category="Essays"
        discussion_id="peer_grading_component"
        discussion_target="Peer Grading"
        display_name="Peer Grading"
        url_name="peer_grading_discussion"
     />
</vertical>
```

If you prefer to create the discussion in its own file, you can create it using the following format.

```
 <vertical display_name="Lesson_1_Unit_1">
    <discussion
        url_name="Introduce_Yourself"
     />
</vertical>
```

You create the file `discussion/Introduce_Yourself.xml` to define the inline discussion component.

Discussion Component XML File Elements

The root element of the XML file for the HTML component is file is `discussion`.

The `discussion` element contains no children.

`discussion` Element Attributes

Attribute	Meaning
`display_name`	Required. The value that is displayed to students as the name of the discussion component. If you do not supply a `display_name` value, "discussion" is supplied for you.
`discussion_category`	The name of the category for the inline discussion as shown in the main **Discussion** tab of the course.
`for`	A string that describes the discussion for students.
`id`	The unique identifier that the discussion forum service uses to refer to this inline discussion component. This value must be unique across all courses in the edX deployment. Therefore it is recommended that you use the standard *<course_name>_<course_run>_<descriptor>* as in the above example. Do not use a period (.) in the ID value.
`discussion_id`	TBD
`discussion_target`	TBD

Example Discussion Component XML File

The following example shows an XML file for a discussion component.

```
<discussion
    discussion_category="Essays"
    discussion_id="6e51dd8f181b44ffa6d91303a287ed3f"
    discussion_target="Peer Grading"
    display_name="Peer Grading"
 />
```

Video Components

You can add video components to any container in your course (such as a vertical or sequential). Studio places all video components inside verticals (which it calls units).

Create the XML File for a Video Component

To add a video component to your course, add it to the course XML tree as follows.

```
<video
  youtube="1.00:o2pLltkrhGM"
  url_name="Introduction_Lecture"
  display_name="Introduction Lecture"
  youtube_id_1_0="o2pLltkrhGM">
</video>
```

If you prefer to place the video component in its own file, you create an XML file in the `video` directory for each video component in your course.

The name of the XML file must match the value of the @url_name attribute of the `video` element in the vertical XML file.

For example, the vertical XML file uses the following format.

```
 <vertical display_name="Lesson_1_Unit_1">
    <video url_name="Introduction_Lecture"/>
    . . .
</vertical>
```

You create the file `video/Introduction_Lecture.xml` to define the video component.

Video Component XML File Elements

The root element of the XML file for the HTML component is file is `video`.

The `video` element contains a single `source` element.

`source` Element

The `source` element contains the following attribute.

Attribute	Meaning
`src`	The file path for the video file.

`video` Element Attributes

Attribute	Meaning
`display_name`	Required. The value that is displayed to students as the name of the video component. If you do not supply a `display_name` value, "video" is supplied for you.
`youtube`	The speed and ID pairings for the YouTube video source. The value can contain multiple speed:ID pairs, separated by commas.
`download_track`	Whether students can download the video track. `true` or `false`.
`download_video`	Whether students can download the video. `true` or `false`.
`html5_sources`	The file path for the HTML5 version of the video.
`show_captions`	Whether students can view the video captions. `true` or `false`.
`source`	TBD
`sub`	TBD
`youtube_id_0_75`	The YouTube ID for the video that plays at 75% normal speed.
`youtube_id_1_0`	The YouTube ID for the video that plays at 100% normal speed.
`youtube_id_1_25`	The YouTube ID for the video that plays at 125% normal speed.
`youtube_id_1_5`	The YouTube ID for the video that plays at 150% normal speed.

Example Video Component XML File

The following example shows an XML file for a video component.

```
<video
  youtube="0.75:xGKlr7nT_Zw,1.00:o2pLltkrhGM,1.25:XGsB9bA6rGU,1.50:_HuIF16HdTA"
  url_name="Introduction_Lecture"
  display_name="Introduction Lecture"
  download_video="true"
  html5_sources="[&quot;https://s3.amazonaws.com/edx-course-videos/school/
↪DemoCourseIntroductionVideo.mov&quot;]"
  source=""
  youtube_id_0_75="xGKlr7nT_Zw"
  youtube_id_1_0="o2pLltkrhGM"
  youtube_id_1_25="XGsB9bA6rGU"
  youtube_id_1_5="_HuIF16HdTA">

  <source src="https://s3.amazonaws.com/edx-course-videos/mit-6002x/6002-Tutorial-
↪00010_100.mov"/>
</video>
```

Problem Components

The problem component allows you to add interactive exercises to the verticals in your course. You can add many different types of exercises and problems. This section covers the basics of problem components: what they look like to you and your learners, and the options that every problem component has.

Problem Component Overview

The format for edX problems is based on the LON-CAPA XML format, although the two are not quite compatible. In the edX variant, problems are composed of the following types of tags.

- `inputtypes` are similar to XBlocks. They define ways for users to enter input into the problem.
- `responsetypes` are graders. They define how inputs are mapped to grades.
- `hinters` are used to provide feedback to problems.
- Standard HTML tags are used for formatting.

OLX is designed to allow mixing and matching of `inputtypes`, `responsetypes`, and `hinters`. For example, a numerical grader could match 7+-0.1%. Ideally, you could use this grader with any `inputtype` that returns numbers as its output, including a text box, equation input, slider, or multiple choice question. In practice, this does not always work. For example, in the example described above, a multiple choice question would not give an output in a format a numerical grader could handle.

In addition, in many cases, there is a 1:1 mapping between graders and inputs. For some types of inputs (especially discipline-specific specialized ones), only one grader is needed.

The most general grader is `customresponse`. This grader uses Python code to evaluate the input. By design, this ought to work with any inputtype, although there are bugs mixing this with a small number of the newer inputtypes.

Like LON-CAPA, OLX allows embedding of code to generate parameterized problems. Unlike LON-CAPA, edX supports Python (and not Perl). Otherwise, the syntax for parameterizing problems is approximately identical.

Creating Graded or Ungraded Problems

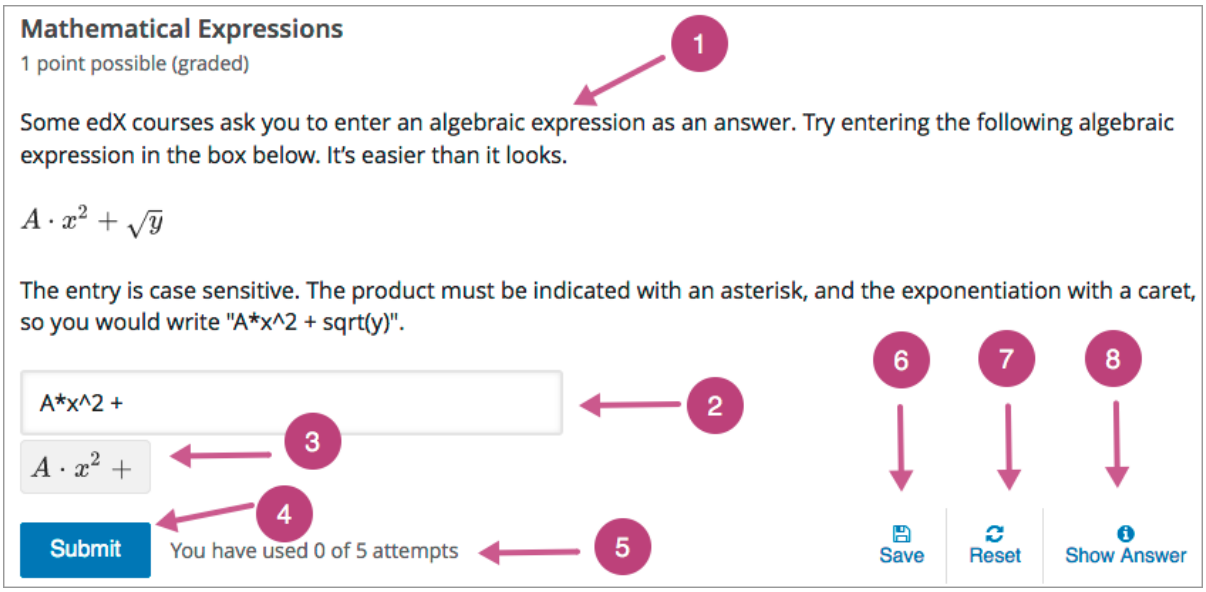
All problems receive a point score. However, when you establish the grading policy for your course, you specify the assignment types that will count toward learners' grades; for example, homework, lab, midterm, and final.

As you develop your course, you can add problem components to the units in any subsection. The problem components that you add automatically inherit the assignment type that is defined at the subsection level. You can only set assignment types at the subsection level, not at the unit or individual component level.

For more information, see *Grading Policy*.

The Learner View of a Problem

All problems on the edX platform have these component parts, some of which can be configured. For configurable options, you can specify whether and when an option is available in problems.



1. **Problem text.** The problem text can contain any standard HTML formatting.

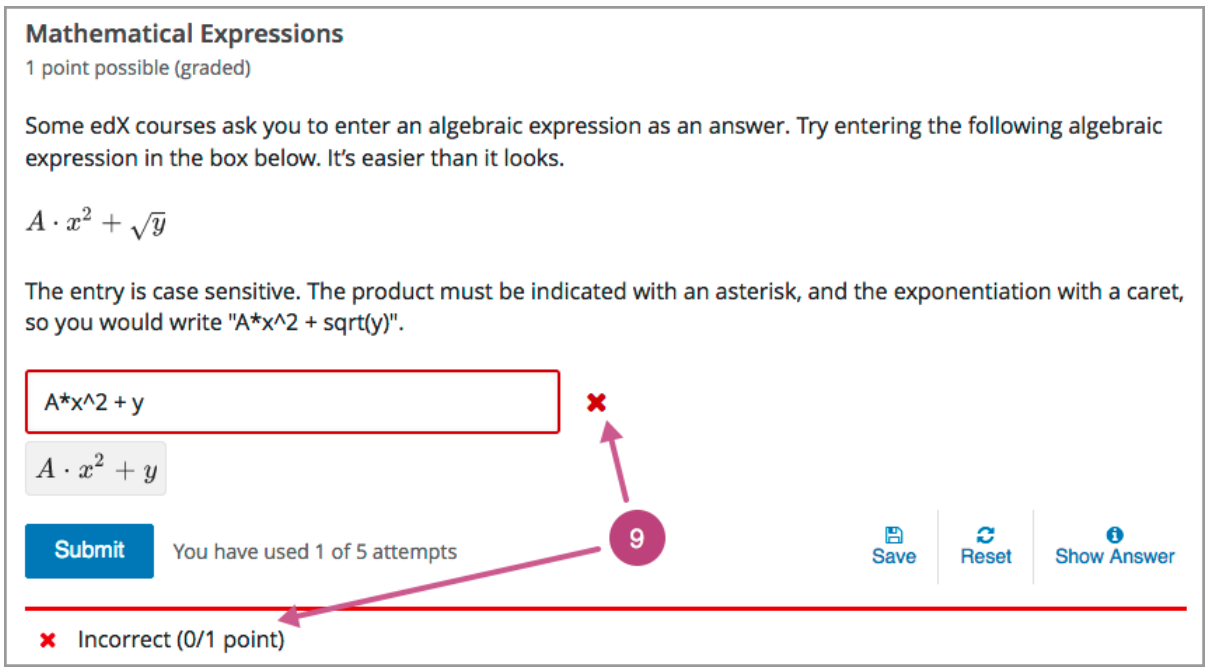
   Within the problem text for each problem component, you must identify a question or prompt, which is, specifically, the question that learners need to answer. This question or prompt also serves as the required accessible label, and is used by screen readers, reports, and Insights. For more information about identifying the question text in your problem, see The Simple Editor.

2. **Response field.** Learners enter answers in response fields. The appearance of the response field depends on the type of the problem.
3. **Rendered answer.** For some problem types, the LMS uses MathJax to render plain text as "beautiful math."
4. **Submit.** When a learner selects **Submit** to submit a response for a problem, the LMS saves the grade and current state of the problem. The LMS immediately provides feedback about whether the response is correct or incorrect, as well as the problem score. The **Submit** option remains available if the learner has unused attempts remaining, so that she can try to answer the problem again.
5. **Attempts.** You can set a specific number of attempts or allow unlimited attempts for a problem. By default, the course-wide **Maximum Attempts** advanced setting is null, meaning that the maximum number of attempts for problems is unlimited.

   In courses where a specific number has been specified for **Maximum Attempts** in Advanced Settings, if you do not specify a value for **Maximum Attempts** for an individual problem, the number of attempts for that problem defaults to the number of attempts defined in Advanced Settings.
6. **Save.** The learner can select **Save** to save his current response without submitting it for grading. This allows the learner to stop working on a problem and come back to it later.
7. **Reset.** You can specify whether the **Reset** option is available for a problem. This setting at the problem level overrides the default setting for the course in **Advanced Settings**.

   If the **Reset** option is available, learners can select **Reset** to clear any input that has not yet been submitted, and try again to answer the question.

   - If the learner has already submitted an answer, selecting **Reset** clears the submission and, if the problem includes a Python script to randomize variables and the randomization setting is **On Reset**, changes the values the learner sees in the problem.
   - If the problem has already been answered correctly, **Reset** is not available.
   - If the number of **Maximum Attempts** that was set for this problem has been reached, **Reset** is not available.
8. **Show Answer.** You can specify whether this option is available for a problem. If a learner selects **Show Answer**, the learner sees both the correct answer and the explanation, if any.
9. **Feedback.** After a learner selects **Submit**, an icon appears beside each response field or selection within a problem. A green check mark indicates that the response was correct, a green asterisk indicates that the response was partially correct, and a red X indicates that the response was incorrect. Underneath the problem, feedback text indicates whether the problem was answered correctly, incorrectly, or partially correctly, and shows the problem score.



In addition to the items above, which are shown in the example, problems also have the following elements.

- **Correct answer.** Most problems require that you specify a single correct answer.
- **Explanation.** You can include an explanation that appears when a learner selects **Show Answer**.
- **Grading.** You can specify whether a group of problems is graded. If a group of problems is graded, an icon of a pen and a piece of paper is shown for that assignment in the course navigation pane.
- **Due date.** The date that the problem is due. Learners cannot submit answers for problems whose due dates have passed, although they can select **Show Answer** to show the correct answer and the explanation, if any.

Note: Problems can be **open** or **closed**. Closed problems, such as problems whose due dates are in the past, do not accept further responses and cannot be reset. Learners can still see questions, solutions, and revealed explanations, but they cannot submit responses or reset problems.

There are also some attributes of problems that are not immediately visible. You can set these attributes in Studio.

- **Accessible Label.** Within the problem text, you can identify the text that is, specifically, the question that learners need to answer. The text that is labeled as the question is used by screen readers, reports, and Insights. For more information, see The Simple Editor.
- **Randomization.** In certain types of problems, you can include a Python script to randomize the values that are presented to learners. You use this setting to define when values are randomized. For more information, see *Randomization*.
- **Weight.** Different problems in a particular problem set can be given different weights. For more information, see *Problem Weight*.

Problem Settings

In addition to the text of the problem, problems that you create have the following settings.

- *Display Name*
- *Maximum Attempts*
- *Problem Weight*
- *Randomization*
- *Show Answer*
- *Show Reset Button*

This section describes the OLX elements and attributes that you define for the problem settings. For a detailed description of each setting, see Defining Settings for Problem Components.

Display Name

Using OLX, you set the display name as an attribute of the `<problem>` element.

```
<problem display_name="Geography Homework"></problem>
```

Maximum Attempts

Using OLX, you set the maximum attempts as an attribute of the `<problem>` element.

```
<problem max_attempts="3"></problem>
```

Problem Weight

Using OLX, you set the component weight as an attribute of the `<problem>` element.

```
<problem weight="2.0"></problem>
```

Randomization

Using OLX, you set value randomization as an attribute of the `<problem>` element.

```
<problem rerandomize="always"></problem>
```

You can specify the following values for this attribute.

- always
- on_reset
- never
- per_student

Show Answer

Using OLX, you set the show answer preference as an attribute of the `<problem>` element.

```
<problem showanswer="correct_or_past_due"></problem>
```

You can specify the following values for this attribute.

- always
- answered
- attempted
- closed
- correct_or_past_due
- finished
- past_due
- never

Show Reset Button

Using OLX, you set the show reset button preference as an attribute of the `<problem>` element.

```
<problem show_reset_button="true"></problem>
```

Modifying a Released Problem

> **Warning:** Be careful when you modify problems after they have been released! Changes that you make to published problems can affect the learner experience in the course and analysis of course data.

After a learner submits a response to a problem, the LMS stores the learner's response, the score that the learner received, and the maximum score for the problem. For problems with a **Maximum Attempts** setting greater than 1, the LMS updates these values each time the learner submits a new response to a problem. However, if you change a problem or its attributes, existing learner information for that problem is not automatically updated.

For example, you might release a problem and specify that its answer is 3. After some learners have submitted responses, you notice that the answer should be 2 instead of 3. When you update the problem with the correct answer, the LMS does not update scores for learners who answered 2 for the original problem and thus received the wrong score.

For another example, you might change the number of response fields to three. Learners who submitted answers before the change have a score of 0, 1, or 2 out of 2.0 for that problem. Learners who submitted answers after the change have scores of 0, 1, 2, or 3 out of 3.0 for the same problem.

If you change the weight setting for the problem in Studio, however, existing learner scores update when the learner's **Progress** page is refreshed. In a live section, learners will see the effect of these changes.

Workarounds

If you have to modify a released problem in a way that affects grading, you have two options within Studio to assure that every learner has the opportunity to submit a new response and be regraded. Note that both options require you to ask your learners to go back and resubmit answers to a problem.

- In the problem component that you changed, increase the number of attempts for the problem. Then ask all your learners to redo the problem.
- Delete the entire Problem component in Studio and create a new Problem component with the content and settings that you want. (If the revisions you must make are minor, duplicate the problem component before you delete it and revise the copy.) Then ask all your learners to complete the new problem.

Multiple Problems in One Component

You can create a problem that includes more than one response type. For example, you might want to create a numerical input problem and then include a multiple choice problem that refers to the numerical input problem. Or, you might want a learner to be able to check the answers to many problems at one time. To do this, you can include multiple problems inside a single `<problem>` element. The problems can be different types.

Each question and its answer options are enclosed by the element that identifies the type of problem, such as `<multiplechoiceresponse>` for a multiple choice question or `<formularesponse>` for a math expression input question.

You can provide a different explanation for each question by using the `<solution>` element.

As a best practice, edX recommends that you avoid including unformatted paragraph text between the questions. Screen readers can skip over text that is inserted among multiple questions.

Elements such as the **Submit**, **Show Answer**, and **Reset** buttons, as well as the settings that you select for the problem component, apply to all of the problems in that component. Thus, if you set the maximum number of attempts to 3, the learner has three attempts to answer the entire set of problems in the component as a whole rather than three attempts to answer each problem individually. If a learner selects **Submit**, the LMS scores all of the problems in the component at once. If a learner selects **Show Answer**, the answers for all the problems in the component appear.

Note: You cannot use a *Custom JavaScript Display and Grading Problem* in a component that contains more than one problem. Each custom JavaScript problem must be in its own component.

Adding Feedback and Hints to a Problem

You can add feedback, hints, or both to the following core problem types.

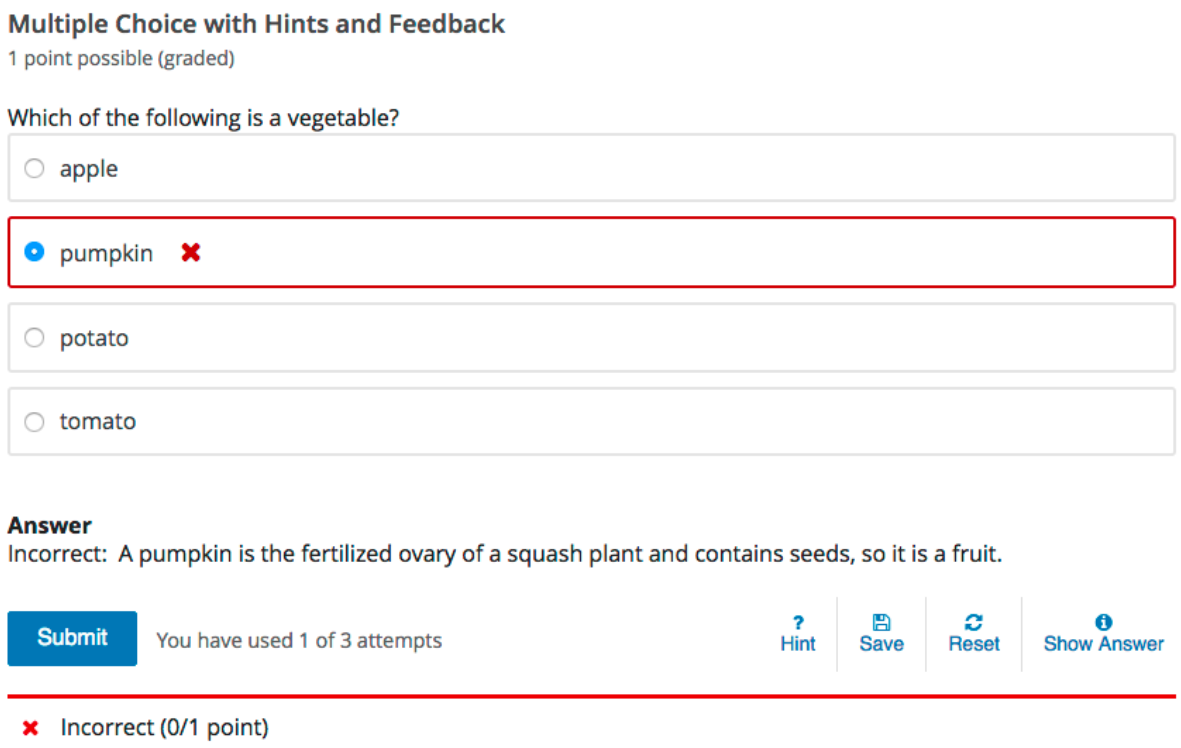
- *Checkbox Problem*
- *Dropdown Problem*
- *Multiple Choice Problem*
- *Numerical Input Problem*
- *Text Input Problem*

By using hints and feedback, you can provide learners with guidance and help as they work on problems.

Feedback in Response to Attempted Answers

You can add feedback that displays to learners after they submit an answer.

For example, the following multiple choice problem provides feedback in response to the selected option when the learner selects **Submit**. In this case, feedback is given for an incorrect answer.



Best Practices for Providing Feedback

The immediacy of the feedback available to learners is a key advantage of online instruction and difficult to do in a traditional classroom environment.

You can target feedback for common incorrect answers to the misconceptions that are common for the level of the learner (for example, elementary, middle, high school, college).

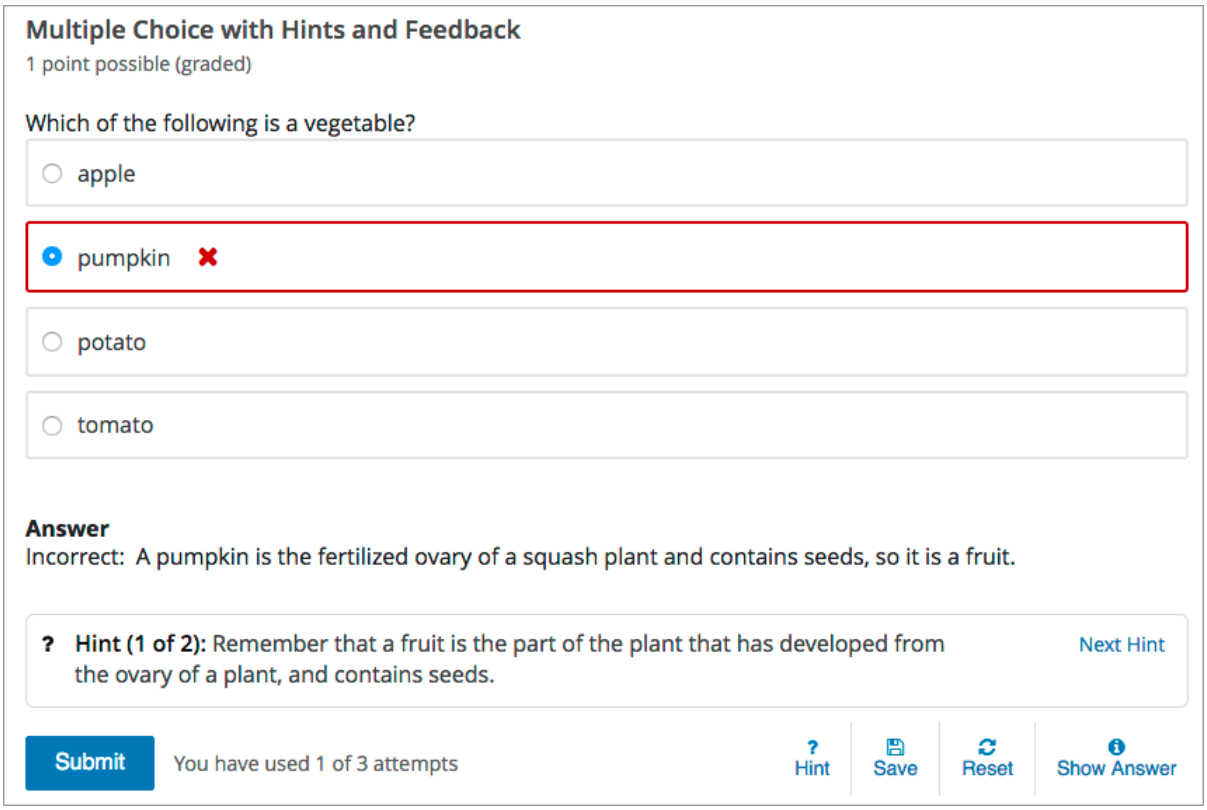
In addition, you can create feedback that provides some guidance to the learner about how to arrive at the correct answer. This is especially important in text input and numeric input problems, because without such guidance, learners might not be able to proceed.

You should also include feedback for the correct answer to reinforce why the answer is correct. Especially in questions where learners are able to guess, such as multiple choice and dropdown problems, the feedback should provide a reason why the selection is correct.

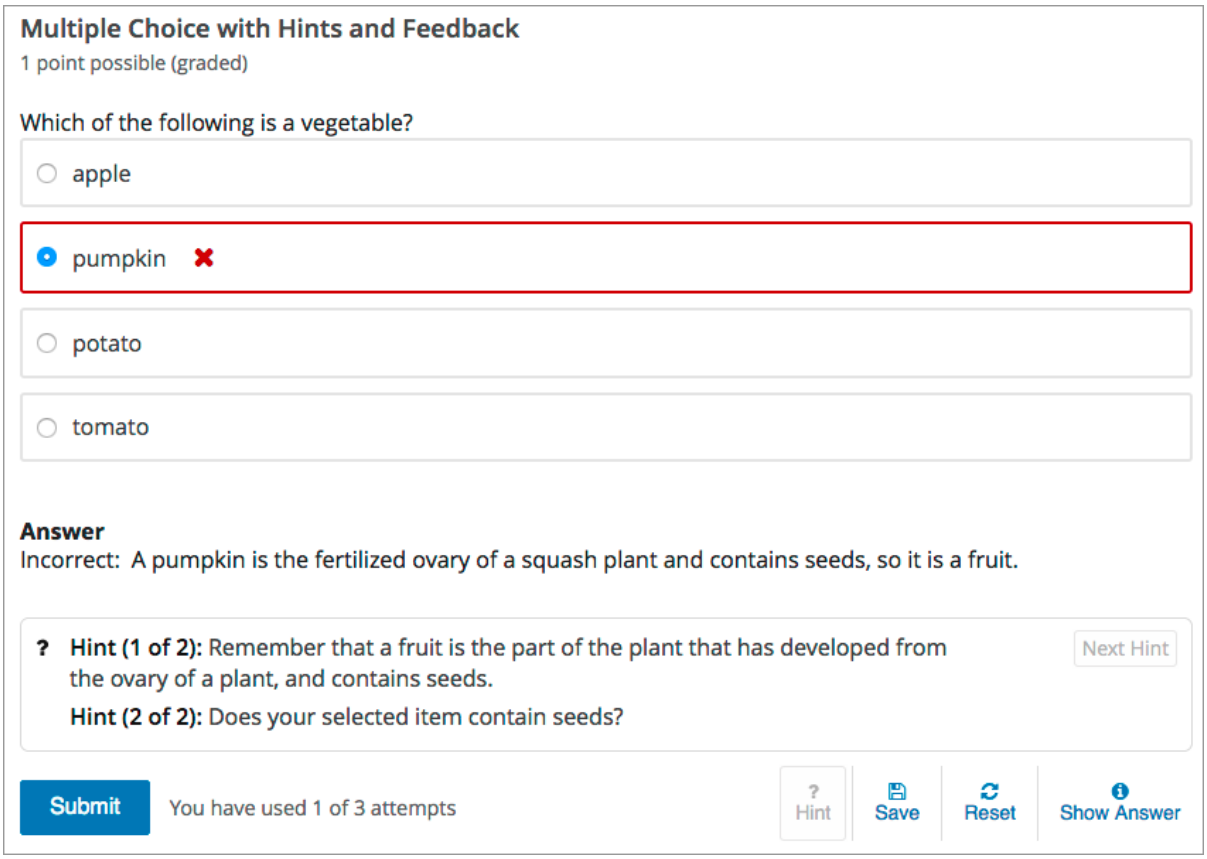
Providing Hints for Problems

You can add one or more hints that are displayed to learners. When you add hints, the **Hint** button is automatically displayed to learners. Learners can access the hints by selecting **Hint** beneath the problem. A learner can view multiple hints by selecting **Hint** multiple times.

For example, in the following multiple choice problem, the learner selects **Hint** after having made one incorrect attempt.



The hint text indicates that it is the first of two hints. After the learner selects **Next Hint**, both of the available hints appear. When all hints have been used, the **Hint** or **Next Hint** option is no longer available.



Best Practices for Providing Hints

To ensure that your hints can assist learners with varying backgrounds and levels of understanding, you should provide multiple hints with different levels of detail.

For example, the first hint can orient the learner to the problem and help those struggling to better understand what is being asked.

The second hint can then take the learner further towards the answer.

In problems that are not graded, the third and final hint can explain the solution for learners who are still confused.

Create Problems with Feedback and Hints

You create problems with feedback and hints in Studio. Templates with feedback and hints configured are available to make creating your own problems easier.

While editing a unit, in the **Add New Component** panel, select **Problem**. In the list that opens, select **Common Problem Types**. Templates for problems with feedback and hints are listed.

Add the problem type you need to the unit, then edit the component. The exact syntax you use to configure hints and feedback depends on the problem type. See the topic for the problem type for more information.

- *Checkbox Problem*
- *Dropdown Problem*
- *Multiple Choice Problem*

- *Numerical Input Problem*
- *Text Input Problem*

Awarding Partial Credit for a Problem

You can configure the following problem types so that learners can receive partial credit for a problem if they submit an answer that is partly correct.

- *Checkbox*
- *Multiple Choice*
- *Numerical Input*
- *Write-Your-Own Grader*

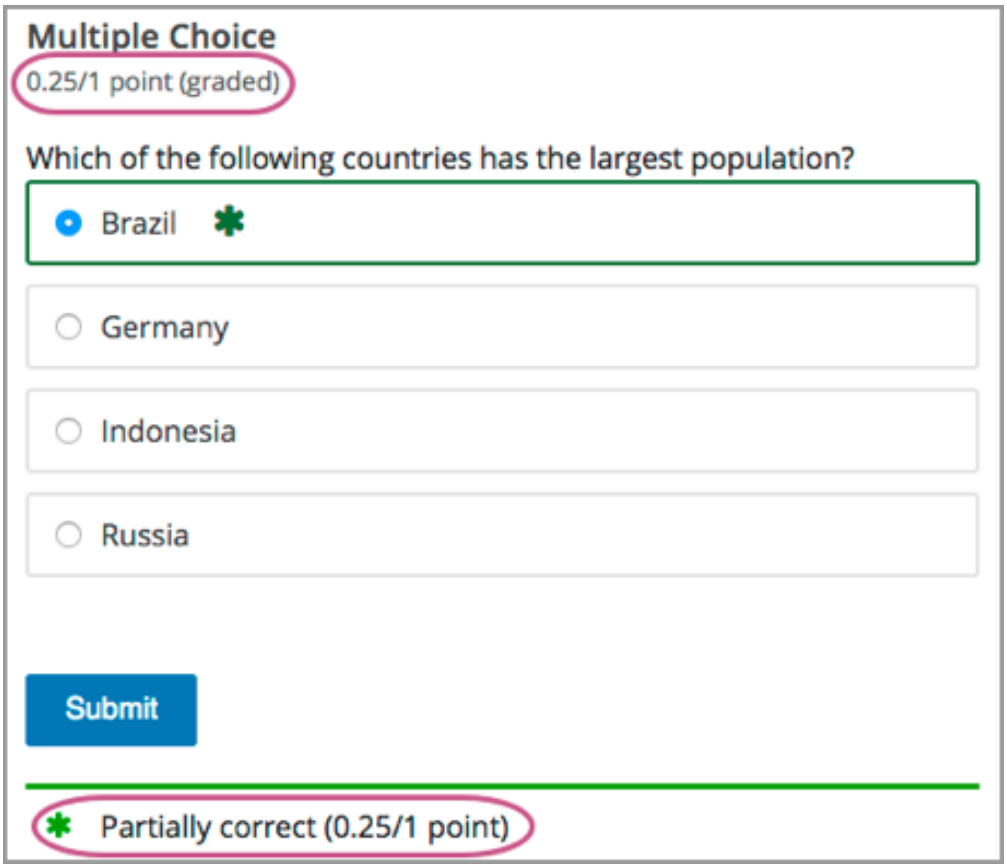
By awarding partial credit for problems, you can motivate learners who have mastered some of the course content and provide a score that accurately demonstrates their progress.

For more information about configuring partial credit, see the topic for each problem type.

How Learners Receive Partial Credit

Learners receive partial credit when they submit an answer in the LMS.

In the following example, the course team configured a multiple choice problem to award 25% of the possible points (instead of 0 points) for one of the incorrect answer options. The learner selected this incorrect option, and received 25% of the possible points.



Partial Credit and Reporting on Learner Performance

When a learner receives partial credit for a problem, the LMS only adds the points that the learner earned to the grade. However, the LMS reports any problem for which a learner receives credit, in full or in part, as correct in the following ways.

- Events that the system generates when learners receive partial credit for a problem indicate that the answer was correct. Specifically, the `correctness` field has a value of `correct`.

  For more information about events, see Problem Interaction Events in the *EdX Research Guide*.

- The **AnswerValue** in the Student Answer Distribution report is **1**, for correct.
- The edX Insights insights:student performance reports count the answer as correct.

Course teams can see that a learner received partial credit for a problem in the learner's submission history. The submission history shows the score that the learner received out of the total available score, and the value in the `correctness` field is `partially-correct`. For more information, see Learner Answer Submissions.

Problem Randomization

Presenting different learners with different problems or with different versions of the same problem is referred to as "problem randomization".

You can provide different learners with different problems by using randomized content blocks, which randomly draw problems from pools of problems stored in content libraries. For more information, see Randomized Content Blocks.

Note: Problem randomization is different from the **Randomization** setting in Studio. Problem randomization offers different problems or problem versions to different learners, whereas the **Randomization** setting controls when a Python script randomizes the variables within a single problem. For more information about the **Randomization** setting, see Randomization.

Create Randomized Problems

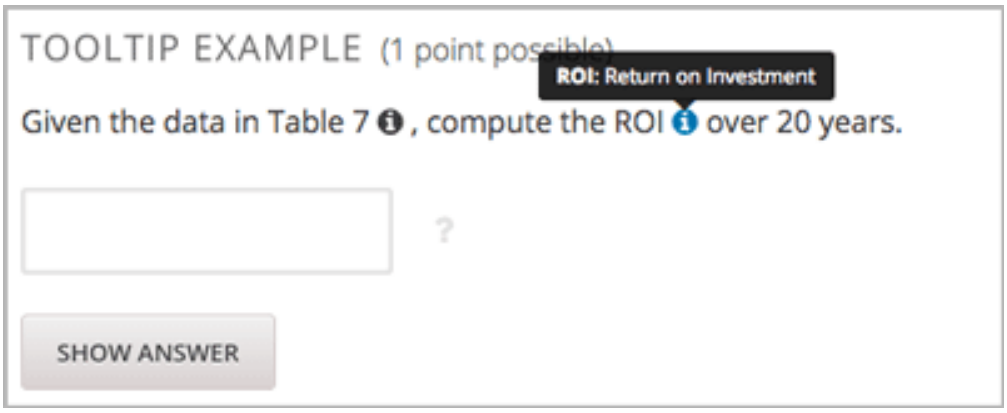
Note: Creating randomized problems by exporting your course and editing some of your course's XML files is no longer supported.

You can provide different learners with different problems by using randomized content blocks, which randomly draw problems from pools of problems stored in content libraries. For more information, see Randomized Content Blocks.

Adding Tooltips to a Problem

To help learners understand terminology or other aspects of a problem, you can add inline tooltips. Tooltips show text to learners when they move their cursors over a tooltip icon.

The following example problem includes two tooltips. The tooltip that provides a definition for "ROI" is being shown.



Note: For learners using a screen reader, the tooltip expands to make its associated text accessible when the screen reader focuses on the tooltip icon.

To add the tooltip, you wrap the text that you want to appear as the tooltip in the `clarification` element. For example, the following problem contains two tooltips.

```
<problem>
    <text>
        <p>Given the data in Table 7 <clarification>Table 7: "Example PV
          Installation Costs", Page 171 of Roberts textbook</clarification>,
          compute the ROI <clarification><strong>ROI</strong>: Return on
          Investment</clarification> over 20 years.
        </p>
     . . .
```

CHAPTER 11

Exercises, Tools, and Problem Types

This section describes the various exercises, tools, and problem types that you can add to your course.

For information about the level of support that edX has designated for each exercise, tool, or problem type, see *Levels of Support*.

Levels of Support

The level of support that edX provides for each exercise, tool, or problem type varies. The levels of support are categorized as full, provisional, or no support. This table provides the definition for each level of support. In this guide, each topic that describes each exercise, tool, or problem type contains a note that indicates the edX level of support for that exercise, tool, or problem type.

Level of Support	Description
Full support	Fully supported tools and features are available on edx.org, are fully tested, have user interfaces where applicable, and are documented in the official edX guides that are available on docs.edx.org.
Provisional support	Provisionally supported tools and features are available on edx.org, but might lack the robustness of functionality that your courses require. Third party tools are classified as provisionally supported because edX does not have control over the quality of the software, or of the content that can be provided using such tools. You should test provisionally supported tools thoroughly before using them in your course, especially in graded sections. Complete documentation might not be available for provisionally supported tools, or documentation might be available from sources other than the official edX guides.
Not supported	Exercises and tools with no support are not maintained by edX, and might be deprecated in the future. They are not recommended for use in courses due to noncompliance with one or more of the base requirements, such as testing, accessibility, internationalization, and documentation.

Annotation Problem

Note: EdX does not support this problem type.

In an annotation problem, you highlight specific text inside a larger text block and then ask questions about that text. The questions appear when learners move their cursors over the highlighted text. The questions also appear in a section below the text block, along with space for learners' responses.

Instructions Collapse Instructions ↑

Annotation Exercise, Hour 12 (to be done after the Hour 12 Question Set has been completed). Carefully read the following text and answer the accompanying question. To see the question while you read the text, hover your mouse over the highlighted selection in the text below.

Guided Discussion Hide Annotations

This exercise first appeared in HarvardX's CB22x: The Ancient Greek Hero course, spring 2013.

| 122 And they [= the Golden Generation of humankind] are superhumans [daimones]. They exist because of the Will of Zeus. | 123 They are the good, the earthbound [epi-khthonioi], the guardians of mortal humans. | 124 They guard acts of justice [dikē] and they guard against wretched acts of evil. | 125 Enveloped in mist, they roam everywhere throughout the earth. | 126 They are givers of prosperity. And they had this as a privilege [geras], a kingly one [basilēion].

QUESTION 1 (2/2 points)

Annotation Exercise Return to Annotation ↑

| 124 They guard acts of justice [dikē] and they guard against wretched acts of evil.

Hesiodic poetry declares that heroes are protectors of justice. Is this declaration consistent with what we have read in Homeric poetry?

Type your response below:

In your response to this question, which tag below do you choose?

- This declaration in Hesiodic poetry is inconsistent with Homeric poetry, since we see that heroes in the Iliad and Odyssey do in fact engage in unjust as well as just actions.
- ✔ This declaration in Hesiodic poetry is consistent with Homeric poetry, since there is no real contradiction here between the two kinds of poetry.
- We cannot be sure whether this declaration in Hesiodic poetry is consistent or inconsistent with Homeric poetry, where the actions of heroes are not evaluated in terms of injustice or justice.

Answer: This declaration in Hesiodic poetry is consistent with Homeric poetry, since there is no real contradiction here between the two kinds of poetry.

If you answered "This declaration in Hesiodic poetry is inconsistent with Homeric poetry, since we see that heroes in the Iliad and Odyssey do in fact engage in unjust as well as just actions," you would be missing the point. The heroes in the Iliad and Odyssey are still alive when they are engaged in their actions. By contrast, the heroes of the Golden Age are not of this world when they are protecting justice. Enveloped in mist and roaming throughout the earth, the heroes of the Golden Age are pictured as otherworldly spirits. This picture contrasts with the depiction of heroes in the Iliad and Odyssey. If you answered "We

Enable Annotation Problems

Before you can add annotation problems to your course, you must enable annotation problems in Studio.

To enable annotation problems in Studio, you add the `"annotatable"` key to the **Advanced Module List** on the **Advanced Settings** page. (Be sure to include the quotation marks around the key value.) For more information, see Enabling Additional Exercises and Tools.

Create an Annotation Problem

To create an annotation problem, you add the **Instructions** and **Guided Discussion** segments of the problem, and then the **Annotation problem** segment of the problem.

Add Instructions and Guided Discussion

To add the **Instructions** and **Guided Discussion** segments of the problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Advanced**.
2. In the list of problem types, select **Annotation**.
3. In the component that appears, select **Edit**.
4. In the component editor, replace the example code with your own code.
5. Select **Save**.

Add Annotation Problem

To add the **Annotation problem** segment of the problem, follow these steps.

1. Under the annotation component, create a new blank advanced problem component.
2. Paste the following code in the advanced problem component, replacing placeholders with your own information.

```
<problem>
    <annotationresponse>
        <annotationinput>
          <text>PLACEHOLDER: Text of annotation</text>
            <comment>PLACEHOLDER: Text of question</comment>
            <comment_prompt>PLACEHOLDER: Type your response below:</comment_
↪prompt>
            <tag_prompt>PLACEHOLDER: In your response to this question, which tag␣
↪below do you choose?</tag_prompt>
          <options>
            <option choice="incorrect">PLACEHOLDER: Incorrect answer (to make␣
↪this option a correct or partially correct answer, change choice="incorrect" to␣
↪choice="correct" or choice="partially-correct")</option>
            <option choice="correct">PLACEHOLDER: Correct answer (to make this␣
↪option an incorrect or partially correct answer, change choice="correct" to␣
↪choice="incorrect" or choice="partially-correct")</option>
```

```
            <option choice="partially-correct">PLACEHOLDER: Partially correct␣
↪answer (to make this option a correct or partially correct answer, change␣
↪choice="partially-correct" to choice="correct" or choice="incorrect")
            </option>
          </options>
        </annotationinput>
    </annotationresponse>
    <solution>
      <p>PLACEHOLDER: Detailed explanation of solution</p>
    </solution>
  </problem>
```

3. Select **Save**.

Checkbox Problem

Note: EdX offers full support for this problem type.

The checkbox problem type is a core problem type that can be added to any course. At a minimum, checkbox problems include a question or prompt and several answer options. By adding hints, feedback, or both, you can give learners guidance and help when they work on a problem.

For more information about the core problem types, see Working with Problem Components.

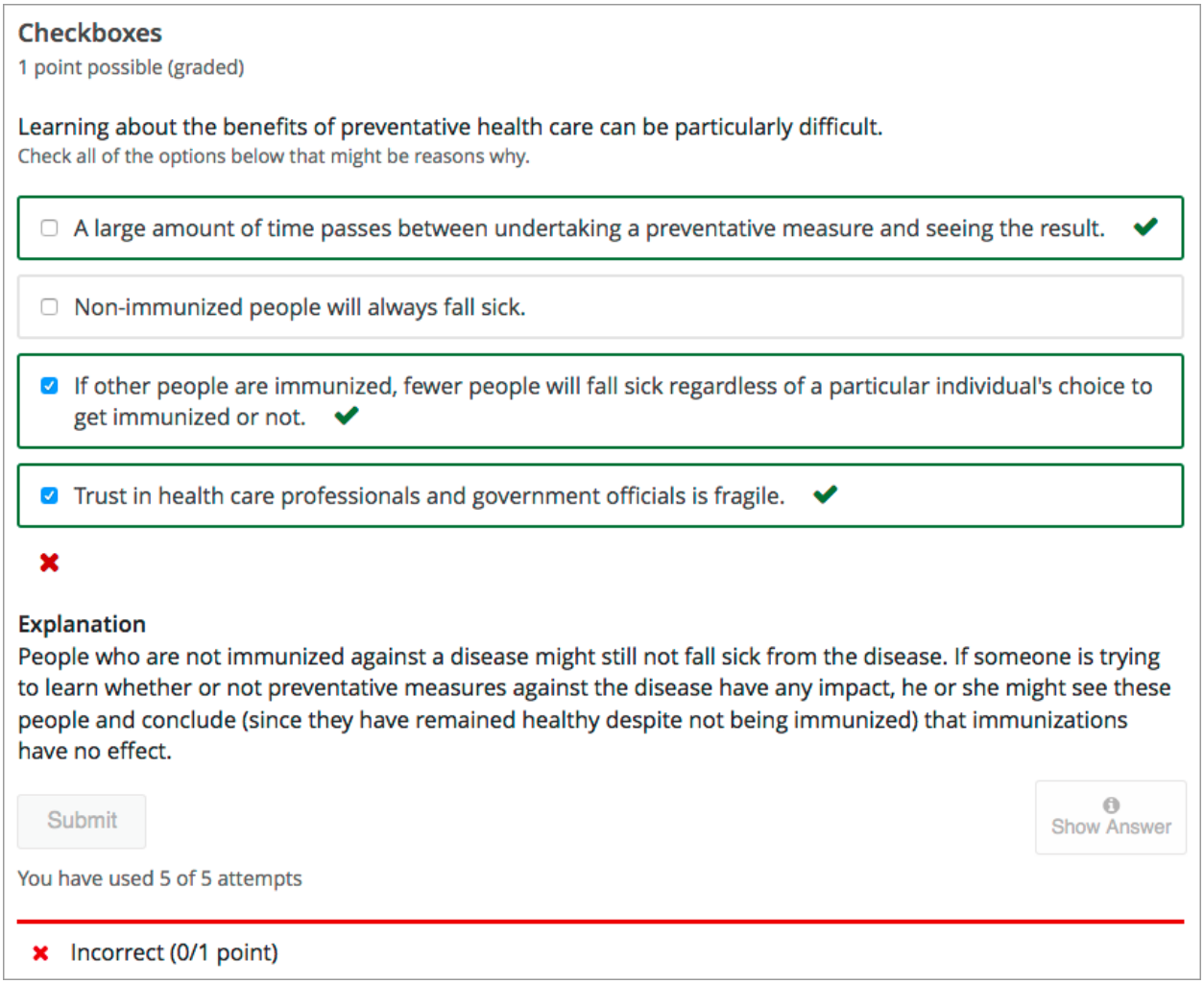
Overview

In checkbox problems, learners select one or more options from a list of possible answers. To answer the problem correctly, a learner must select all of the options that are correct answers, and none of the options that are incorrect. The course team must set up each checkbox problem to have at least one correct answer.

As a best practice, be sure that all of the answer choices are unambiguous, and avoid trick questions. Checkbox problems with ambiguity can be frustrating to learners, especially if the problems have a limited number of attempts.

Example Checkbox Problem

In the LMS, learners complete a checkbox problem by selecting the answer options that they believe are correct as well as leaving unselected the answer options that they believe are incorrect. An example of a completed checkbox problem follows.



This problem was incorrectly answered because the learner selected only two of the three required answer options. This example also shows that the learner selected **Show Answer** to reveal the correct answer and an explanation.

To add the example problem illustrated above, in Studio you use the simple editor to enter the following text and Markdown formatting.

```
>>Learning about the benefits of preventative health care can be particularly␣
↪difficult.||Check all of the options below that might be reasons why.<<

[x] A large amount of time passes between undertaking a preventative measure
and seeing the result.
[ ] Non-immunized people will always fall sick.
[x] If others are immunized, fewer people will fall sick regardless of a
particular individual's choice to get immunized or not.
[x] Trust in health care professionals and government officials is fragile.

[explanation]
People who are not immunized against a disease might still not fall sick
from the disease. If someone is trying to learn whether or not preventative
measures against the disease have any impact, he or she might see these
people and conclude, since they have remained healthy despite not being
immunized, that immunizations have no effect. Consequently, he or she would
tend to believe that immunization (or other preventative measures) have
```

```
fewer benefits than they actually do.
[explanation]
```

The OLX (open learning XML) markup for this example checkbox problem follows.

```
<problem>
  <choiceresponse>
    <label>Learning about the benefits of preventative health care can be
    particularly difficult.</label>
    <description>Check all of the options below that might be reasons why.</
↪description>
    <checkboxgroup>
      <choice correct="true">A large amount of time passes between
       undertaking a preventative measure and seeing the result.</choice>
      <choice correct="false">Non-immunized people will always fall sick.</choice>
      <choice correct="true">If others are immunized, fewer people will fall
       sick regardless of a particular individual's choice to get immunized
       or not.</choice>
      <choice correct="true">Trust in health care professionals and
       government officials is fragile.</choice>
    </checkboxgroup>
    <solution>
      <div class="detailed-solution">
        <p>Explanation</p>
        <p>People who are not immunized against a disease might still not
         fall sick from the disease. If someone is trying to learn whether
         or not preventative measures against the disease have any impact,
         he or she might see these people and conclude, since they have
         remained healthy despite not being immunized, that immunizations
         have no effect. Consequently, he or she would tend to believe that
         immunization (or other preventative measures) have fewer benefits
         than they actually do.</p>
      </div>
    </solution>
  </choiceresponse>
</problem>
```

Analyzing Performance on Checkbox Problems

For the checkbox problems in your course, you can use edX Insights to review aggregated learner performance data and examine submitted answers. For more information, see Using edX Insights.

Adding a Checkbox Problem

You add checkbox problems in Studio by selecting the **Problem** component type and then using either the simple editor or the advanced editor to specify the prompt and the answer options.

- *Use the Simple Editor to Add a Checkbox Problem*
- *Use the Advanced Editor to Add a Checkbox Problem*

Note: You can begin work on the problem in the simple editor, and then switch to the advanced editor. However,

after you save any changes you make in the advanced editor, you cannot switch back to the simple editor.

Use the Simple Editor to Add a Checkbox Problem

When you add a checkbox problem, you can choose one of these templates.

- **Checkboxes**
- **Checkboxes with Hints and Feedback**

These templates include the Markdown formatting that you use in the simple editor to add a problem without, or with, hints and feedback. To use the simple editor to add a problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Problem**.
2. From the list of **Common Problem Types**, select the type of problem you want to add. Studio adds a template for the problem to the unit.
3. Select **Edit**. The simple editor opens to a template that shows the Markdown formatting that you use for this problem type.
4. Replace the guidance provided by the template to add your own text for the question or prompt, answer options, explanation, and so on.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Use the Advanced Editor to Add a Checkbox Problem

You can use the advanced editor to identify the elements of a checkbox problem with OLX. For more information, see *Checkbox Problem OLX Reference*. To use the advanced editor to add a problem, follow these steps.

1. Follow steps 1-3 for creating the problem in the simple editor.
2. Select **Advanced Editor**. The advanced editor opens the template and shows the OLX markup that you can use for this problem type.
3. Replace the guidance provided by the template to add your own text. For example, replace the question or prompt, answer options, and explanation.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.
4. Update the OLX to add optional elements and attributes required for your problem.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Adding Feedback to a Checkbox Problem

For an overview of feedback in problems, see *Adding Feedback and Hints to a Problem*. For checkbox problems, you can add feedback for each of the answer options you provide in the problem. You can also identify different combinations of answer options that learners are likely to select, and add compound feedback for those combinations.

You can add feedback to a checkbox problem using the simple editor or the advanced editor.

Adding Feedback for Individual Options

In checkbox problems, you can provide feedback for each option that a learner can select, with distinct feedback depending on whether or not the learner selects that option. This means that there are several possible types of feedback.

- The learner selects a correct option. This type of feedback should indicate why the option is correct.
- The learner does not select a correct option. This type of feedback should indicate that the learner missed checking this option and why it is correct.
- The learner selects an incorrect option. This type of feedback should indicate that the learner incorrectly checked this option and why it is incorrect.
- The learner does not select an incorrect option. This type of feedback should reinforce why the learner correctly left this option unselected.

Adding Compound Feedback

You can configure the checkbox problem to provide compound feedback. Compound feedback is feedback given for a specific combination of options. For example, if you have three possible option in the problem, you can define specific feedback for when a learner selects each combination of possible options.

- A
- B
- C
- A, B
- B, C
- A, C
- A, B, C

For problems with more than three options, providing specific feedback for each combination can become difficult. For such problems, you might choose to define compound feedback for more likely combinations of option or for combinations of option that reflect common learner misunderstandings. If you do not define feedback for a combination that a learner selects, the learner receives feedback for the individual selections.

Configuring Feedback in the Simple Editor

You can configure individual option or compound feedback in the simple editor. When you create a new checkbox problem, select the template **Checkboxes with Hints and Feedback**. This template has example formatted feedback that you can replace with your own text.

Configure Feedback for Individual Options

In the simple editor, you configure individual option feedback with the following Markdown formatting.

```
[x] answer  {{ selected: Feedback when learner selects this option. },
{unselected: Feedback when the learner does not select this option.}}
```

Note: You can use `S` for `selected` and `U` for unselected.

For example, the following problem has feedback for every answer option, whether learners select a given option or leave it unselected.

```
>>Which of the following is an example of a fruit?||Select all that apply.<<

[x] apple {{ selected: You are correct that an apple is a fruit because it
is the fertilized ovary that comes from an apple tree and contains seeds. },
{ unselected: Remember that an apple is also a fruit.}}

[x] pumpkin {{ selected: You are correct that a pumpkin is a fruit because it
is the fertilized ovary of a squash plant and contains seeds.}, { unselected:
Remember that a pumpkin is also a fruit.}}

[ ] potato {{ U: You are correct that a potato is a vegetable because it is
an edible part of a plant in tuber form.}, { S: A potato is a vegetable, not
a fruit, because it does not come from the flower on a plant or tree and does
not contain seeds.}}

[x] tomato {{ S: You are correct that a tomato is a fruit because it is the
fertilized ovary of a tomato plant and contains seeds. }, { U: Many people
mistakenly think a tomato is a vegetable. However, because a tomato is the
fertilized ovary of a tomato plant and contains seeds it is classified as a
fruit.}}
```

Configure Compound Feedback

In the simple editor, you configure compound feedback after the possible options, with the following syntax.

```
{{ ((Answer Combination)) Feedback when learner selects this combination of
options.}}
```

For example, the following compound feedback is used when learners select options **A, B, and D** or **A, B, C, and D**.

```
{{ ((A B D)) An apple, pumpkin, and tomato are all fruits as they are all the
fertilized ovaries of a plant and contain seeds. }}

{{ ((A B C D)) You are correct that an apple, pumpkin, and tomato are all
fruits as they are all the fertilized ovaries of a plant and contain seeds.
However, a potato is not a fruit as it is an edible part of a plant in tuber
form and is classified as a vegetable.  }}
```

Note: If you configure individual option feedback for every answer, and you also provide compound feedback, when learners select the exact combination of answer choices defined, they only see the compound feedback. In this example, learners who select apple (A), pumpkin (B), and tomato (D) see the message "An apple, pumpkin, and tomato are all fruits as they are all the fertilized ovaries of a plant and contain seeds." They do not also see the individual feedback for selecting A, B, and D, and for leaving C unselected.

Configuring Feedback in the Advanced Editor

You can configure individual option and compound feedback in the advanced editor.

Configure Individual Option Feedback

In the advanced editor, you configure individual option feedback with the following syntax.

```
<choice correct="true">Choice label
  <choicehint selected="true">Feedback for when learner selects this
   answer.</choicehint>
  <choicehint selected="false">Feedback for when learner does not select
   this answer.</choicehint>
</choice>
```

For example, the following problem has feedback for each option, selected or unselected.

```
<problem>
  <choiceresponse>
    <label>Which of the following is an example of a fruit?</label>
    <description>Select all that apply.</description>
    <checkboxgroup>
      <choice correct="true">apple
        <choicehint selected="true">You are correct that an apple is a fruit
         because it is the fertilized ovary that comes from an apple tree and
         contains seeds.</choicehint>
        <choicehint selected="false">Remember that an apple is also a
         fruit.</choicehint>
      </choice>
      <choice correct="true">pumpkin
        <choicehint selected="true">You are correct that a pumpkin is a fruit
         because it is the fertilized ovary of a squash plant and contains
         seeds.</choicehint>
        <choicehint selected="false">Remember that a pumpkin is also a
         fruit. </choicehint>
      </choice>
      <choice correct="false">potato
        <choicehint selected="true">A potato is a vegetable, not a fruit,
         because it does not come from the flower on a plant or tree and does
         not contain seeds.</choicehint>
        <choicehint selected="false">You are correct that a potato is
         classified as a vegetable because it is an edible part of a plant in
         tuber form.</choicehint>
      </choice>
      <choice correct="true">tomato
        <choicehint selected="true">You are correct that a tomato is
         classified as a fruit because it is the fertilized ovary of a tomato
         plant and contains seeds.</choicehint>
        <choicehint selected="false">Many people mistakenly think a tomato is
         a vegetable. However, because a tomato is the fertilized ovary of a
         tomato plant and contains seeds it is classified as a fruit.</choicehint>
      </choice>
    </checkboxgroup>
  </choiceresponse>
</problem>
```

Configure Compound Feedback

In the advanced editor, you define compound feedback by adding a `<compoundhint>` element within the `<checkboxgroup>` element.

```
    .
    .
    .
  </choice>
  <compoundhint value="Answer Combination">Feedback when learner selects
   this combination of answers.</compoundhint>
</checkboxgroup>
```

For example, the following compound feedback is used when learners select options **A, B, and D** or **A, B, C, and D**.

```
    .
    .
    .
  </choice>
  <compoundhint value="A B D">An apple, pumpkin, and tomato are all
   fruits as they all are fertilized ovaries of a plant and contain
   seeds.</compoundhint>
  <compoundhint value="A B C D">You are correct that an apple, pumpkin,
   and tomato are all fruits as they all are fertilized ovaries of a
   plant and contain seeds. However, a potato is not a fruit as it is an
   edible part of a plant in tuber form and is classified as a vegetable.
  </compoundhint>
</checkboxgroup>
```

Adding Hints to a Checkbox Problem

You can add hints to a checkbox problem using the simple editor or the advanced editor. For an overview of hints in problems, see *Adding Feedback and Hints to a Problem*.

Configure Hints in the Simple Editor

In the simple editor, you configure hints with the following syntax.

```
||Hint 1||
||Hint 2||
||Hint n||
```

Note: You can configure any number of hints. The learner views one hint at a time and views the next one by selecting **Hint** again.

For example, the following problem has two hints.

```
||A fruit is the fertilized ovary from a flower.||
||A fruit contains seeds of the plant.||
```

Configure Hints in the Advanced Editor

In the advanced editor, you add the `<demandhint>` element immediately before the closing `</problem>` tag, and then configure each hint using the `<hint>` element.

```
.
.
.
<demandhint>
  <hint>Hint 1</hint>
  <hint>Hint 2</hint>
  <hint>Hint 3</hint>
</demandhint>
</problem>
```

For example, the following OLX for a multiple choice problem shows two hints.

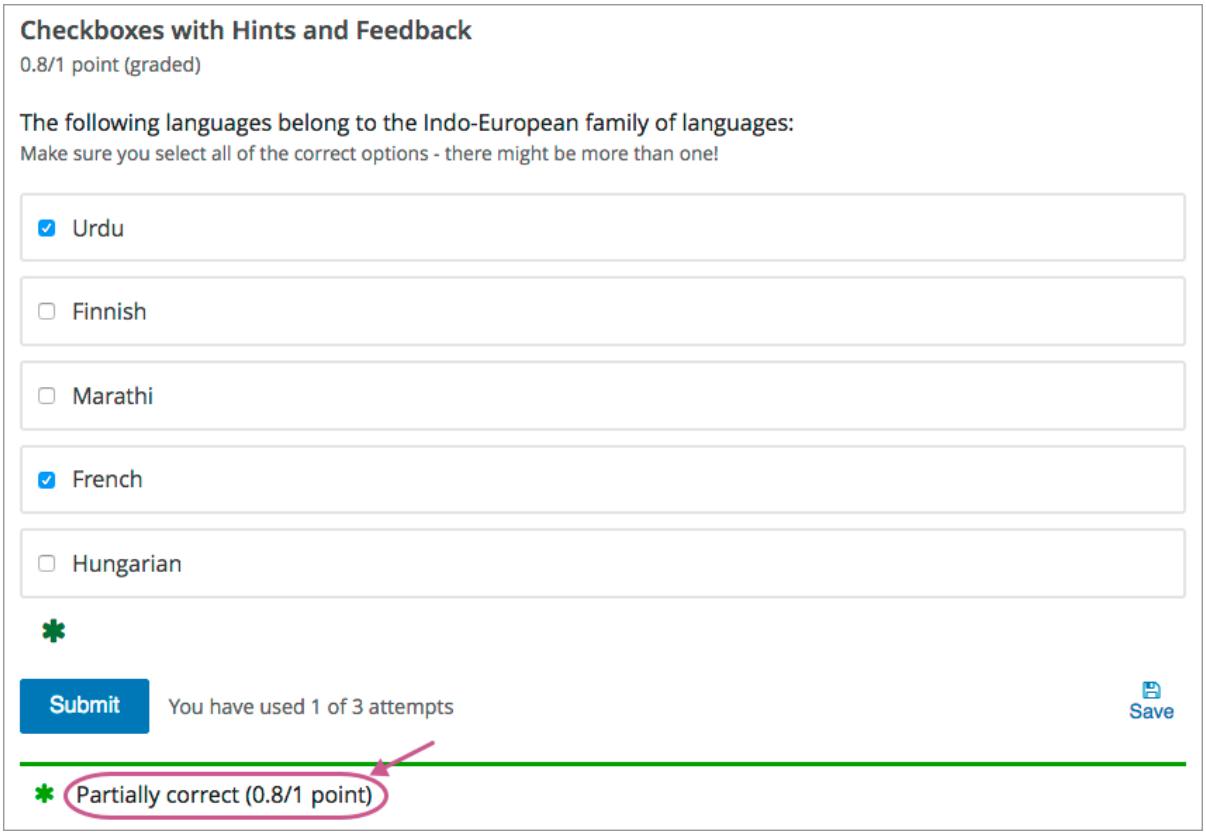
```
.
.
.
</multiplechoiceresponse>
<demandhint>
  <hint>A fruit is the fertilized ovary from a flower.</hint>
  <hint>A fruit contains seeds of the plant.</hint>
</demandhint>
</problem>
```

Awarding Partial Credit in a Checkbox Problem

You can configure a checkbox problem to award partial credit to learners who submit an answer that is partly correct. You must use the advanced editor to configure partial credit.

For an overview of partial credit in problems, see *Awarding Partial Credit for a Problem*.

In the following example, the learner selected two of the three correct choices, and did not select any incorrect choices. The learner therefore had four out of five correct answers. Because the course team set this problem up to award partial credit for every correct answer selected and every incorrect answer left unselected (known as *every decision counts*), the learner earned 80% of the points for this problem.



You can use the following methods to award partial credit in a checkbox problem.

Using the Every Decision Counts Method

You can configure a checkbox problem so that the learner's response for every option is evaluated and scored. This method is known as every decision counts (EDC).

With EDC, for each option the learner gets wrong, either by not selecting a correct option or selecting an incorrect option, the learner's score is reduced by 1/n, where "n" is the number of options.

For example, if there are four options, each one is worth 25% of the total score. If a learner's response is wrong for one option, she receives 75% of the points for the problem.

The following table describes the learner's score for different submissions for EDC problems with a variety of correct answer options.

Learner's Selections	Correct Answers	Incorrect Answers	Score
A, B, C	A, B, D	C	75%
A	A, C, D	B	75%
A, C	A, D	B, C	50%
C, D		A, B, C, D	0%

Configure an EDC Checkbox Problem

To configure an EDC checkbox problem, you add the `partial_credit="EDC"` attribute to the `<choiceresponse>` element in the problem OLX.

For example, the following OLX shows the checkbox problem template after it is updated to provide partial credit.

```
<problem>
  <choiceresponse partial_credit="EDC">
    <label>Which of the following is a fruit?</label>
    <description>Select all that apply.</description>
    <checkboxgroup>
      <choice correct="true">apple</choice>
      <choice correct="true">pumpkin</choice>
      <choice correct="false">potato</choice>
      <choice correct="true">tomato</choice>
    </checkboxgroup>
  </choiceresponse>
</problem>
```

Using the By Halves Method

You can configure a checkbox problem so that for every option that a learner gets wrong, either by not selecting a correct option or by selecting an incorrect option, half of the remaining points are deducted from the learner's score. This method is known as scoring by halves.

Note: By design, partial credit by halves requires the number of answer options to be more than twice the number of incorrect answers. In addition, partial credit is not given for more than two wrong answers, regardless of the total number of answer options. In other words, two wrong answers is scored at 25% only if there are at least 5 answer options. Three or more wrong answers is always scored at 0%, regardless of the number of total answer options.

Partial credit using the by halves method is calculated as follows.

- If a learner makes no errors, she receives full credit for the problem.
- If a learner makes one error, she receives 50% of the possible points, as long as there are three or more choices in the problem. If a learner makes one error and there are only two choices in the problem, no credit is given.
- If a learner makes two errors, she receives 25% of the possible points, as long as there are five or more choices in the problem. If a learner makes two errors and there are only three choices or four choices in the problem, no credit is given.
- If a learner makes three errors, she receives no credit for the problem, regardless of how many answer options there are.

The following tables illustrate partial credit score using the halves method, for problems with an increasing number of total answer options.

Number of Incorrect Answers	Number of Answer Options	Credit Given (%)
0	2	100
1	2	0
2	2	0

Number of Incorrect Answers	Number of Answer Options	Credit Given (%)
0	3	100
1	3	0
2	3	0
3	3	0

Number of Incorrect Answers	Number of Answer Options	Credit Given (%)
0	4	100
1	4	50
2	4	0
3	4	0
4	4	0

Number of Incorrect Answers	Number of Answer Options	Credit Given (%)
0	5	100
1	5	50
2	5	25
3	5	0
4	5	0
5	5	0

Number of Incorrect Answers	Number of Answer Options	Credit Given (%)
0	7	100
1	7	50
2	7	25
3	7	0
4	7	0
5	7	0

Configure a By Halves Checkbox Problem

To configure a by halves checkbox problem, you add the `partial_credit="halves"` attribute to the `<choiceresponse>` element in the problem OLX.

The following example shows a checkbox problem that provides partial credit by halves.

```
<problem>
  <choiceresponse partial_credit="halves">
    <label>Which of the following is a fruit?</label>
    <description>Select all that apply.</description>
    <checkboxgroup>
      <choice correct="true">apple</choice>
      <choice correct="true">pumpkin</choice>
      <choice correct="false">potato</choice>
      <choice correct="true">tomato</choice>
    </checkboxgroup>
  </choiceresponse>
</problem>
```

Checkbox Problem OLX Reference

Note: You can also set attributes and options by adding a <script> element. For more information, see *Using the Script Element*.

Template

```
<problem>
  <choiceresponse>
    <label>Question or prompt text</label>
    <description>Information about how to answer the question</description>
    <checkboxgroup>
      <choice correct="false">Answer option A (incorrect)</choice>
      <choice correct="true">Answer option B (correct)</choice>
      <choice correct="true">Answer option C (correct)</choice>
    </checkboxgroup>
  <solution>
    <div class="detailed-solution">
      <p>Optional header for the explanation or solution</p>
      <p>Optional explanation or solution text</p>
    </div>
  </solution>
  </choiceresponse>
  <demandhint>
    <hint>Hint 1</hint>
    <hint>Hint 2</hint>
  </demandhint>
</problem>
```

Elements

For checkbox problems, the <problem> element can include this hierarchy of child elements.

```
<choiceresponse>
    <label>
    <description>
    <checkboxgroup>
          <choice>
              <choicehint>
          <compoundhint>
    <solution>
<demandhint>
    <hint>
```

In addition, standard HTML tags can be used to format text.

<choiceresponse>

Required. Indicates that the problem is a checkbox problem.

Attributes

Attribute	Description
`partial_credit`	Optional. Specifies the type of partial credit given. `EDC` or `halves`.

Children

- `<label>`
- `<description>`
- `<checkboxgroup>`
- `<solution>`

`<label>`

Required. Identifies the question or prompt. You can include HTML tags within this element.

Attributes

None.

Children

None.

`<description>`

Optional. Provides clarifying information about how to answer the question. You can include HTML tags within this element.

Attributes

None.

Children

None.

`<checkboxgroup>`

Required. Indicates the beginning of the list of options.

Attributes

None.

Children

- <choice>
- <compoundhint>

<choice>

Required. Designates an answer option.

Attributes

Attribute	Description
correct	Indicates a correct or incorrect answer. • When set to "true", the choice is a correct answer. At least one required. • When set to "false", the choice is an incorrect answer.

Children

<choicehint>

<choicehint>

Optional. Specifies feedback for the answer.

Attributes

Attribute	Description
selected	Required. true or false. Indicates if the feedback is given when the answer option is selected, or when it is not selected.

Children

None.

<compoundhint>

Optional. Specifies feedback for a specific combination of answers.

Attributes

Attribute	Description
`value` (at least one required)	Indicates the combination of selected answers that triggers this feedback. Answers are identified by uppercase letters, in ascending alphabetical order.

Children

None.

<solution>

Optional. Identifies the explanation or solution for the problem, or for one of the questions in a problem that contains more than one question.

This element contains an HTML division `<div>`. The division contains one or more paragraphs `<p>` of explanatory text.

<demandhint>

Optional. Specifies hints for the learner. For problems that include multiple questions, the hints apply to the entire problem.

Attributes

None.

Children

`<hint>`

<hint>

Required. Specifies additional information that learners can access if needed.

Attributes

None.

Children

None.

Advanced Options for Checkbox Problems

Using the Script Element

You can use the <script> element to programmatically set attributes and options for your checkbox problems. You could use this feature to display different questions/answers depending on variable factors, like time of day, or randomly generated numbers.

Use the Advanced Editor to Configure the Script Element

You must use the advanced editor to configure a <script> element.

The contents of the <script> element must be enclosed in <![CDATA[...]]> markers, to indicate that the enclosed code should not be interpreted as XML.

The code in the <script> element is run on the server before the problem is shown to learners. Note that only Python script types are supported.

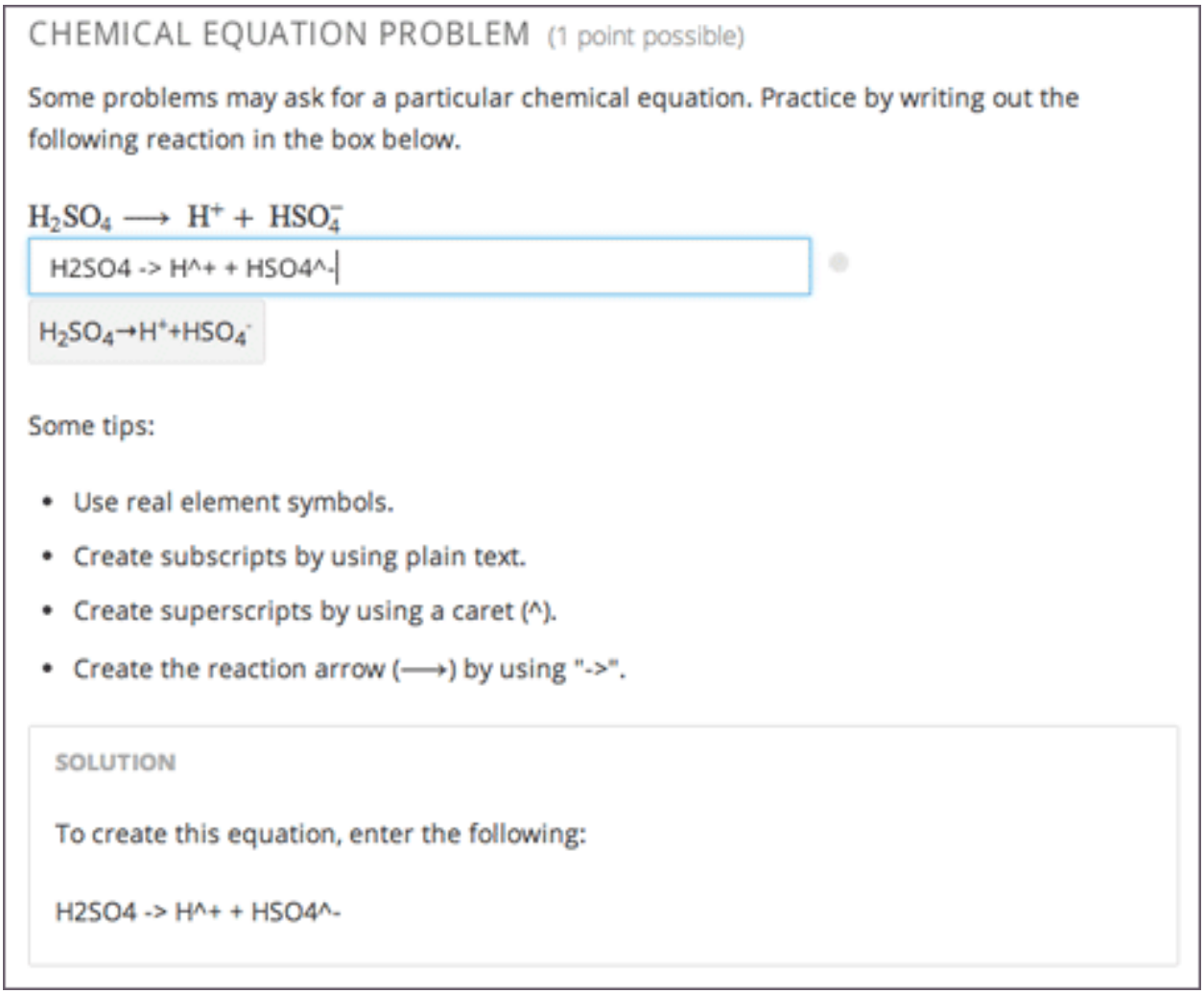
The following OLX example uses random numbers to generate different answer choices for each learner, and mathematical operators to determine each choice's correctness.

```
<problem>
    <script type="text/python">
    <![CDATA[
    random.seed(anonymous_student_id)  # Use different random numbers for each␣
↪student.
    a = random.randint(1,10)
    b = random.randint(1,10)
    c = a + b

    ok0 = c % 2 == 0 # check remainder modulo 2
    text0 = "$a + $b is divisible by 2"

    ok1 = c % 3 == 0 # check remainder modulo 3
    text1 = "$a + $b is divisible by 3"

    ok2 = c % 5 == 0 # check remainder modulo 5
    text2 = "$a + $b is divisible by 5"

    ok3 = not any([ok0, ok1, ok2])
    text3 = "None of the above statements is true."
    ]]>
    </script>
    <choiceresponse>
      <label>Which statements about the number $a+$b are true? Select all that apply.
↪</label>
      <checkboxgroup direction="vertical">
        <choice correct="$ok0">$text0 ... (should be $ok0)</choice>
        <choice correct="$ok1">$text1 ... (should be $ok1)</choice>
        <choice correct="$ok2">$text2 ... (should be $ok2)</choice>
        <choice correct="$ok3">$text3 ... (should be $ok3)</choice>
      </checkboxgroup>
    </choiceresponse>
</problem>
```

Chemical Equation Problem

Note: EdX does not support this problem type.

The chemical equation problem type allows the learner to enter text that represents a chemical equation into a text box. The system converts that text into a chemical equation below the text box. The grader evaluates the learner's response by using a Python script that you create and embed in the problem.



- *Create a Chemical Equation Problem*
- *Chemical Equation Problem XML*

Note: You can make a calculator available to your learners on every unit page. For more information, see Calculator Tool.

Create a Chemical Equation Problem

Chemical equation problems use MathJax to create formulas. For more information about using MathJax in Studio, see *Using MathJax for Mathematics*.

To create the above chemical equation problem, follow these steps.

1. In the unit where you want to create the problem, select **Problem** under **Add New Component**, and then select the **Advanced** tab.
2. Select **Blank Advanced Problem**.
3. In the component that appears, select **Edit**.
4. In the component editor, paste the code from below.
5. Select **Save**.

Sample Chemical Equation Problem Code

```
<problem>
  <startouttext/>
  <p>Some problems may ask for a particular chemical equation. Practice by writing␣
↪out the following reaction in the box below.</p>

\( \text{H}_2\text{SO}_4 \longrightarrow \text { H}^+ + \text{ HSO}_4^-\)

  <customresponse>
    <chemicalequationinput size="50" label="Enter the chemical equation"/>
    <answer type="loncapa/python">

if chemcalc.chemical_equations_equal(submission[0], 'H2SO4 -> H^+ + HSO4^-'):
    correct = ['correct']
else:
    correct = ['incorrect']

    </answer>
  </customresponse>
  <p>Some tips:</p>
  <ul>
  <li>Use real element symbols.</li>
  <li>Create subscripts by using plain text.</li>
  <li>Create superscripts by using a caret (^).</li>
  <li>Create the reaction arrow (\(\longrightarrow\)) by using "->".</li>
  </ul>

  <endouttext/>

 <solution>
 <div class="detailed-solution">
 <p>Solution</p>
 <p>To create this equation, enter the following:</p>
   <p>H2SO4 -> H^+ + HSO4^-</p>
 </div>
 </solution>
</problem>
```

Chemical Equation Problem XML

Template

```
<problem>
  <startouttext/>
  <p>Problem text</p>

  <customresponse>
    <chemicalequationinput size="NUMBER" label="LABEL TEXT"/>
    <answer type="loncapa/python">

if chemcalc.chemical_equations_equal(submission[0], 'TEXT REPRESENTING CHEMICAL␣
↪EQUATION'):
    correct = ['correct']
else:
    correct = ['incorrect']

    </answer>
  </customresponse>

  <endouttext/>

 <solution>
 <div class="detailed-solution">
 <p>Solution or Explanation Header</p>
 <p>Solution or explanation text</p>
 </div>
 </solution>
</problem>
```

Tags

- `<customresponse>`: Indicates that this problem has a custom response.
- `<chemicalequationinput>`: Specifies that the answer to this problem is a chemical equation.
- `<answer type=loncapa/python>`: Contains the Python script that grades the problem.

Tag: `<customresponse>`

Indicates that this problem has a custom response. The `<customresponse>` tags must surround the `<chemicalequation>` tags.

Attributes

(none)

Children

- `<chemicalequationinput>`
- `<answer>`

Tag: `<chemicalequationinput>`

Indicates that the answer to this problem is a chemical equation and creates a response field where the learner enters an answer.

Attributes

Attribute	Description
size	Specifies the size of the response field, in characters.
label (required)	Contains the text of the principal question in the problem.

Children

(none)

Tag: `<answer>`

Contains the Python script that grades the problem.

Attributes

Attribute	Description
type (required)	Must be “loncapa/python”.

Children

(none)

Circuit Schematic Builder Problem

Note: EdX does not support this problem type.

In circuit schematic builder problems, students can arrange circuit elements such as voltage sources, capacitors, resistors, and MOSFETs on an interactive grid. They then submit a DC, AC, or transient analysis of their circuit to the system for grading.

Lab 2

(1 point possible)

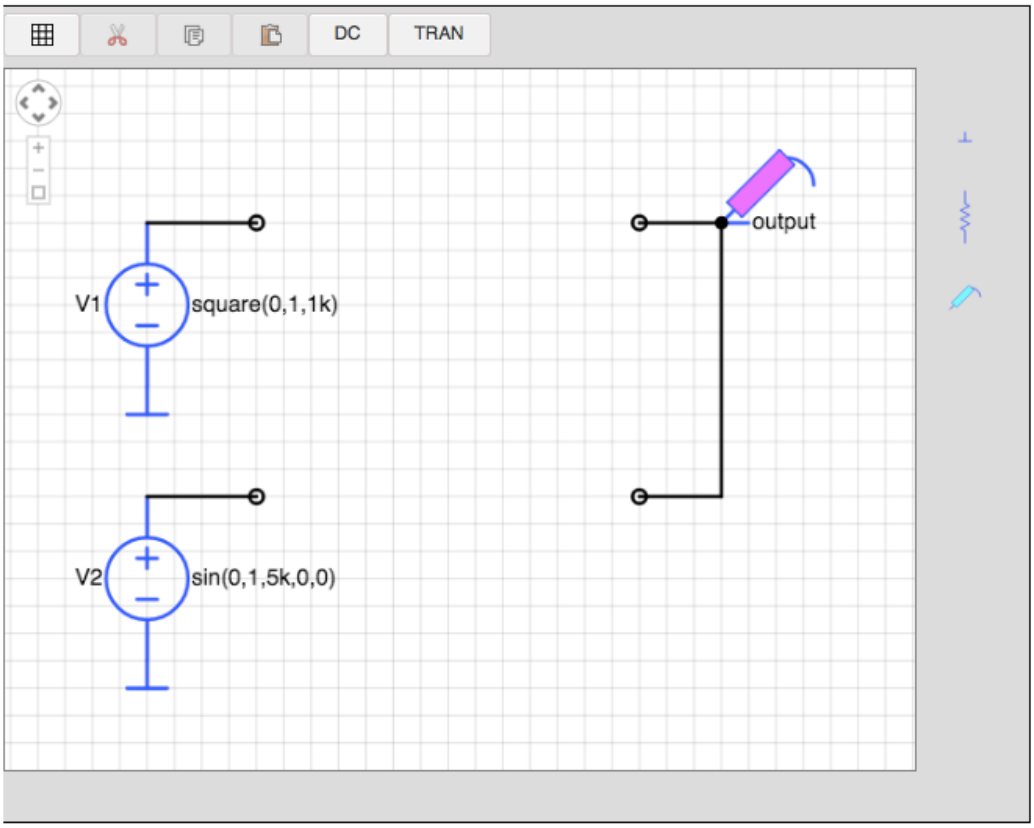
A circuit that combines two or more signals is called a *mixer*. In this lab, your goal is to build a mixer that combines the signals generated by two voltage sources, V1 and V2, where:

- V1 is a 1 kHz square wave that varies between 0V and +1V, and
- V2 is a 5 kHz sine wave that varies between -1V and +1V.

Please design a circuit that mixes V1 and V2 to produce Vout such that

$$V_{\text{out}} \approx \frac{1}{2}V_1 + \frac{1}{6}V_2.$$

Enter your circuit below, using the appropriate configuration of resistors. Please do not modify the wiring or parameters of the voltage sources -- your goal is to take the signals they generate and combine them, not to change what is generated. Run a 5ms transient analysis to verify the correct operation of your circuit. We will be checking for the transient waveform at the "output" node.



Create a Circuit Schematic Builder Problem

1. In the unit where you want to create the problem, under **Add New Component** select **Problem**, and then select **Advanced**.
2. Select **Circuit Schematic Builder**.
3. In the component that appears, select **Edit**.
4. In the component editor, replace the example code with your own code.
5. Select **Save**.

Problem Code

The illustration above shows a condensed version of an actual problem from MITx's 6.002.1x. To create the entire problem, paste the following code into the advanced editor.

```
<problem>
 <p>A circuit that combines two or more signals is called a <i>mixer</i>.  In
 this lab, your goal is to build a mixer that combines the signals generated
 by two voltage sources, V1 and V2, where:</p>
  <ul style="margin-left:2em;">
    <li>
      <p>V1 is a 1 kHz square wave that varies between 0V and +1V, and</p>
    </li>
    <li>
      <p>V2 is a 5 kHz sine wave that varies between -1V and +1V.</p>
    </li>
  </ul>
 <p>Please design a circuit that mixes V1 and V2 to produce Vout such that</p>
 <center>\[V_\mathrm{out} \approx \frac{1}{2}V_1 + \frac{1}{6}V_2.\]</center>
 <p>The resulting output should be similar to that shown in Figure 1.  The
 maximum value of the output is approximately \(667mV\) and the minimum value
 is approximately \(-167mV\).</p>
 <center><img src="/static/images/circuits/Lab2B_1.png"/><br/>Figure 1.  Desired␣
↪output waveform</center>
 <p>Hint: Figure 2 shows a simple resistive mixer for combining two signals.</p>
 <center><img src="/static/images/circuits/Lab2B_2.png"/><br/>Figure 2.  Simple␣
↪resistive mixer</center>
  <p>Enter your circuit below, using the appropriate configuration of
  resistors. Please do not modify the wiring or parameters of the voltage
  sources -- your goal is to take the signals they generate and combine them,
  not to change what is generated. Run a 5ms transient analysis to verify the
  correct operation of your circuit. We will be checking for the transient
  waveform at the "output" node.</p>
  <schematicresponse>
    <center>
      <schematic height="500" width="650" parts="g,r,s" analyses="dc,tran" submit_
↪analyses="{&quot;tran&quot;:[[&quot;output&quot;,0.00025,0.00035,0.00065,0.00075]]}
↪" initial_value="[[&quot;v&quot;,[56,48,0],{&quot;name&quot;:&quot;V1&quot;,&quot;
↪value&quot;:&quot;square(0,1,1k)&quot;,&quot;_json_&quot;:0},[&quot;2&quot;,&quot;0&
↪quot;]],[&quot;g&quot;,[56,96,0],{&quot;_json_&quot;:1},[&quot;0&quot;]],[&quot;v&
↪quot;,[56,128,0],{&quot;name&quot;:&quot;V2&quot;,&quot;value&quot;:&quot;sin(0,1,
↪5k,0,0)&quot;,&quot;_json_&quot;:2},[&quot;1&quot;,&quot;0&quot;]],[&quot;g&quot;,
↪[56,176,0],{&quot;_json_&quot;:3},[&quot;0&quot;]],[&quot;w&quot;,[56,48,88,48]],[&
↪quot;w&quot;,[56,128,88,128]],[&quot;L&quot;,[224,48,3],{&quot;label&quot;:&quot;
↪output&quot;,&quot;_json_&quot;:6},[&quot;output&quot;]],[&quot;w&quot;,[224,48,200,
↪48]],[&quot;w&quot;,[224,48,224,128]],[&quot;w&quot;,[224,128,200,128]],[&quot;s&
↪quot;,[224,48,0],{&quot;color&quot;:&quot;magenta&quot;,&quot;_json_&quot;:10},[&
↪quot;output&quot;]],[&quot;view&quot;,0,0,2,&quot;5&quot;,&quot;10&quot;,&quot;
↪10MEG&quot;,null,&quot;100&quot;,&quot;5ms&quot;]]"/>
```

```
    </center>
    <answer type="loncapa/python">
# for a schematic response, submission[i] is the json representation
# of the diagram and analysis results for the i-th schematic tag

def get_tran(json,signal):
  for element in json:
    if element[0] == 'transient':
      return element[1].get(signal,[])
  return []

output = get_tran(submission[0],'output')

answer = [[0.00025, 0.666],
          [0.00035, 0.333],
          [0.00065, 0.166],
          [0.00075, -0.166]]

okay = True
if not output or output[0][1] == 'undefined': # No transient or output node floating
  okay = False
else:
  for (at,av) in answer:
    for (t,v) in output:
      if at==t and abs(av - v) &lt; 0.05*abs(av):
        # found a good match for this answer, on to the next one
        break
    else:
      print 'check',at,av
      # no submission matched answer, complain
      okay = False;
      break;

correct = ['correct' if okay else 'incorrect']

</answer>
  </schematicresponse>
  <p>When you're done or if you wish to save your work, please click CHECK.
  The checker will be verifying the voltage of the output node at several
  different times, so you'll earn a point only <i>after</i> you've performed
  the transient simulation so that the checker will have a waveform to check!</p>
  <solution>
    <div class="detailed-solution"><p>Explanation:</p>
    <p>The goal is to design a mixer circuit with characteristics of
    \(V_{out}=\frac{1}{2}\cdot V_1+\frac{1}{6}\cdot V_2\)
    You might have started to design your mixer with two resistors only as the␣
↪example suggests.
    But working through the math, soon you'll realize that the equations return no␣
↪non-zero value for the resistor components.
    Thus you have to change the design. The next simplest design will be to add a␣
↪resistor \(R_3\) that connects the node Vout to ground.
    See the schematic below:</p>
    <img src="/static/images/circuits/lab2fmt.png"/>
    <p>Since we are going to use only linear elements in this circuit
    (resistors are linear), superposition will hold
    and thus one can look at the effect of each source \(V_1\) and \(V_2\)
    one at the time:</p>
  [mathjax] V_{out1} = V_1 \cdot \frac{\left(R_2 \parallel R_3\right)}
```

```
  {\left(R_2 \parallel R_3+R_1\right)}\\ V_{out2} = V_2 \cdot
  \frac{\left(R_1 \parallel R_3\right)}{\left(R_1 \parallel R_3+R_2\right)}
  \\ V_{out} = V_{out 1} + V_{out 2} \\ V_{out} = V_1 \cdot
  \frac{\left(R_2 \parallel R_3\right)}{\left(R_2\parallel R_3+R_1\right)} +
  V_2 \cdot \frac{\left(R_1 \parallel R_3\right)}{\left(R_1 \parallel R_3 +
  R_2\right)} = \frac{1}{2} \cdot V_1+\frac{1}{6} \cdot V_2 [/mathjax]
  <p>Therefore:</p>
  [mathjax] \frac{ \left(R_2 \parallel R_3 \right) }
  { \left( R_2 \parallel R_3 + R_1 \right)} =
  \frac{1}{2} \\\frac{\left( R_1 \parallel R_3 \right) }
  { \left(R_1 \parallel R_3 + R_2 \right)} = \frac{1}{6}  [/mathjax]
  <p>So we have to solve for the resistors given these two equations. You
  might notice that we have 2 equations and 3 unknowns, and that there is
  therefore not a unique solution. That is okay, though. We only have to
  worry if there is no solution, not if there are too many solutions. We will
  simply find one of the many possible correct answers by arbitrarily
  choosing a value for one of the variables later.</p>
  <p>The first equation simplifies to \( R_1 = R_2\parallel R_3\) and the
  second simplifies to \(R_2 = 5 \cdot R_1\parallel R_3\)
  Expanding the notation gives: </p>
  [mathjax]\frac{1}{R_1}=\frac{1}{R_2}+\frac{1}{R_3}
  \tag{*} \\\frac{1}{R_1}+\frac{1}{R_3}=\frac{5}{R_2}  [/mathjax]
  <p>Subtracting these two equations will yield \(R_2 = 2 \cdot R_3\)
  And putting this back to the starred equation , will result in
  \(R_1 = \frac{2}{3} \cdot R_3\)
  So now we have \(R_2\) and \(R_1\) in terms of \(R_3\) with the following
  ratios:</p>
  [mathjax]R_2 = 2 \cdot R_3 \\ R_1 = \frac{2}{3} \cdot R_3 \\[/mathjax]
  <p>Since the design hadn't mentioned anything about the resistances, one can
  use a simple value of \(R_3= 3Ω\) and find the rest accordingly:</p>
  [mathjax]R_1= 2Ω \\
  R_2= 6Ω \\
  R_3= 3Ω \\
  [/mathjax]
  <p>With these resistor values, doing a transient analysis shows a result which␣
↪meets the required specs of \(V_{out}\).</p>
  </div>
 </solution>
</problem>
```

Completion Tool

Note: EdX does not support this tool.

The completion tool provides learners with a way to mark sections of the course as completed. It helps learners to track their progress through sections of the course, including for ungraded activities such as reading text, watching video, or participating in course discussions.

Overview

The completion tool provides learners with a way to indicate both to themselves and to the course team that they have completed an activity.

The completion tool is itself a graded component and is therefore always included on the learner **Progress** page. However, depending on whether it is used in a graded or ungraded section of the course, it appears either as a part of the learner's final grade or as a practice score, respectively.

If you use the completion tool in an ungraded section of the course, the score for completing the activity that it is associated with is listed as a practice score on the **Progress** page. Practice scores are not included in the learner's final grade for the course.

You can also use the completion tool in graded sections of the course. If you do so, a score for completing the activity is included in learners' final grades. For example, if you include a completion component in a particular unit of the course content, learners who use the component to mark the unit as complete receive 1/1 points, while learners who do not mark the unit as complete receive 0/1 points. These scores are included in learners' final grades on the **Progress** page.

Note: EdX recommends using the completion tool primarily to track progress for ungraded activities such as reading assigned texts, watching videos, or participating in course discussions.

The Completion Control

When you add a completion component to a unit, learners see a control that is labeled **Mark as complete**. In this example, the completion component follows an HTML component.

After a learner selects this control, the label changes to **Unmark**. Learners who revisit their work in a unit and want to change the completion status can select this control as many times as needed.

Enable the Completion Tool

Before you can add a completion component to your course, you must enable the completion tool in Studio.

To enable the completion tool in Studio, add the "done" key to the **Advanced Module List** on the **Advanced Settings** page, then select **Save Changes**. (Be sure to include the quotation marks around the key value.) For more information, see Enabling Additional Exercises and Tools.

Add a Completion Component

After you have enabled the completion tool in Studio, to add a completion component to a unit in a course, follow these steps.

1. In the course outline in Studio, locate the unit to which you want to add the completion component.
2. Under **Add New Component**, select **Advanced**.
3. Select **Completion**. The completion component is added to the unit.

Note: Select **Edit** in the completion component for information about the tool.

Add the Completion Tool to an OLX Course

To add the completion tool to a unit in an OLX (open learning XML) course, it is sufficient to add the <done> tag to the OLX.

EdX recommends that you also explicitly specify a url_name within the <done> tag, as shown in the following example. If you do not explicitly specify a url_name, a value is automatically assigned, which can be problematic if the same course is imported several times. For example, if the url_name value is automatically generated each time you import your course, and if you import your course more than once, the learner state for the associated problems is lost each time a new url_name value is assigned.

```
<done url_name="video_3_completion"/>
```

Conditional Module

Note: EdX offers provisional support for this problem type.

A conditional module controls the content that learners see after a response that they make meets a certain condition. For example, learners who answer "Yes" to a poll question see a different block of text from the learners who answered "No" to the same question.

Format description

The main tag of conditional module input is conditional.

```
<conditional> ... </conditional>
```

conditional can include any number of any Xmodule tags (html, video, poll, etc.) or show tags.

`conditional` Tag

The main container for a single instance of a conditional module. The following attributes can be specified for this tag.

```
sources - location id of required modules, separated by ';'
[message | ""] - message for case, where one or more are not passed. Here you can use␣
↪variable {link}, which generate link to required module.

[submitted] - map to `is_submitted` module method.
(pressing RESET button makes this function to return False.)

[correct] - map to `is_correct` module method
[attempted] - map to `is_attempted` module method
[poll_answer] - map to `poll_answer` module attribute
[voted] - map to `voted` module attribute
```

`show` Tag

Symlink to some set of Xmodules. The following attribute can be specified for this tag.

```
sources - location id of modules, separated by ';'
```

Examples

Example: conditional depends on poll

```
<conditional sources="i4x://MITx/0.000x/poll_question/first_real_poll_seq_with_reset"␣
↪poll_answer="man"
message="{link} must be answered for this to become visible.">
    <html>
        <h3>You see this because your vote value for "First question" was "man"</h3>
    </html>
</conditional>
```

Example: conditional depends on poll (use <show> tag)

```
<conditional sources="i4x://MITx/0.000x/poll_question/first_real_poll_seq_with_reset"␣
↪poll_answer="man"
message="{link} must be answered for this to become visible.">
    <html>
        <show sources="i4x://MITx/0.000x/problem/test_1; i4x://MITx/0.000x/Video/Avi_
↪resources; i4x://MITx/0.000x/problem/test_1"/>
    </html>
</conditional>
```

Examples of conditional depends on problem

```
<conditional sources="i4x://MITx/0.000x/problem/Conditional:lec27_Q1" attempted="True
↪">
    <html display_name="HTML for attempted problem">You see this because "lec27_Q1"␣
↪was attempted.</html>
```

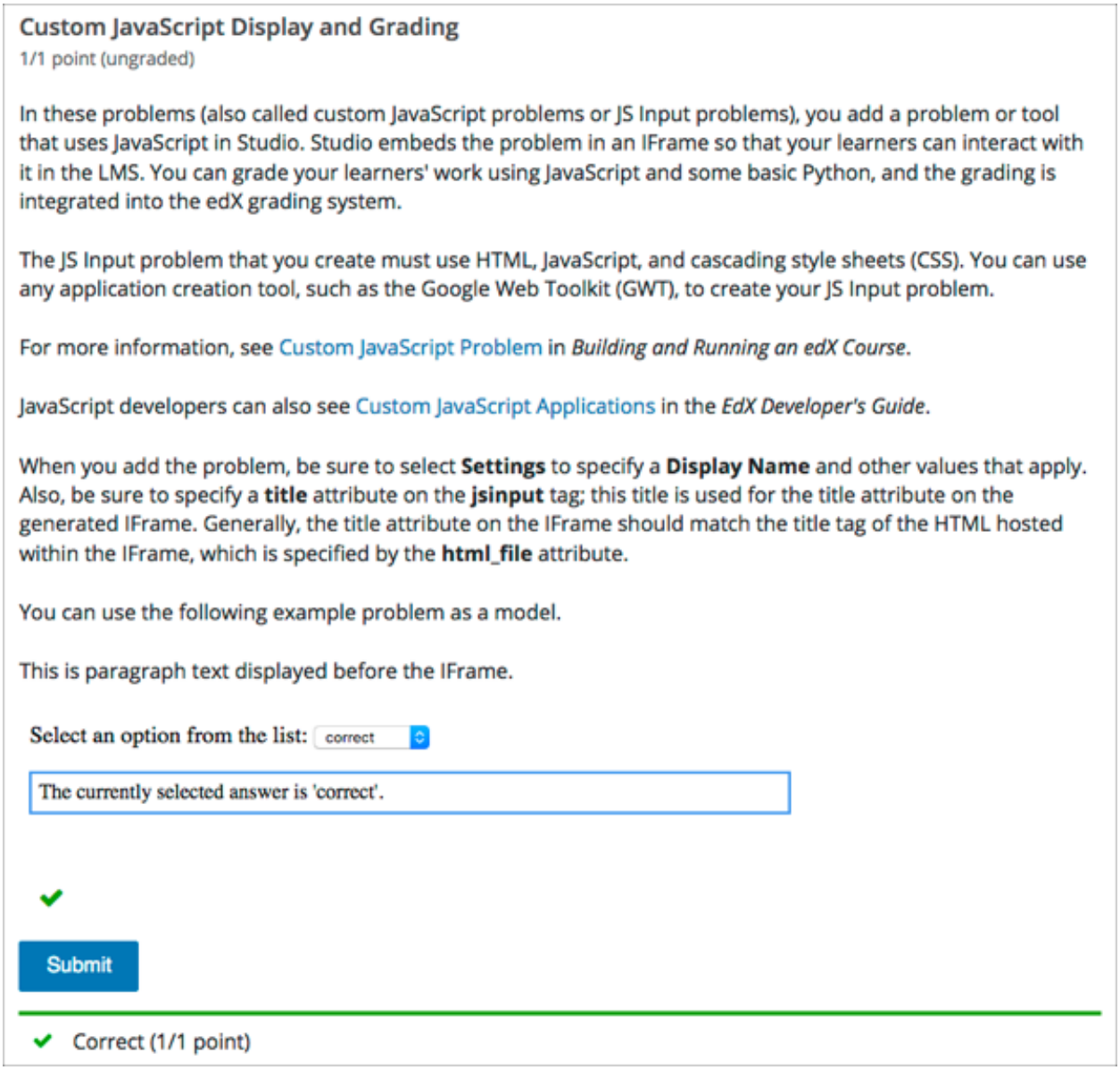
```
</conditional>
<conditional sources="i4x://MITx/0.000x/problem/Conditional:lec27_Q1" attempted="False
↪">
    <html display_name="HTML for not attempted problem">You see this because "lec27_Q1
↪" was not attempted.</html>
</conditional>
```

Custom JavaScript Display and Grading Problem

Note: EdX offers full support for this problem type.

Custom JavaScript display and grading problems (also called custom JavaScript problems or JS input problems) allow you to create a custom problem or tool that uses JavaScript and then add the problem or tool directly into Studio. When you create a JS input problem, Studio embeds the problem in an inline frame (an HTML iframe element) so that your learners can interact with it in the LMS. You can grade your learners' work using JavaScript and some basic Python, and the grading is integrated into the edX grading system.

The JS input problem that you create must use HTML, JavaScript, and cascading style sheets (CSS). You can use any application creation tool, such as the Google Web Toolkit (GWT), to create your JS input problem.



Caution:

- You cannot use a custom JavaScript problem in a component that contains more than one problem. Each custom JavaScript problem must be in its own component. See *Multiple Problems in One Component* for more information.
- The **Show Answer** button does not work for JS input problems. By default, the **Show Answer** option is set to **Never**. If you change this option in the problem component, a **Show Answer** button appears in the learner's view of the problem in the LMS, but the button does not work.

Create a Custom JavaScript Display and Grading Problem

1. Create your JavaScript application, and then upload all files associated with that application to the **Files & Uploads** page.

2. In the unit where you want to create the problem, under **Add New Component** select **Problem** , and then select **Advanced**.
3. Select **Custom JavaScript Display and Grading**.
4. In the component that appears, select **Edit**.
5. In the component editor, modify the example code according to your problem. Be sure to specify a `title` attribute on the `jsinput` tag. This title is used for the title attribute on the generated inline frame.
6. (Optional) To add a **Save** button to your problem, select **Settings**, and then set **Maximum Attempts** to a number larger than zero.
7. Select **Save**.

Note: All problems include more than one resource. If all the resources in a problem have the same protocol, host, and port, then the problem conforms to the same-origin policy (SOP). For example, the resources `http://store.company.com:81/subdirectory_1/JSInputElement.html` and `http://store.company.com:81/subdirectory_2/JSInputElement.js` have the same protocol (`http`), host (`store.company.com`), and port (`81`).

If any resources in your problem use a different protocol, host, or port, you need to bypass the SOP. For example, `https://info.company.com/JSInputElement2.html` uses a different protocol, host, and port from `http://store.company.com:81/subdirectory_1/JSInputElement.html`.

To bypass the SOP, change `sop="true"` to `sop="false"`. In the example problem code, this attribute is just before the closing `customresponse` tag.

If you bypass the same-origin policy, you require an additional file. The example problem uses the file `jschannel.js` to bypass the SOP.

For more information, see the same-origin policy page on the Mozilla Developer Network site or on Wikipedia.

JavaScript Input Example Problem Code

The following code recreates the JavaScript Input problem example shown in the overview. The example problem uses these files.

- https://files.edx.org/custom-js-example/jsinput_example.html
- https://files.edx.org/custom-js-example/jsinput_example.js
- https://files.edx.org/custom-js-example/jsinput_example.css
- https://files.edx.org/custom-js-example/jschannel.js (This file is used only because this example bypasses the SOP, as indicated by the line `sop="false"`)

```
<problem>
    <customresponse cfn="check_function">
        <script type="loncapa/python">
            <![CDATA[
            import json
            def check_function(e, ans):
                """
                "response" is a dictionary that contains two keys, "answer" and
                ""state".

                The value of "answer" is the JSON string that "getGrade" returns.
                The value of "state" is the JSON string that "getState" returns.
```

```
                Clicking either "Submit" or "Save" registers the current state.
                """
                response = json.loads(ans)

                # You can use the value of the answer key to grade:
                answer = json.loads(response["answer"])
                return answer == "correct"

                # Or you can use the value of the state key to grade:
                """
                state = json.loads(response["state"])
                return state["selectedChoice"] == "correct"
                """
            ]]>
        </script>
        <p>This is paragraph text displayed before the iframe.</p>
        <jsinput
            gradefn="JSInputDemo.getGrade"
            get_statefn="JSInputDemo.getState"
            set_statefn="JSInputDemo.setState"
            initial_state='{"selectedChoice": "incorrect1", "availableChoices":
            ["incorrect1", "correct", "incorrect2"]}'
            width="600"
            height="100"
            html_file="https://files.edx.org/custom-js-example/jsinput_example.html"
            title="Dropdown with Dynamic Text"
            sop="false"/>
    </customresponse>
</problem>
```

Note: Keep the following points in mind about this example problem.

- The jsinput_example.js file defines three JavaScript functions (**JSInputDemo.getGrade**, **JSInputDemo.getState**, and **JSInputDemo.setState**).
- The JavaScript input problem code uses **JSInputDemo.getGrade**, **JSInputDemo.getState**, and **JSInputDemo.setState** to grade, save, or restore a problem. These functions must be global in scope.
- **JSInputDemo.getState** and **JSInputDemo.setState** are optional. You need to define these functions only if you want to conserve the state of the problem.
- **Width** and **height** represent the dimensions of the inline frame that holds the application.
- The response is graded as correct if the `correct` option is selected in the dropdown control when the user selects **Submit**.
- Selecting **Submit** registers the problem's current state.

JavaScript Input Problem XML

JSInput allows problem authors to turn stand-alone HTML files into problems that can be integrated into the edX platform. Since its aim is flexibility, it can be seen as the input and client-side equivalent of **CustomResponse**.

A JSInput exercise creates an inline frame (iframe) in a static HTML page, and passes the return value of author-specified functions to the enclosing response type (generally **CustomResponse**). JSInput can also store and retrieve state.

Template

The following is the basic format of a JSInput problem.

```
<problem>
       <script type="loncapa/python">
           def all_true(exp, ans): return ans == "hi"
       </script>
       <customresponse cfn="all_true">
           <jsinput gradefn="gradefn"
               height="500"
               get_statefn="getstate"
               set_statefn="setstate"
               html_file="/static/jsinput.html"
               title="iframe Title"/>
       </customresponse>
</problem>
```

The accepted attributes are:

Attribute Name	Value Type	Required	Default
html_file	URL string	Yes	None
title	string	Yes	`Problem Remote Content`
gradefn	Function name	Yes	`gradefn`
set_statefn	Function name	No	None
get_statefn	Function name	No	None
height	Integer	No	`300`
width	Integer	No	`400`
title	String	No	None

Required Attributes

- **html_file**

  The **html_file** attribute specifies the HTML file that the iframe will point to. The HTML file must be located in the content directory.

  The iframe is created using the sandbox attribute. Although pop-ups, scripts, and pointer locks are allowed, the iframe cannot access its parent's attributes.

  The HTML file must contain a **gradefn** function that the JSInput file can access. To determine whether the **gradefn** function is accessible, in the console, make sure that **gradefn** returns the right thing. When JSInput uses the **gradefn** function, *gradefn* is called with *gradefn*.call(*obj*), where **obj** is the object-part of **gradefn**. For example, if **gradefn** is **myprog.myfn**, JSInput calls **myprog.myfun.call(myprog)**.

  The HTML file has no specific requirements other than the **gradefn** function. Note that inheriting CSS or JavaScript from the parent (except for the Chrome-only **seamless** attribute, which is set to `True` by default) is not currently supported.

- **title**

  The **title** attribute specifies the title for the generated iframe. Generally, the title attribute on the iframe should match the title tag of the HTML file that is hosted within the iframe.

- **gradefn**

  The **gradefn** attribute specifies the name of the function that will be called when a user selects **Submit**, and that returns the learner's answer. Unless both the **get_statefn** and **set_statefn** attributes are also used, this answer is

passed as a string to the enclosing response type. In the **customresponse** example above, this means **cfn** will be passed this answer as `ans`.

If the **gradefn** function throws an exception when a learner attempts to submit a problem, the submission is aborted, and the learner receives a generic alert. The alert can be customized by making the exception name `Waitfor Exception`; in that case, the alert message will be the exception message.

Important: To make sure the learner's latest answer is passed correctly, make sure that the **gradefn** function is not asynchronous. Additionally, make sure that the function returns promptly. Currently the learner has no indication that her answer is being calculated or produced.

Optional Attributes

- **set_statefn**

  Sometimes a problem author will want information about a learner's previous answers ("state") to be saved and reloaded. If the attribute **set_statefn** is used, the function given as its value will be passed the state as a string argument whenever there is a state, and the learner returns to a problem. The function has the responsibility to then use this state appropriately.

  The state that is passed is:

  - The previous output of **gradefn** (i.e., the previous answer) if **get_statefn** is not defined.
  - The previous output of **get_statefn** (see below) otherwise.

  It is the responsibility of the iframe to do proper verification of the argument that it receives via **set_statefn**.

- **get_statefn**

  Sometimes the state and the answer are quite different. For instance, a problem that involves using a JavaScript program that allows the learner to alter a molecule may grade based on the molecule's hydrophobicity, but from the hydrophobicity it might be incapable of restoring the state. In that case, a *separate* state may be stored and loaded by **set_statefn**. Note that if **get_statefn** is defined, the answer (i.e., what is passed to the enclosing response type) will be a json string with the following format:

```
{
    answer: `[answer string]`
    state: `[state string]`
}
```

  The enclosing response type must then parse this as json.

- **height** and **width**

  The **height** and **width** attributes are straightforward: they specify the height and width of the iframe. Both are limited by the enclosing DOM elements, so for instance there is an implicit max-width of around 900.

  In the future, JSInput may attempt to make these dimensions match the HTML file's dimensions (up to the aforementioned limits), but currently it defaults to `300` and `400` for **height** and **width**, respectively.

Write-Your-Own-Grader Problem

Note: EdX offers provisional support for this problem type.

This section provides information about writing your own grader directly in a problem component.

Overview

In custom Python-evaluated input (also called "write-your-own-grader" problems), the grader uses a Python script that you create and embed in the problem to evaluate a learner's response or provide hints. These problems can be any type. *Numerical input* and *text input* problems are the most common write-your-own-grader problems.

Custom Python-evaluated input problems can include the following advanced problem types.

- *Chemical Equation Problem*
- *Custom JavaScript Display and Grading Problem*
- *Gene Explorer Tool*
- *Molecule Editor Tool*
- *Protex Protein Builder Tool*

Create a Custom Python-Evaluated Input Problem in Studio

1. In the unit where you want to create the problem, select **Problem** under **Add New Component**, and then select the **Advanced** tab.
2. Select **Custom Python-Evaluated Input**.
3. In the component that appears, select **Edit**.
4. In the component editor, edit the problem in *Script Tag Format*.
5. Select **Save**.

Script Tag Format

The script tag format encloses a Python script that contains a "check function" in a `<script>` tag, and adds the `cfn` attribute of the `<customresponse>` tag to reference that function.

This section contains the following information about using the `<script>` tag.

- *Create a Randomized Custom Python-Evaluated Input Problem*

The `check` Function

The `check` function in a `<script>` tag accepts two arguments.

- `expect` is the value of the `expect` attribute of `<customresponse>`. If `expect` is not provided as an argument, the function must have another way to determine if the answer is correct.
- `answer` is one of the following two options.
  - The value of the answer the learner provided, if the problem only has one response field.
  - An ordered list of answers the learner provided, if the problem has multiple response fields.

The `check` function can return any of the following values to indicate whether the learner's answer is correct.

- `True`: Indicates that the learner answered correctly for all response fields.
- `False`: Indicates that the learner answered incorrectly. All response fields are marked as incorrect.
- `"Partial"`: Indicates that the learner's answer was partially correct. By default, the learner receives 50% of the points for the problem. For more information, see *Award Half Credit*.
- A dictionary of the form `{ 'ok': True, 'msg': 'Message' }`. If the dictionary's value for `ok` is set to `True`, all response fields are marked correct. If it is set to `False`, all response fields are marked incorrect. If it is set to `"Partial"`, the learner receives 50% of the problem points. The `msg` is displayed below all response fields, and it can contain XHTML markup.
- A dictionary of the form

```
{ 'overall_message': 'Overall message',
    'input_list': [
        { 'ok': True, 'msg': 'Feedback for input 1'},
        { 'ok': False, 'msg': 'Feedback for input 2'},
        { 'ok': 'Partial', 'msg': 'Feedback for input 3'}
        ... ] }
```

The last form is useful for responses that contain multiple response fields. It allows you to provide feedback for each response field individually, as well as a message that applies to the entire response.

Example with the Script Tag

In the following example, `<customresponse>` tags reference the `test_add_to_ten` and `test_add` functions that are in the `<script>` tag.

Important: Python honors indentation. Within the `<script>` tag, the `def check_func(expect, ans):` line must have no indentation.

```
<problem>

<script type="loncapa/python">

def test_add(expect, ans):
    try:
        a1=int(ans[0])
```

```
        a2=int(ans[1])
        return (a1+a2) == int(expect)
    except ValueError:
        return False

def test_add_to_ten(expect, ans):
    return test_add(10, ans)

</script>

<p>Enter two integers that sum to 10. </p>
<customresponse cfn="test_add_to_ten">
  <textline size="10"/><br/>
  <textline size="10"/>
</customresponse>

<p>Enter two integers that sum to 20: </p>
<customresponse cfn="test_add" expect="20">
  <textline size="40" correct_answer="11" label="Integer #1"/><br/>
  <textline size="40" correct_answer="9" label="Integer #2"/>
</customresponse>

<solution>
  <div class="detailed-solution">
    <p>Explanation</p>
    <p>Any set of integers on the line \(y = 10 - x\) and \(y = 20 - x\)
       satisfies these constraints.</p>
    <p>You can also add images within the solution clause, like so:</p>
    <img src="/static/images/placeholder-image.png"/>
  </div>
</solution>

</problem>
```

Example of the `check` Function Returning a Dictionary

The following example shows a `check` function that returns a dictionary.

```
def check(expect, answer_given):
    check1 = (int(answer_given[0]) == 1)
    check2 = (int(answer_given[1]) == 2)
    check3 = (int(answer_given[2]) == 3)
    return {'overall_message': 'Overall message',
                'input_list': [
                    { 'ok': check1, 'msg': 'Feedback 1'},
                    { 'ok': check2, 'msg': 'Feedback 2'},
                    { 'ok': check3, 'msg': 'Feedback 3'} ] }
```

The function checks that the user entered `1` for the first input, `2` for the second input, and `3` for the third input. It provides feedback messages for each individual input, as well as a message displayed below the entire problem.

Script Tag Attributes

The following table explains the important attributes and values in the preceding example.

<script type="loncapa/python">	Indicates that the problem contains a Python script.
<customresponse cfn="test_add_to_ten">	Indicates that the function test_add_to_ten is called when the learner checks the answers for this problem.
<customresponse cfn="test_add" expect="20">	Indicates that the function test_add is called when the learner checks the answers for this problem and that the expected answer is 20.
<textline size="10" correct_answer="3"/>	This tag includes the size, correct_answer, and label attributes. The correct_answer attribute is optional.

Create a Custom Python-Evaluated Input Problem in Script Tag Format

To create a custom Python-evaluated input problem using a <script> tag, follow these steps.

1. In the component editor, modify the example as needed.
2. Select **Save**.

Problem Code:

```
<problem>
<p>This question has two parts.</p>

<script type="loncapa/python">

def test_add(expect, ans):
    try:
        a1=int(ans[0])
        a2=int(ans[1])
        return (a1+a2) == int(expect)
    except ValueError:
        return False

def test_add_to_ten(expect, ans):
    return test_add(10, ans)

</script>

<p>Part 1: Enter two integers that sum to 10. </p>
<customresponse cfn="test_add_to_ten">
        <textline size="10" correct_answer="3" label="Integer #1"/><br/>
        <textline size="10" correct_answer="7" label="Integer #2"/>
</customresponse>

<p>Part 2: Enter two integers that sum to 20. </p>
<customresponse cfn="test_add" expect="20">
        <textline size="10" label="Integer #1"/><br/>
        <textline size="10" label="Integer #2"/>
</customresponse>

<solution>
    <div class="detailed-solution">
        <p>Explanation</p>
        <p>For part 1, any two numbers of the form <i>n</i> and <i>10-n</i>,
        where <i>n</i> is any integer, will work. One possible answer would
        be the pair 0 and 10.</p>
        <p>For part 2, any two numbers of the form <i>n</i> and <i>20-n</i>, where <i>
↪n</i> is any integer, will work. One possible answer would be the pair 1 and 19</p>
```

```
    </div>
</solution>
</problem>
```

Templates

The following template includes answers that appear when the learner selects **Show Answer**.

```
<problem>

<script type="loncapa/python">
def test_add(expect,ans):
  a1=float(ans[0])
  a2=float(ans[1])
  return (a1+a2)== float(expect)
</script>

<p>Problem text</p>
<customresponse cfn="test_add" expect="20">
        <textline size="10" correct_answer="11" label="Integer #1"/><br/>
        <textline size="10" correct_answer="9" label="Integer #2"/>
</customresponse>

    <solution>
        <div class="detailed-solution">
          <p>Solution or Explanation Heading</p>
          <p>Solution or explanation text</p>
        </div>
    </solution>
</problem>
```

The following template does not return answers when the learner selects **Show Answer**. If your problem does not include answers for the learner to see, make sure to set **Show Answer** to **Never** in the problem component.

```
<problem>

<script type="loncapa/python">
def test_add(expect,ans):
  a1=float(ans[0])
  a2=float(ans[1])
  return (a1+a2)== float(expect)
</script>

<p>Enter two real numbers that sum to 20: </p>
<customresponse cfn="test_add" expect="20">
        <textline size="10"  label="Integer #1"/><br/>
        <textline size="10"  label="Integer #2"/>
</customresponse>

    <solution>
        <div class="detailed-solution">
          <p>Solution or Explanation Heading</p>
          <p>Solution or explanation text</p>
        </div>
    </solution>
</problem>
```

Award Partial Credit

You can configure a custom Python-evaluated input problem so that learners who give a partially correct answer receive partial credit for the problem. You can award 50% of the points for the problem, or you can award a different percentage of points. For more information, see the following sections.

- *Award Half Credit*
- *Award a Percentage of Credit*

Award Half Credit

You can configure a problem to award 50% of the possible points. To provide a learner with a more granular score, see *Award a Percentage of Credit*.

The `check` function must return the value `"Partial"` in one of the following ways.

- Return the value `"Partial"` directly.
- Return the value `"Partial"` in the dictionary that is returned, in the following form.

  ```
  { 'ok': 'Partial', 'msg': 'Message' }
  ```

- Return the value `"Partial"` as part of the input list for multi-part problems.

```
{ 'overall_message': 'Overall message',
    'input_list': [
        { 'ok': True, 'msg': 'Feedback for input 1'},
        { 'ok': False, 'msg': 'Feedback for input 2'},
        { 'ok': 'Partial', 'msg': 'Feedback for input 3'}
        ... ] }
```

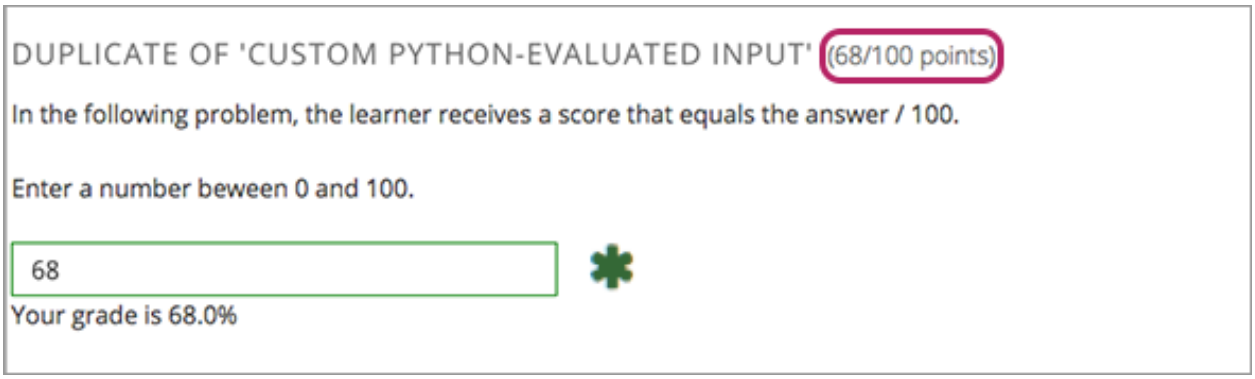
With all of these options, `True` awards learners with 100% of the available points for the problem, `'Partial'` with 50%, and `False` with 0%.

For more information about `check` function return values, see *The check Function*.

Award a Percentage of Credit

You can configure a problem to return a percent value as a grade. This method provides greater flexibility in assigning the learner a score than *awarding half credit*.

In the following example, the learner's score equals the answer divided by 100.



The following code shows the configuration of this problem.

```
<problem>
<p>In the following problem, the learner receives a score that equals the
   answer / 100. If the learner's answer is greater than 100 or less than 0,
   the score equals 0.</p>

<script type="loncapa/python">

def give_partial_credit(expect, ans):
  ans = float(ans)
  if ans &gt; 100 or ans &lt; 0:
      # Assign a score of zero if the answer is less than zero or over 100.
      ans = 0
  grade = ans/100
  return {
      'input_list': [
         { 'ok': True, 'msg': 'Your grade is ' + str(ans) + '%', 'grade_decimal
↪':grade},
      ]
  }
</script>

<p>Enter a number beween 0 and 100.</p>
<customresponse cfn="give_partial_credit">
  <textline points="100" size="40" label="Ans1"/><br/>
</customresponse>
</problem>
```

This example illustrates the following points.

- The `points` attribute of the `<customresponse>` element specifies that the question is worth 100 points.
- The `give_partial_credit` function checks that the answer is between 0 and 100, and if so divides the learner's answer by 100 to determine the grade.
- The `input_list` that is returned specifies that:
  - The answer is acceptable and can receive partial or full credit, with the item `'ok': True`.
  - The learner receives the message `Your grade is` followed by the percent grade, with the item `'msg': 'Your grade is ' + str(ans) + '%'`.
  - The grade assigned is the learner's answer divided by 100, with the item `'grade_decimal':grade`.

You can enhance and apply this example for your own partial credit problems.

Create a Randomized Custom Python-Evaluated Input Problem

You can create a custom Python-evaluated input problem that randomizes variables in the Python code.

Note: In the problem settings, you must set the **Randomization** value to something other than **Never** to have Python variables randomized. See *Randomization* for more information.

The following example demonstrates using randomization with a Python-evaluated input problem.

Note: This example uses the method `random.randint` to generate random numbers. You can use any standard Python library for this purpose.

```
<problem>
  <p>Some problems in the course will utilize randomized parameters.
     For such problems, after you check your answer you will have the option
     of resetting the question, which reconstructs the problem with a new
     set of parameters.</p>
<script type="loncapa/python">
x1 = random.randint(0, 100)
x2 = random.randint(0, 100)
y = x1+x2
</script>
<p>Let (x_1 = $x1) and (x_2 = $x2). What is the value of (x_1+x_2)?</p>
<numericalresponse answer="$y">
  <responseparam type="tolerance" default="0.01%" name="tol"
    description="Numerical Tolerance"/>
  <textline size="10"/>
</numericalresponse>
<solution>
  <p><b>Explanation:</b></p>
</solution>
</problem>
```

Drag and Drop Problem

Note: EdX offers full support for this problem type.

In drag and drop problems, learners respond to a question by dragging text or images to a specific location on a background image. This section explains how to use drag and drop problems in your course.

Note: This drag and drop problem type is intended as a replacement for an older drag and drop problem type. This drag and drop problem type includes significant improvements and you should use it for any new course development. For more information about the previous, deprecated drag and drop problem type, see Drag and Drop Problem (Deprecated).

Overview of Drag and Drop Problems

A drag and drop problem includes a background image and one or more items that learners can drag into target zones on the background image. You can include as many draggable items and as many target zones as you need. You can include decoy items that do not have a target, and you can include decoy targets that do not correspond to draggable items.

When learners view a drag and drop problem in the LMS, the problem includes a top section and a bottom section. Learners drag items from the top section on to the background image in the section below it.

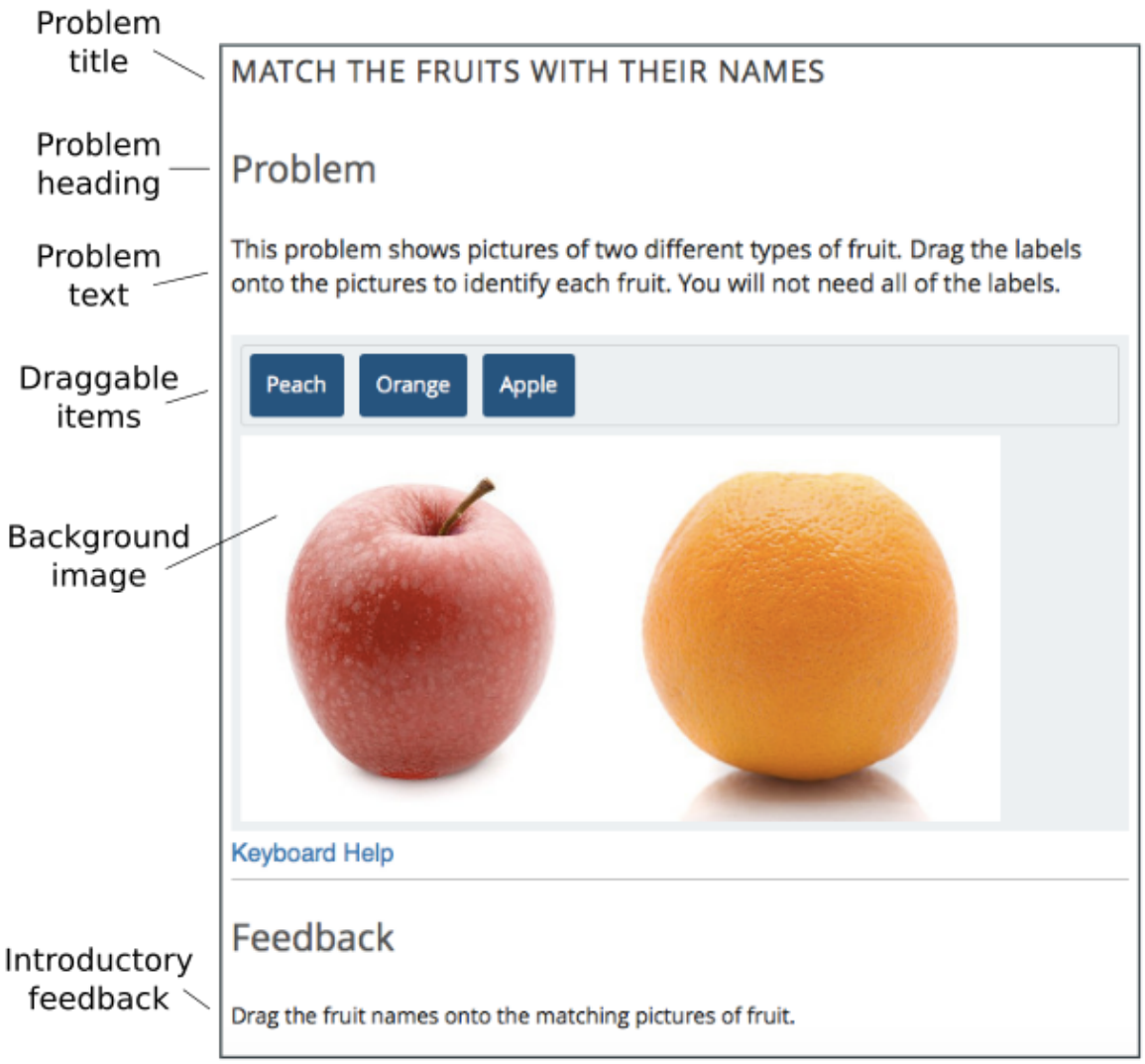
The way that a learner selects, or grabs, an item depends on the type of browser that the learner uses. For example, a learner might click and hold on a draggable item with a mouse pointer to select it, drag the item to a target, and release the mouse pointer to drop the item on the target. A learner who accesses the problem on a mobile device with a touch screen might swipe an item from the top section into a target zone. A learner who uses a keyboard interface might use the navigation keys to select an item and then match it to a target zone.

You can configure a drag and drop problem to give learners unlimited attempts to match items to target zones or you can configure it to behave restrictively, like a test.

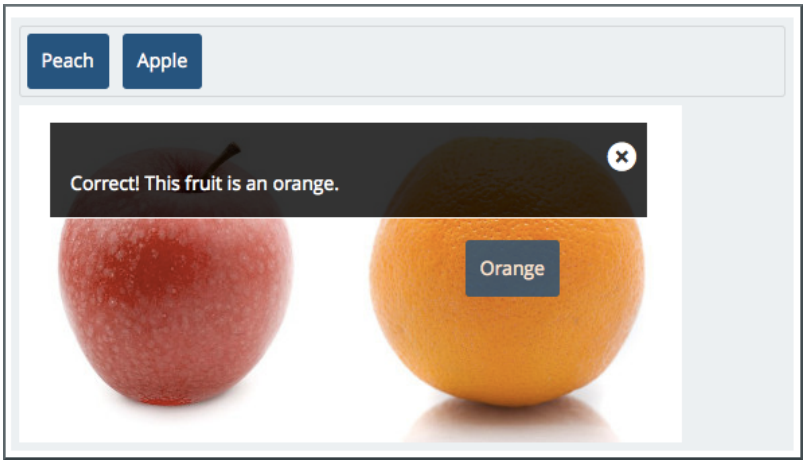
- In standard mode, the problem gives learners unlimited attempts to match items and it provides immediate feedback to indicate whether an item is matched correctly.
- In assessment mode, learners must match all of the draggable items to target zones and then submit the problem. The problem does not reveal whether items are matched correctly until the learner submits the problem.

For more information about assessment mode and standard mode, see *Choosing a Drag and Drop Problem Mode*.

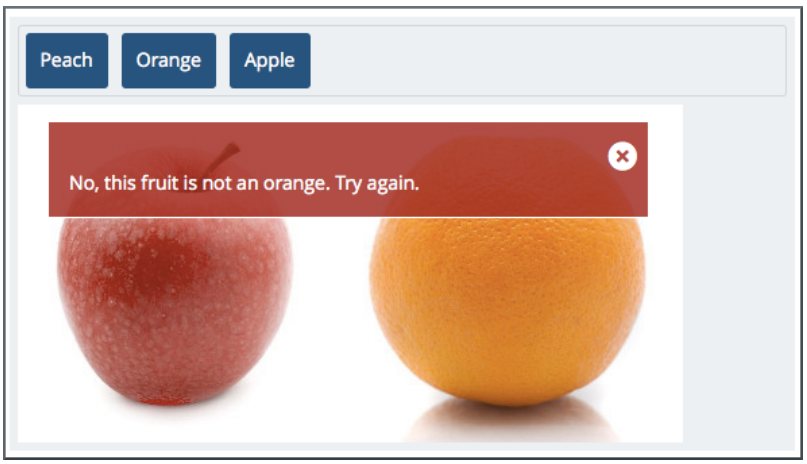
The following image shows an example drag and drop problem.



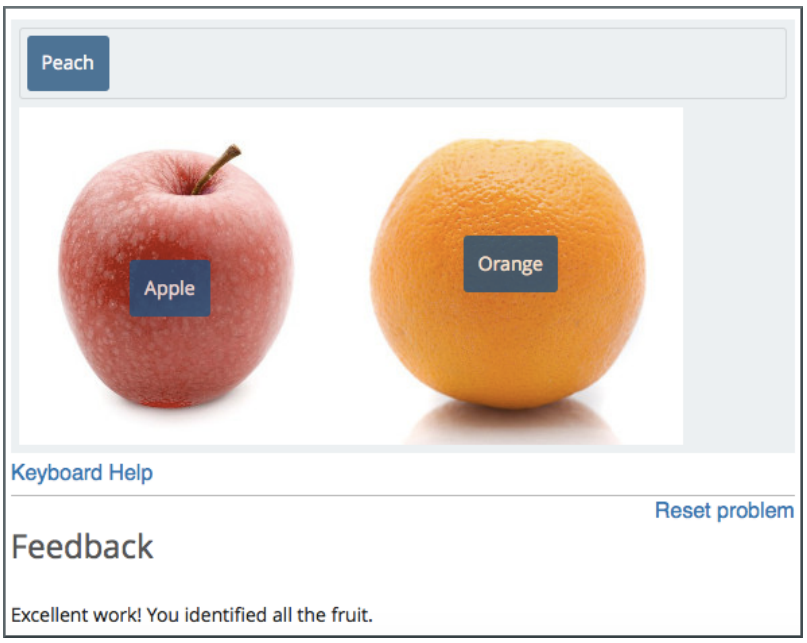
The following image shows the success feedback message that learners see when they match a draggable item with its target zone. Each draggable item has its own success feedback message.



The following image shows the error feedback message that learners see when they match a draggable item with an incorrect target zone. Each draggable item has its own error feedback message.



The following image shows a completed drag and drop problem. The final feedback message informs the learner that the problem is complete.



Understanding Background Images

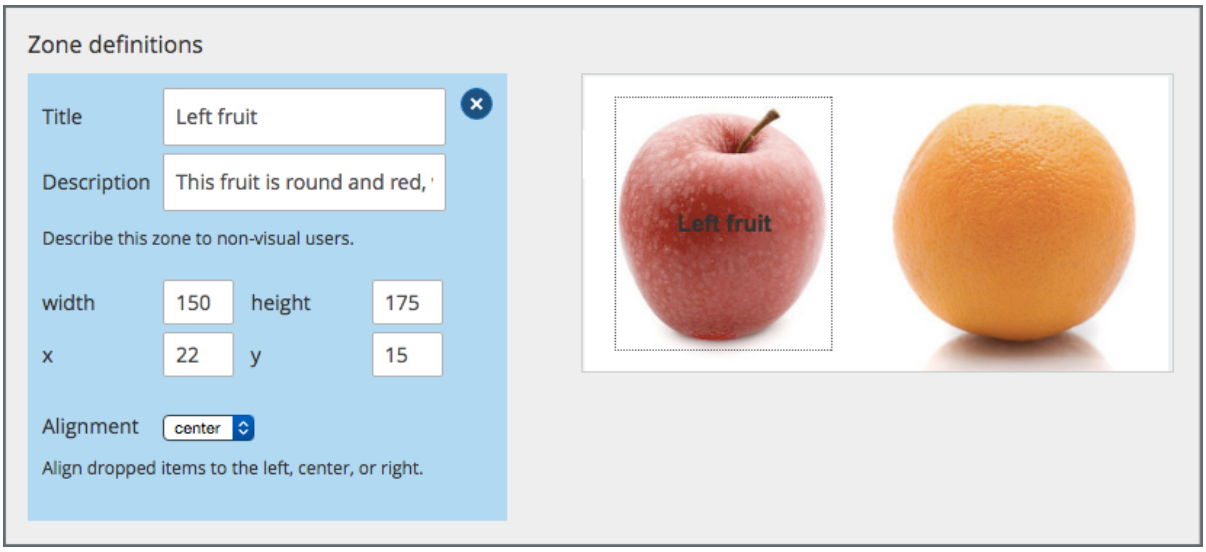
The background image for a drag and drop problem is the surface that learners drag items onto.

A target zone is a rectangular area on the background image. You can show or hide the borders of a zone for learners. You can add titles for zones or leave the **Title** field empty. However, you must fill in the **Description** field for each zone. The description is only exposed to screen readers. The description must describe the content of the zone for visually impaired learners. For example, a zone that includes an apple might have a description that says "A round, red fruit with a stem."

A background image must fit within the course screen. The LMS automatically scales images that are too wide. If you choose a background image that is extremely large, you should consider how it appears to learners after scaling. The width of the course screen depends on the device and browsing software that a learner uses.

You define a target zone by specifying its width, height, horizontal offset (`x`), and vertical offset (`y`). Each specification is in pixels. The horizontal offset is the distance between the left side of the background image and the left side of the target zone. The vertical offset is the distance between the top of the background image and the top of the target zone.

The following image shows a background image and target zones in the drag and drop problem editing dialog box. For more information about editing drag and drop problems, see *Creating a Drag and Drop Problem*.



Note: The pixel coordinates that you use to specify the size and location of target zones are also used by common image editing software. You can open a background image in an image editing program to find the pixel coordinates of a target zone.

Understanding Draggable Items

A draggable item is a rectangle that contains either a label or an image. Learners grab draggable items from the top of a drag and drop problem and drag them to targets on the background image.

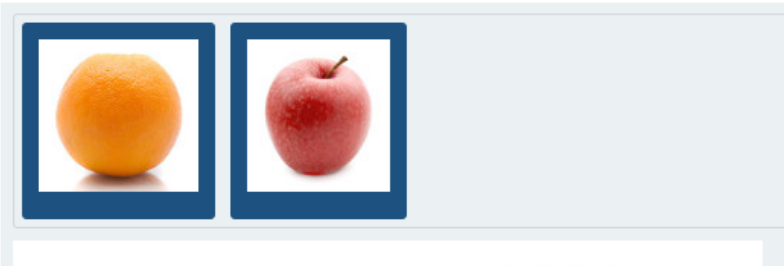
You can set the size of the rectangle for a draggable item as a percentage of the width of the problem. If you do not specify the size, the LMS sets it based on the length of the text in the label or the size of the image in it. You can set the background color and the label text color for the items in a problem.

Each draggable item can match one target zone on the background image, multiple target zones, or no target zones.

Each item must have a text label to identify it in the drag and drop problem. If you include only a text label, that label appears in the draggable item. If you include both a text label and an image for an item, the image appears as its label.

Using Draggable Items with Images

The following image shows draggable items with images. For examples of items with text, see *Overview of Drag and Drop Problems*.



Images for draggable items have alternative image descriptions. The alternative description explains the image content in text. If a learner cannot access the visual image content, the text description helps that learner to complete the

problem.

Images for draggable items must fit within the top section of the problem. The LMS automatically scales large images to fit. If you use a large image in a draggable item, you should consider how that image appears after scaling.

Note: If an image file is unavailable, or cannot be displayed, the LMS displays the text description as the button label.

Choosing a Drag and Drop Problem Mode

You can configure drag and drop problems to allow learners to experiment with matching draggable items to target zones until all items are matched correctly, or to require that learners match all items to target zones without any input and then submit their attempts for grading. You can choose either **Standard Mode** or **Assessment Mode** to control the behavior of the problem.

- In standard mode, learners have unlimited attempts to match items and the problem provides immediate feedback to indicate whether an item is matched correctly.
- In assessment mode, learners must match all of the draggable items to target zones and then choose to submit the problem. The problem does not reveal whether items are matched correctly until the learner submits the problem. You can limit the number of attempts a learner is allowed, or allow unlimited attempts.

Using Standard Mode

Standard mode configures a drag and drop problem to give learners unlimited attempts to match draggable items with target zones until all of the items are matched to the correct targets. Each time a learner drops an item on a target zone, the problem reports whether the match is correct. If the match is not correct, the draggable item is returned to the item bank for a new attempt.

A learner completes a drag and drop problem in standard mode when all of the items are matched to target zones correctly. Learners receive the maximum score for the problem when the problem is complete.

Using Assessment Mode

Assessment mode configures a drag and drop problem to behave like a test. In assessment mode, learners must match all of the draggable items to target zones before the problem reveals whether the items are matched correctly.

Learners select **Submit** when they believe that they have completed the problem. If all items are matched correctly, the problem is complete. If any items are not matched correctly, and the maximum number of attempts has not been reached, the learner can correct items and select **Submit** again. When the learner reaches the maximum number of attempts, the problem is complete.

The score for the problem is calculated by dividing the maximum score based on the percent of draggable items that are matched correctly. If a learner attempts the problem multiple times, the score for the best attempt is the final score for the problem.

In assessment mode, you can specify the number of times that learners can submit a drag and drop problem. If you allow more than one attempt, the problem reveals which items are correctly matched and gives learners an opportunity to move items that are not correct. If you do not specify a limit, learners have unlimited attempts.

Creating a Drag and Drop Problem

To create a drag and drop problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Advanced**.
2. From the list of advanced components, select **Drag and Drop**.

   Studio adds the drag and drop problem to the unit.
3. Select **Edit**. The **Editing** dialog box opens.

   Configure your drag and drop problem. For detailed information about individual controls in the **Editing** dialog box, see *Understanding the Drag and Drop Editing Controls*.

   The **Editing** dialog box includes multiple screens. Configure each screen and select **Continue**. On the final screen, select **Save** to exit the configuration dialog box and save your changes.

   In particular, configure the aspects of the drag and drop problem described below.

   - Edit the problem title, problem text, introductory feedback, and final feedback for the problem. For more information about how the text in these fields appears in a drag and drop problem, see *Overview of Drag and Drop Problems*.
   - Specify a background image in the **Background URL** field. Enter the URL of a file you have added to your course or the URL of an image on the web. For more information about working with course files, see Adding Files to a Course. For more information about background images, see *Understanding Background Images*.

     Select **Change background** after you enter the URL for your background image.

     If you specify the URL of an image on the web, make sure that you are legally authorized to use the image and that the image is available to learners around the world.
   - Remove the target zones for the example drag and drop problem. Select **Add a zone** to add each target zone for your problem. For more information about target zones, see *Understanding Background Images*.
   - Remove the draggable items for the example drag and drop problem. Select **Add an item** to add draggable items for your problem. Select a matching target zone for each item in the **Zone** field. Add a label, success feedback text, and error feedback text. For more information about how the text in these fields appears, see *Overview of Drag and Drop Problems*. For more information about draggable items, see *Understanding Draggable Items*.

Understanding the Drag and Drop Editing Controls

The following table explains the controls in the **Editing** dialog box.

Control	Explanation
Problem title	The heading that appears above the drag and drop problem. For an example, see *Overview of Drag and Drop Problems*.
Show title	Controls whether the problem title appears above the problem in the LMS.
Problem mode	Controls whether the problem allows learners to experiment with matching draggable items to target zones (standard mode) or requires learners to match all items before providing feedback and optionally restricts the number of attempts (assessment mode). For more information, see *Choosing a Drag and Drop Problem Mode*.

Continued on next page

Table 11.1 – continued from previous page

Control	Explanation
Maximum attempts (assessment mode only)	Controls the number of times that learners can match items to target zones and submit the problem for grading. If you do not enter a maximum number, learners have unlimited attempts. For more information, see *Choosing a Drag and Drop Problem Mode*.
Maximum score	The total number of points that learners receive for completing the problem. For more information about scores and grading, see Establishing a Grading Policy For Your Course.
Problem text	Text that appears above the problem in the LMS. You can use this text to provide instructions or explain the problem. For an example, see *Overview of Drag and Drop Problems*.
Show "Problem" heading	Controls whether the word **Problem** appears above the problem text.
Introductory Feedback	The text that appears in the feedback section of the problem before a learner begins.
Final Feedback	The text that appears in the feedback section of the problem after a learner matches all items to their target zones.
Background URL	The URL of the image that contains target zones for the problem. The URL can be relative to a file you add to your course or to a file on the web. For more information, see *Understanding Background Images*. You must select **Change background** when you enter a new URL in this field. If you do not select **Change background**, the new value will not be saved when you save other changes in the **Editing** dialog box.
Background description	A description of the background image. This description is used by learners who cannot access the visual image.
Display label names on the image	Controls whether the text for target zones appears on the background image in the LMS.
Display zone borders on the image	Controls whether the outlines of target zones appear on the background image in the LMS.
Zone Text	A name for a target zone. You select the name of a target zone in the configuration for draggable items.
Zone Description	Text that describes a target zone. This description is available to learners who cannot access the target zone visually.
Zone width	The horizontal size of a target zone in pixels.
Zone height	The vertical size of a target zone in pixels.
Zone X	The horizontal distance (in pixels) between the left edge of the background image and the left edge of a target zone.
Zone Y	The vertical distance (in pixels) between the top edge of the background image and the top edge of a target zone.
Zone Alignment	Controls the way that the problem aligns draggable items after learners drop them on a target zone. Available options are "left", "center", and "right".

Continued on next page

Table 11.1 – continued from previous page

Control	Explanation
Add a zone	Adds a set of controls for a new zone to the **Editing** dialog box.
Background color	The color that appears behind the text or image label of a draggable item. You can specify the color using a hexadecimal color code (including the # character) or any other valid CSS color specification. For more information, see the W3C CSS color specification. This is an optional configuration. If you do not set the background color, the LMS will apply the default color to your draggable items.
Text color	The color of the text label for a draggable item. You can specify the color using a hexadecimal color code (including the # character) or any other valid CSS color specification. For more information, see the W3C CSS color specification. This is an optional configuration. If you do not set the background color, the LMS will apply the default color to your text.
Item Text	Controls the text that appears on the draggable item in the problem.
Item Zones	Controls the target zones that match the draggable item. Learners must drag the item to any one of the target zones that you select.
Item Image URL	(Optional) the URL of an image that appears on a draggable item. The image appears on the draggable item in the problem. The URL can be relative to a file you add to your course or to a file on the web.
Item Image description	Text that describes the image label for a draggable item. The description is used by learners who cannot access the visual image label.
Item Success Feedback	The text message that appears above the background image when a learner places a draggable item on its matching target zone. For an example, see *Overview of Drag and Drop Problems*. This is an optional configuration. If you do not enter a success feedback message, the LMS will not display one.
Item Error Feedback	The text message that appears above the background image when a learner places a draggable item on an incorrect matching target zone. For an example, see *Overview of Drag and Drop Problems*. This is an optional configuration. If you do not enter an error feedback message, the LMS will not display one.
Item Show advanced settings	Opens additional controls for configuring a draggable item.
Item Preferred width	The horizontal size of a draggable item as a percent of the problem width. The percent value must be a whole number between 0 and 100.
Add an item	Adds a set of controls for a new draggable item to the **Editing** dialog box.

Changing the Visual Style of a Drag and Drop Problem

You can change the visual appearance of drag and drop problems in your courses.

The **Background color** and **Text color** controls for the draggable items in a drag and drop problem set the appearance of items for an individual problem. You can choose colors for the background and text of items when you create or edit a drag and drop problem.

Dropdown Problem

Note: EdX offers full support for this problem type.

The dropdown problem type is a core problem type that can be added to any course. At a minimum, dropdown problems include a question or prompt and several answer options. By adding hints, feedback, or both, you can give learners guidance and help when they work on a problem.

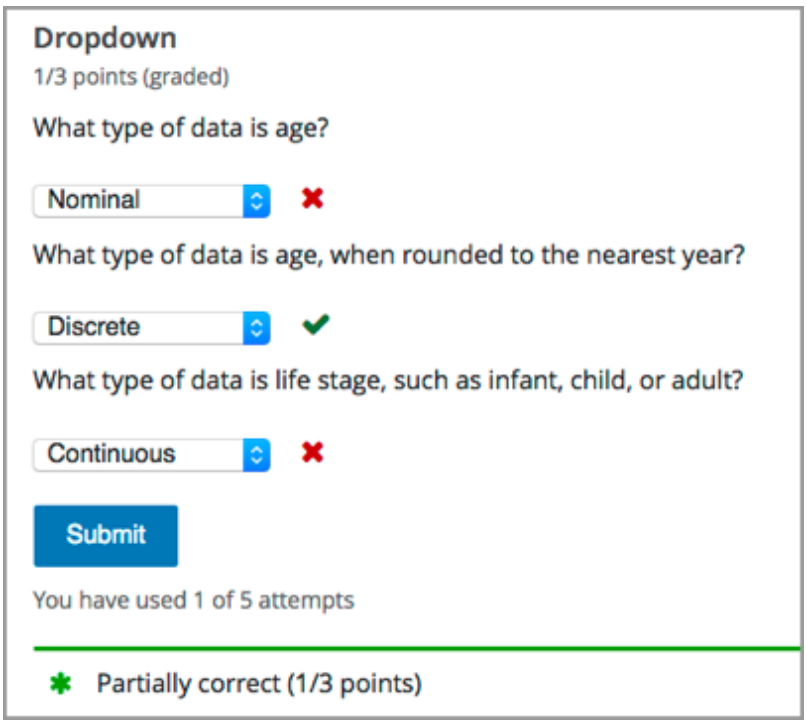
For more information about the core problem types, see Working with Problem Components.

Overview

In dropdown problems, learners select one option from a list of answer options. Unlike *multiple choice* problems, where the answer choices are always visible directly below the question, the answer options for dropdown problems do not appear until the learner selects the dropdown arrow.

Example Dropdown Problem

In the LMS, learners select a single answer option to complete a dropdown problem. An example of a completed dropdown problem follows.



In this example, a single problem component contains multiple questions, all of them using the dropdown problem type. To add the example illustrated above, you enter the following text and Markdown formatting in the simple editor in Studio. Then, select **Settings** for the problem to define settings. To specify that each question is worth one point, leave the **Problem Weight** field empty.

```
>>What type of data is age?<<
[[Nominal, Discrete, (Continuous)]]
---
>>What type of data is age, when rounded to the nearest year?<<
[[
Nominal
(Discrete)
Continuous
]]
---
>>What type of data is life stage, such as infant, child, or adult?<<
[[(Nominal), Discrete, Continuous]]
```

Note: You separate *multiple questions* in a problem component with three hyphen (`---`) characters. You can separate the answer options with either comma (`,`) characters or new lines.

The OLX markup for this example problem follows.

```
<problem>
  <optionresponse>
    <label>What type of data is age?</label>
```

```
    <optioninput options="('Nominal','Discrete','Continuous')"
     correct="Continuous"></optioninput>
  </optionresponse>
  <optionresponse>
    <label>What type of data is age, when rounded to the nearest year?</label>
    <optioninput options="('Nominal','Discrete','Continuous')"
     correct="Discrete"></optioninput>
    </optionresponse>
  <optionresponse>
    <label>What type of data is life stage, such as infant, child, or adult?</label>
    <optioninput options="('Nominal','Discrete','Continuous')"
     correct="Nominal"></optioninput>
  </optionresponse>
</problem>
```

Analyzing Performance on Dropdown Problems

For the dropdown problems in your course, you can use edX Insights to review aggregated learner performance data and examine submitted answers. For more information, see Using edX Insights.

Adding a Dropdown Problem

You add dropdown problems in Studio by selecting the **Problem** component type and then using either the simple editor or the advanced editor to specify the prompt and the answer options.

Note: You can begin work on the problem in the simple editor, and then switch to the advanced editor. However, after you save any changes you make in the advanced editor, you cannot switch back to the simple editor.

Use the Simple Editor to Add a Dropdown Problem

When you add a dropdown problem, you can choose one of these templates.

- **Dropdown**
- **Dropdown with Hints and Feedback**

These templates include the Markdown formatting that you use in the simple editor to add a problem without, or with, hints and feedback. To use the simple editor to add a problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Problem**.
2. From the list of **Common Problem Types**, select the type of problem you want to add. Studio adds a template for the problem to the unit.
3. Select **Edit**. The simple editor opens to a template that shows the Markdown formatting that you use for this problem type.

4. Replace the guidance provided by the template to add your own text for the question or prompt, answer options, explanation, and so on.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.

5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.

6. Select **Save**.

Use the Advanced Editor to Add a Dropdown Problem

You can use the advanced editor to identify the elements of a dropdown problem with OLX. For more information, see *Dropdown Problem OLX Reference*. To use the advanced editor to add a problem, follow these steps.

1. Follow steps 1-3 for creating the problem in the simple editor.
2. Select **Advanced Editor**. The advanced editor opens the template and shows the OLX markup that you can use for this problem type.
3. Replace the guidance provided by the template to add your own text. For example, replace the question or prompt, answer options, and explanation.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.

4. Update the OLX to add optional elements and attributes required for your problem.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Adding Feedback to a Dropdown Problem

For an overview of feedback in problems, see *Adding Feedback and Hints to a Problem*. You can add feedback for each of the answer options you provide in the problem. Use the following guidelines when providing feedback.

- Use feedback for the incorrect answers to target common misconceptions and mistakes.
- Ensure feedback provides some guidance to the learner about how to arrive at the correct answer.
- Use feedback for the correct answer to reinforce why the answer is correct. Because learners are able to guess, ensure that feedback provides a reason why the answer is correct for learners who might have selected that answer by chance.

You can add feedback in a dropdown problem using the simple editor or the advanced editor.

Configuring Feedback in the Simple Editor

You can configure feedback in the simple editor. When you add a dropdown problem, select the template **Dropdown with Hints and Feedback**. This template has example feedback syntax that you can replace.

```
[[
Wrong Answer {{Feedback for learners who select this answer.}}
Wrong Answer {{Feedback for learners who select this answer.}}
(Correct Answer) {{Feedback for learners who select this answer.}}
]]
```

Note: When you include feedback, you might find it more convenient to use new lines to separate the answer options.

For example, the following problem has feedback for each possible answer.

```
>>A/an _________ is an example of a vegetable.<<

[[
  apple {{An apple is the fertilized ovary that comes from an apple tree and
    contains seeds classifying it as a fruit.}}
  pumpkin {{A pumpkin is the fertilized ovary of a squash plant and contains
    seeds classifying it as a fruit.}}
  (potato) {{A potato is an edible part of a plant in tuber form and is
    classified as a vegetable}}
  tomato {{Many people mistakenly think a tomato is a vegetable. However,
    because a tomato is the fertilized ovary of a tomato plant and contains
    seeds it is classified as a fruit.}}
]]
```

Configuring Feedback in the Advanced Editor

In the advanced editor, you configure answer feedback with the following syntax.

```
<option correct="False">Option Label
  <optionhint>Feedback for when a learner selects this incorrect answer.</optionhint>
</option>
```

For example, the following problem has feedback for each answer.

```
<problem>
  <optionresponse>
    <label>A/an _________ is an example of a vegetable.</label>
    <optioninput>
      <option correct="False">apple
        <optionhint>An apple is the fertilized ovary that comes from an
         apple tree and contains seeds classifying it as a fruit.</optionhint>
      </option>
      <option correct="False">pumpkin
        <optionhint>A pumpkin is the fertilized ovary of a squash plant and
         contains seeds classifying it as a fruit.</optionhint>
      </option>
      <option correct="True">potato
        <optionhint>A potato is an edible part of a plant in tuber form and
         is classified as a vegetable.</optionhint>
      </option>
      <option correct="False">tomato
        <optionhint>Many people mistakenly think a tomato is a vegetable.
         However, because a tomato is the fertilized ovary of a tomato plant
         and contains seeds it is classified as a fruit.</optionhint>
      </option>
    </optioninput>
  </optionresponse>
</problem>
```

Customizing Feedback Labels

By default, the feedback labels shown to learners are **Correct** and **Incorrect**. If you do not define feedback labels, learners see these terms when they submit an answer, as in the following example.

```
Incorrect:
An apple is the fertilized ovary that comes from an apple tree and contains
seeds classifying it as a fruit.
```

You can configure the problem to override the default labels. For example, you can configure a custom label for a specific wrong answer.

```
Not Quite:
Many people mistakenly think a tomato is a vegetable. However, because a
tomato is the fertilized ovary of a tomato plant and contains seeds it is
classified as a fruit.
```

Note: The default labels **Correct** and **Incorrect** display in the learner's requested language. If you provide custom labels, they display as you define them to all learners. They are not translated into different languages.

Customize Feedback Labels in the Simple Editor

In the simple editor, you configure custom feedback labels with the following syntax.

```
[[
Incorrect Answer {{Label:: Feedback for learners who select this answer.}}
.
.
.
]]
```

That is, you provide the label text, followed by two colon (:) characters, before the feedback text.

For example, the following feedback is configured to use a custom label.

```
[[
tomato {{Not Quite:: Many people mistakenly think a tomato is a
vegetable. However, because a tomato is the fertilized ovary of a tomato
plant and contains seeds, it is a fruit.}}
.
.
.
]]
```

Customize Feedback Labels in the Advanced Editor

In the advanced editor, you configure custom feedback labels with the following syntax.

```
<option correct="False">Answer
  <optionhint label="Custom Label">Feedback for learners who select this answer.</
↪optionhint>
</option>
```

For example, the following feedback is configured to use a custom label.

```
<option correct="False">tomato
  <optionhint label="Not Quite">Many people mistakenly think a tomato is a
   vegetable. However, because a tomato is the fertilized ovary of a tomato
   plant and contains seeds it is classified as a fruit.</optionhint>
</option>
```

Adding Hints to a Dropdown Problem

You can add hints to a dropdown problem using the simple editor or the advanced editor. For an overview of hints in problems, see *Adding Feedback and Hints to a Problem*.

Configure Hints in the Simple Editor

In the simple editor, you configure hints with the following syntax.

```
||Hint 1||
||Hint 2||
||Hint n||
```

Note: You can configure any number of hints. The learner views one hint at a time and views the next one by selecting **Hint** again.

For example, the following problem has two hints.

```
||A fruit is the fertilized ovary from a flower.||
||A fruit contains seeds of the plant.||
```

Configure Hints in the Advanced Editor

In the advanced editor, you add the `<demandhint>` element immediately before the closing `</problem>` tag, and then configure each hint using the `<hint>` element.

```
.
.
.
<demandhint>
  <hint>Hint 1</hint>
  <hint>Hint 2</hint>
  <hint>Hint 3</hint>
</demandhint>
</problem>
```

For example, the following OLX for a multiple choice problem shows two hints.

```
.
.
.
</multiplechoiceresponse>
<demandhint>
  <hint>A fruit is the fertilized ovary from a flower.</hint>
```

```
  <hint>A fruit contains seeds of the plant.</hint>
</demandhint>
</problem>
```

Dropdown Problem OLX Reference

Template

```
<problem>
  <optionresponse>
    <label>Question or prompt text</label>
    <description>Optional information about how to answer the question</description>
    <option correct="False">Option Label
      <optionhint>Feedback for when learner selects this answer.</optionhint>
    </option>
    <option correct="True">Option Label
      <optionhint>Feedback for when learner selects this answer.</optionhint>
    </option>
    <solution>
      <div class="detailed-solution">
        <p>Explanation or Solution Header</p>
        <p>Explanation or solution text</p>
      </div>
    </solution>
  </optionresponse>
  <demandhint>
    <hint>Hint 1</hint>
    <hint>Hint 2</hint>
    <hint>Hint 3</hint>
  </demandhint>
</problem>
```

Elements

For dropdown problems, the `<problem>` element can include this hierarchy of child elements.

```
<optionresponse>
    <label>
    <description>
    <optioninput>
          <option>
              <optionhint>
    <solution>
<demandhint>
    <hint>
```

In addition, standard HTML tags can be used to format text.

`<optionresponse>`

Required. Indicates that the problem is a dropdown problem.

Attributes

None.

Children

- <label>
- <description>
- <optioninput>
- <solution>

<label>

Required. Identifies the question or prompt. You can include HTML tags within this element.

Attributes

None.

Children

None.

<description>

Optional. Provides clarifying information about how to answer the question. You can include HTML tags within this element.

Attributes

None.

Children

None.

<optioninput>

Required. Designates an answer option.

Attributes

Attribute	Description
`options`	Either this attribute or a set of `<option>` child elements for `<optioninput>` is required. Accepts a comma separated list of values in the following format. `options="('Answer1','Answer2', 'Answer3')"`
`correct`	Used if the `options` attribute is set. Required. Indicates which of the answer options is correct.

Children

- `<option>`
- `<optionhint>`

`<option>`

Designates an answer option. Either a set of `<option>` child elements or the `options` attribute for `<optioninput>` is required.

Attributes

Attribute	Description
`correct`	Required. Indicates whether the answer option is correct or incorrect. When set to `"true"`, the choice is a correct answer. At least one required. When set to `"false"`, the choice is an incorrect answer.

If the `<option>` element is used, `<optionhint>` is a child of `<option>`.

`<optionhint>`

Optional. Specifies feedback for the answer.

Attributes

None.

Children

None.

<solution>

Optional. Identifies the explanation or solution for the problem, or for one of the questions in a problem that contains more than one question.

This element contains an HTML division . The division contains one or more paragraphs <p> of explanatory text.

<demandhint>

Optional. Specifies hints for the learner. For problems that include multiple questions, the hints apply to the entire problem.

Attributes

None.

Children

<hint>

<hint>

Required. Specifies additional information that learners can access if needed.

Attributes

None.

Children

None.

Full Screen Image Tool

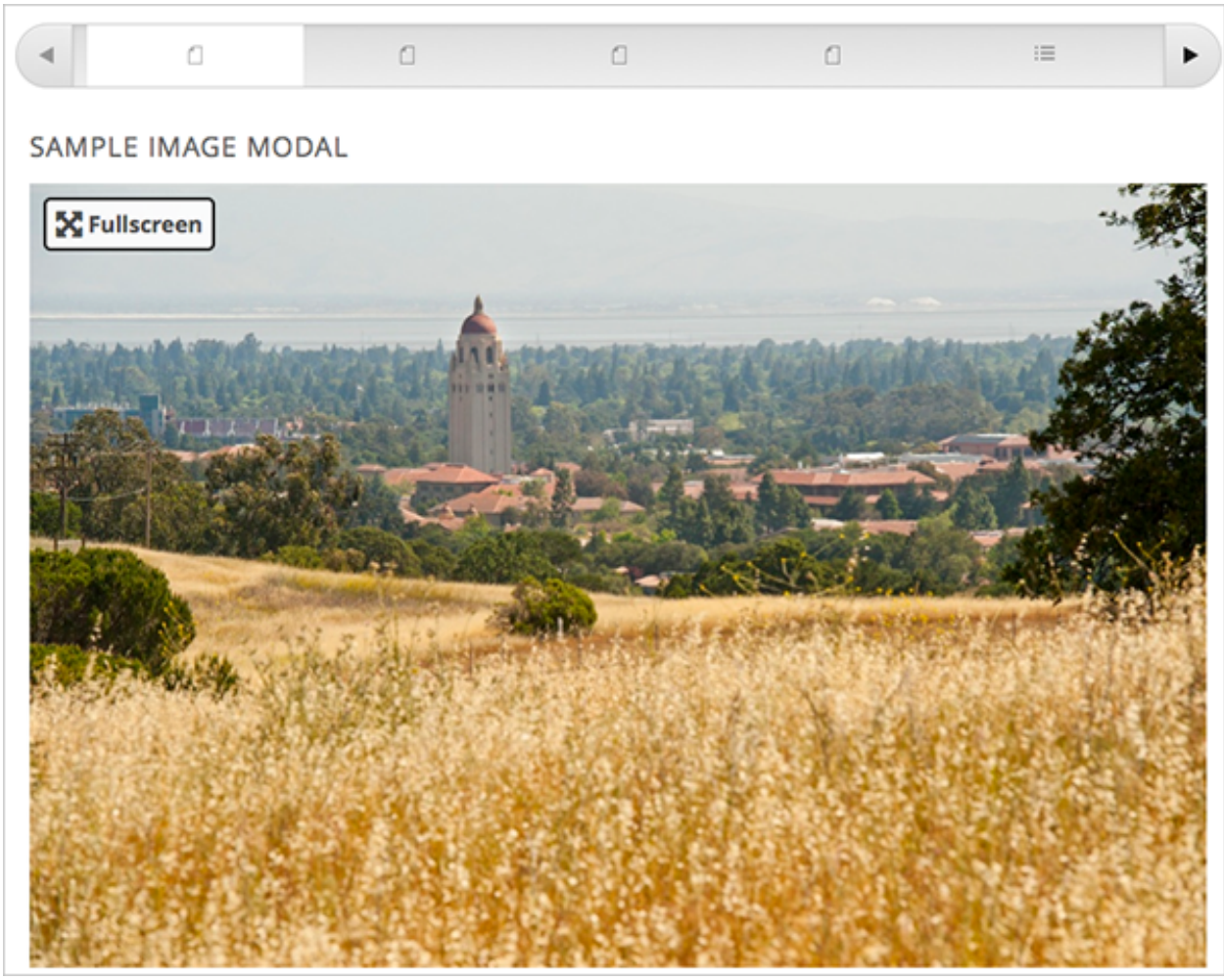
Note: EdX does not support this problem type.

Some large images are difficult for learners to view in the courseware. The full screen image tool allows learners to enlarge the image, so they can see all the detail in context.

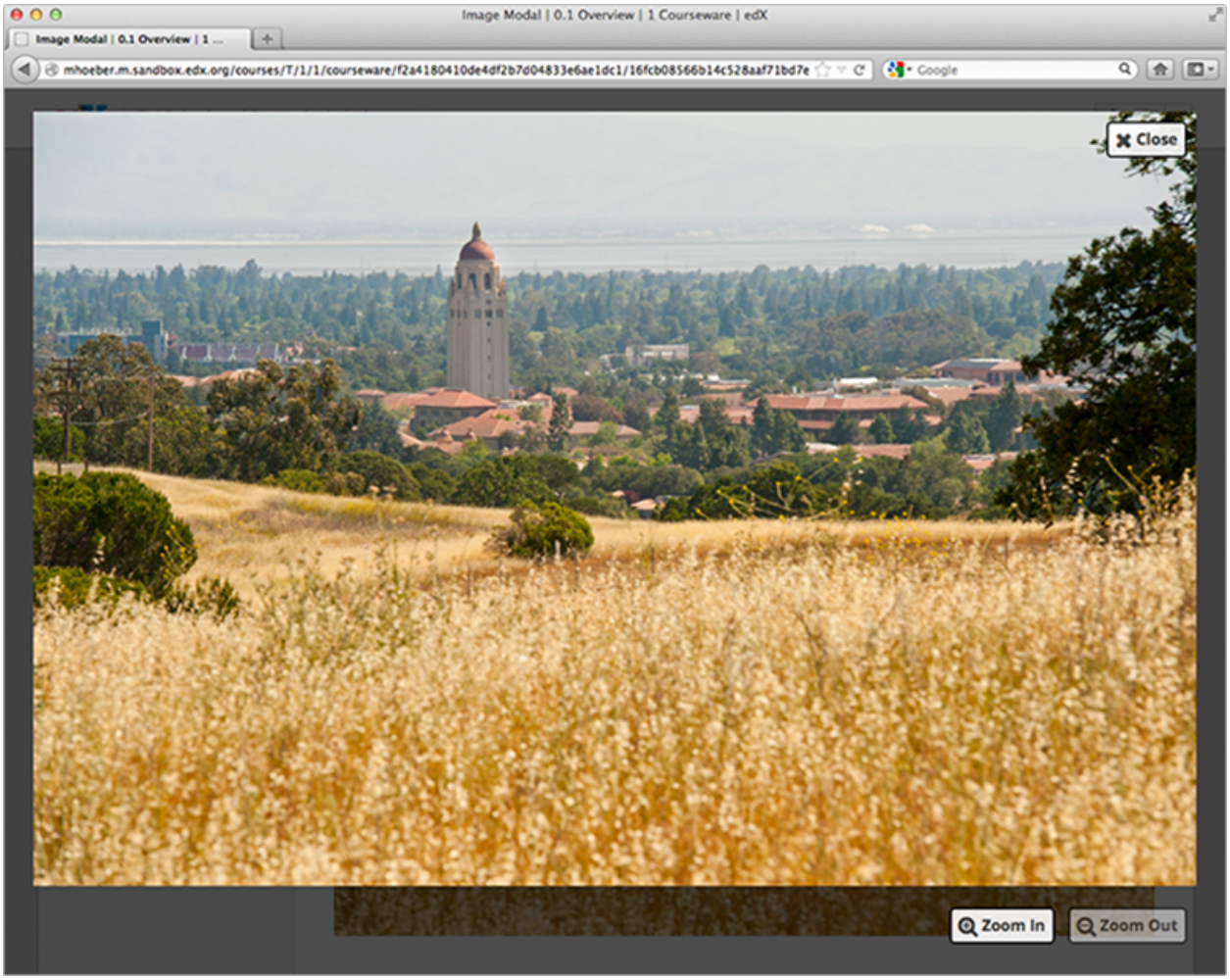
- *The Learner View of a Full Screen Image*
- *Create a Full Screen Image*

The Learner View of a Full Screen Image

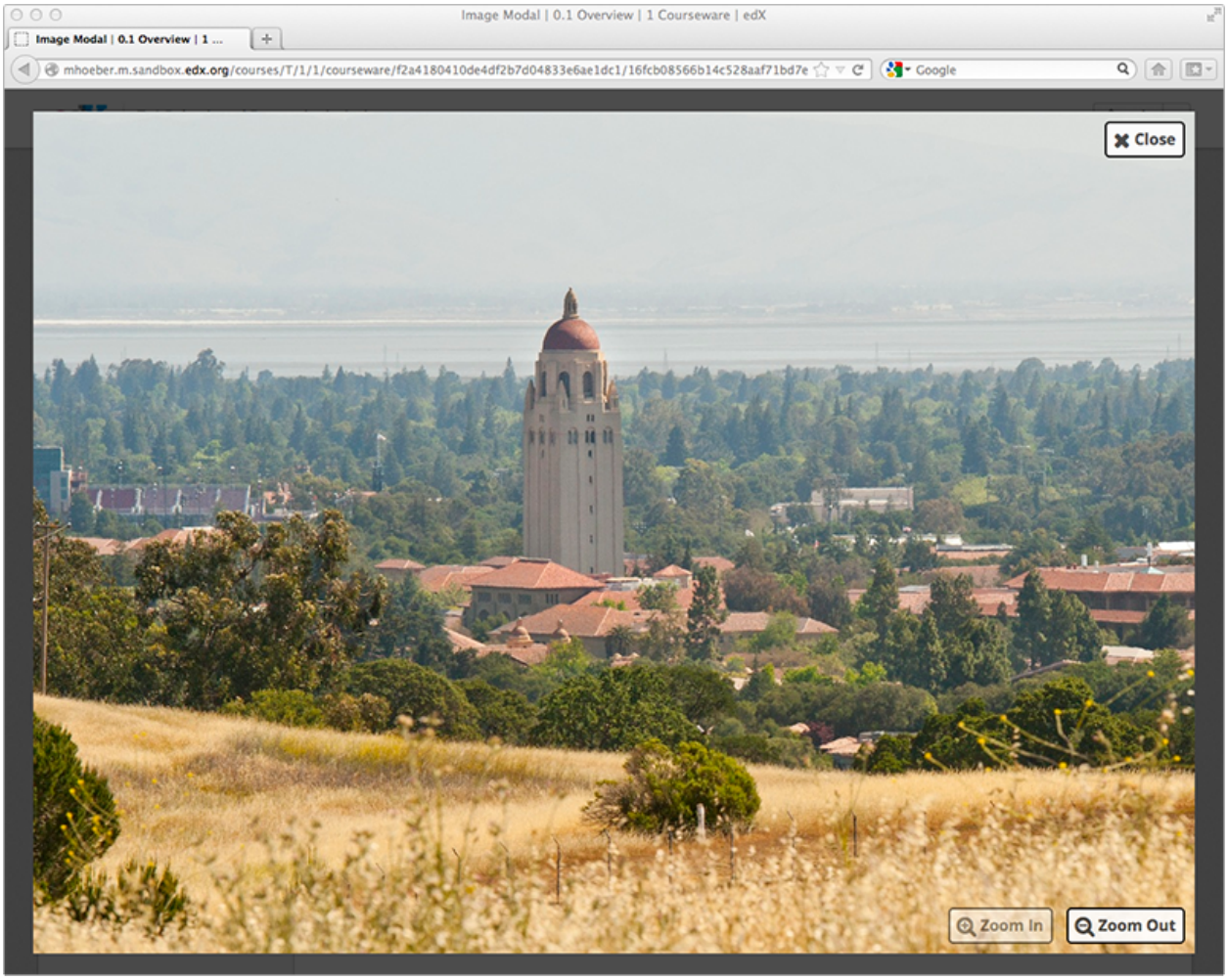
The learner sees the full screen image in a unit page. When the learner moves the cursor over the image, the **Fullscreen** option appears.



When the learner selects **Fullscreen**, the image opens and expands in the full browser window. **Close**, **Zoom In**, and **Zoom Out** options appear.



The learner can then zoom in on the image, and drag the image to view a specific part of it.



Create a Full Screen Image

1. Upload your image file to the **Files & Uploads** page. For more information, see Adding Files to a Course.
2. Under **Add New Component**, select **HTML**, and then select **Full Screen Image Tool**.
3. In the new component that appears, select **Edit**.
4. In the component editor, replace the default title, remove the instructional paragraph, and add text as needed.
5. Select **HTML** from the toolbar to edit the HTML source code.
6. Scroll down in the file, and then replace the following placeholders with your own content.
   - Replace the value of the `<a>` element's `href` attribute with the path to your image. Do not change the value of the class attribute. For example:

     ```
     <a href="/static/Image1.jpg" class="modal-content">
     ```

   - Replace the value of the `<img>` element's `src` attribute with the path to your image. For example:

     ```
     <img alt="Full screen image" src="/static/Image1.jpg"/>
     ```

   - Ensure that the value of the `href` and `src` attributes are the same, and that you do not change the class attribute. Your sample code should look like the following example.

```
<h3>Sample Image Modal</h3>
<a href="/static/Image1.jpg" class="modal-content">
<img alt="Full screen image" src="/static/Image1.jpg"/>
</a>
```

Note: You can use this same HTML code in any HTML component, not just those components you created as full screen images.

7. Select **Save**.

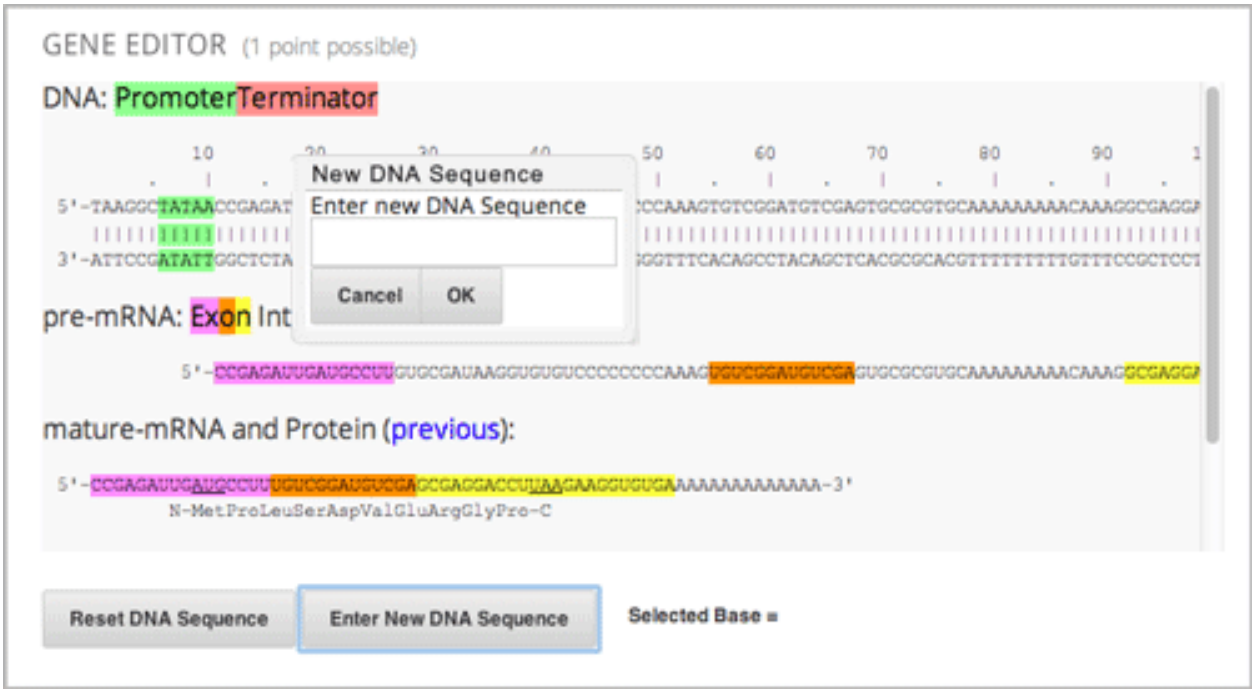
Gene Explorer Tool

Note: EdX does not support this tool.

The gene explorer (GeneX), from the biology department at UMB, simulates the transcription, splicing, processing, and translation of a small hypothetical eukaryotic gene. GeneX allows learners to make specific mutations in a gene sequence, and it then calculates and displays the effects of the mutations on the mRNA and protein.

Specifically, the gene explorer does the following:

1. Finds the promoter and terminator.
2. Reads the DNA sequence to produce the pre-mRNA.
3. Finds the splice sites.
4. Splices and tails the mRNA.
5. Finds the start codon.
6. Translates the mRNA.



For more information about the gene explorer, see The Gene Explorer.

Gene Explorer Code

```
<problem>
<p>Make a single base pair substitution mutation in the gene below that results in a␣
↪protein that is longer than the protein produced by the original gene. When you are␣
↪satisfied with your change and its effect, click the <b>SUBMIT</b> button.</p>
<p>Note that a "single base pair substitution mutation" is when a single base is␣
↪changed to another base; for example, changing the A at position 80 to a T.␣
↪Deletions and insertions are not allowed.</p>
<script type="loncapa/python">
def genex_grader(expect,ans):
    if ans=="CORRECT": return True
    import json
    ans=json.loads(ans)
    return ans["genex_answer"]=="CORRECT"
</script>
<customresponse cfn="genex_grader">
<editageneinput width="818" height="1000" dna_sequence=
↪"TAAGGCTATAACCGAGATTGATGCCTTGTGCGATAAGGTGTGTCCCCCCCCCAAAGTGTCGGATGTCGAGTGCGCGTGCAAAAAAAAACAAAGGCGAG
↪" genex_dna_sequence=
↪"TAAGGCTATAACCGAGATTGATGCCTTGTGCGATAAGGTGTGTCCCCCCCCCAAAGTGTCGGATGTCGAGTGCGCGTGCAAAAAAAAACAAAGGCGAG
↪" genex_problem_number="2"/>
</customresponse>
</problem>
```

In this code:

- **width** and **height** specify the dimensions of the application, in pixels.
- **genex_dna_sequence** is the default DNA sequence that appears when the problem opens.
- **dna_sequence** contains the application's state and the learner's answer. This value must be the same as **genex_dna_sequence**.
- **genex_problem_number** specifies the number of the problem. This number is based on the five gene editor problems in the MITx 7.00x course–for example, if you want this problem to look like the second gene editor problem in the 7.00x course, you would set the **genex_problem_number** value to 2. The number must be 1, 2, 3, 4, or 5.

Google Calendar Tool

Note: EdX offers provisional support for this tool.

This topic describes how to embed Google Calendars in your course.

Before you make content from an external site available through your course, be sure to review the content to ensure that it is accessible to people with disabilities. For more information, see Accessibility Best Practices for Developing Course Content.

You can also add Google Drive files, such as documents, spreadsheets, and images, to your course. For more information, see *Google Drive Files Tool*.

Note: Google services are not available in some regions and countries. If Google services are not available in a learner's area, the learner might see an "image unavailable" message in the place of the Google Drive file or Google Calendar. EdX strongly suggests that you provide alternative resources for learners in these areas.

Overview

You can embed a Google Calendar in your course so that learners see the calendar in the course body. You can use a Google Calendar to share quiz dates, office hours, or other schedules of interest to learners.

Embedding a Google Calendar in Your Course

To embed a Google Calendar in your course, follow these steps.

Enable the Google Calendars Tool

Before you can add Google Calendars to your course, you must enable the Google Calendars tool in Studio or OLX (open learning XML).

To enable the Google Calendars tool in Studio, you add the `"google-calendar"` key to the **Advanced Module List** on the **Advanced Settings** page. (Be sure to include the quotation marks around the key value.) For more information, see Enabling Additional Exercises and Tools.

Alternatively, you can use OLX to enable the Google Calendars tool.

Enable Google Calendars in OLX

To enable Google Calendars in your course, you edit the XML file that defines the course structure. You locate the `course` element's `advanced_modules` attribute, and add the string `google-calendar` to it.

For example, the following XML code enables Google Calendars in a course. It also enables Google Drive files.

```
<course advanced_modules="[&quot;google-document&quot;,
    &quot;google-calendar&quot;]" display_name="Sample Course"
    start="2014-01-01T00:00:00Z">
    ...
</course>
```

For more information, see OLX Course Building Blocks in the *EdX Open Learning XML Guide*.

Make the Google Calendar Public and Obtain Its ID

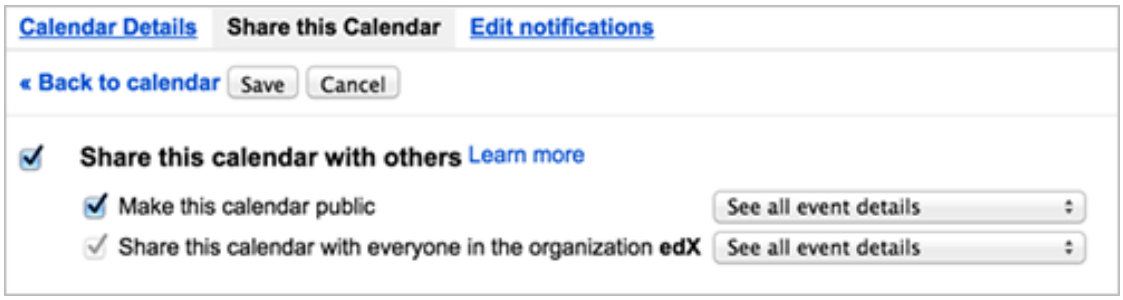
Before you can add a Google Calendar to your course, you must make the Calendar public and obtain its ID.

Important: The tasks described in this section rely on the use of third-party software. Because the software is subject to change by its owner, the steps provided here are intended as guidelines and not as an exact procedure.

Make the Google Calendar Public

1. Open the Google Calendar.
2. From the **Settings** menu, select **Settings**.
3. Select the **Calendars** tab.

   You might have multiple calendars on the **Calendars** tab. Find the calendar that you want to share in your course.
4. In the row for the calendar to share, in the **Sharing** column, select **Edit Settings**.
5. Select the **Share this Calendar** tab, and then select **Make this calendar public**.

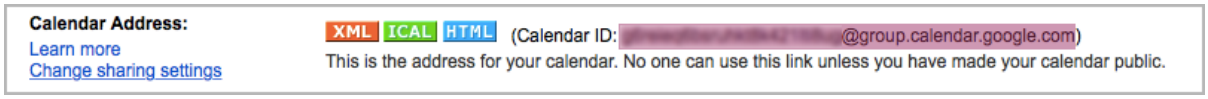


1. Select **Save**.

   The **Calendar Settings** page closes, and you return to the **Calendars** tab. You continue by *obtaining the Google Calendar's ID*.

Obtain the Google Calendar ID

1. On the **Calendars** tab, select the name of the calendar.
2. Select the **Calendar Details** tab.
3. Next to the **Calendar Address** label, look to the right of the three colored **XML**, **ICAL**, and **HTML** buttons. In parentheses, you can see the calendar ID.



The calendar ID resembles the following text.

```
abcdefghijklmnop1234567890@group.calendar.google.com
```

Select and copy the calendar ID. You use this value to configure the Google Calendar component in your course.

Add a Google Calendar in the Course Body

To add a Google calendar in the course body, you create an advanced component in Studio or create a Google Calendar XBlock in OLX.

Add a Google Calendar Component in edX Studio

Make sure that you *enable Google Calendars* for your course before you add a Google Calendar component.

To add a Google Calendar component, follow these steps.

1. On the **Course Outline** page, open the unit where you want to add the Google Calendar component.
2. Under **Add New Component**, select **Advanced**, and then select **Google Calendar**.

   The new component is added to the unit, with the default edX Google Calendar embedded.
3. In the new component, select **Edit**.
4. In the **Display Name** field, type the name for the component.
5. In the **Public Calendar ID** field, paste the calendar ID you copied in the *Obtain the Google Calendar ID* task.
6. For the **Default View** field, select **Month**, **Week**, or **Agenda**.

   This is the initial view that your learners have of the calendar. Each learner can change his or her view.
7. Select **Save**.

You can then Previewing Draft Content to see how the unit with the Google Calendar will appear to learners.

Add a Google Calendar XBlock in OLX

To add a Google Calendar XBlock in OLX, create the `google-calendar` element. You can embed this element in the `vertical` element, or you can embed this element in its own file that is referenced within the vertical. For more information, see OLX Course Building Blocks in the *EdX Open Learning XML Guide*.

An example follows.

```
<google-calendar url_name="4115e717366045eaae7764b2e1f25e4c"
  calendar_id="abcdefghijklmnop1234567890@group.calendar.google.com"
  default_view="1" display_name="Class Schedule"/>
```

The value of the `calendar_id` attribute is the calendar ID that you copied in the *Obtain the Google Calendar ID* task.

Note: The edX Learning Management System sets the height and width values for Google Calendars. If you add these attributes, the LMS overrides your changes.

Editing Google Calendars

When you make changes to a Google Calendar that is embedded in your course, learners see the updates immediately. You make changes to calendars with the Google user interface. You do not need to edit the Google Calendar component.

Google Drive Files Tool

Note: EdX offers provisional support for this tool.

This topic describes how to embed Google Drive files, such as documents, spreadsheets, and images, in your course.

Before you make content from an external site available through your course, be sure to review the content to ensure that it is accessible to people with disabilities. For more information, see Accessibility Best Practices for Developing Course Content.

You can also use *Google calendars* in the course body. For more information, see *Google Calendar Tool*.

Note: Google services are not available in some regions and countries. If Google services are not available in a learner's area, the learner might see an "image unavailable" message in the place of the Google Drive file or calendar. EdX strongly suggests that you provide alternative resources for learners in these areas.

Overview

You can embed a Google Drive file in your course so that learners see the file in the course body. For example, you can share a Google spreadsheet with learners.

You can embed the following types of Google Drive files.

- Google Docs (text documents)
- Google Drawings (images)
- Google Forms (forms or surveys)
- Google Slides (presentations)
- Google Sheets (spreadsheets)

Embedding a Google Drive File in Your Course

To embed a Google Drive file in your course, follow these steps.

Enable the Google Drive Files Tool

Before you can add Google Drive files to your course, you must enable the Google Drive tool in Studio or OLX (open learning XML).

To enable the Google Drive tool in Studio, you add the `"google-document"` key to the **Advanced Module List** on the **Advanced Settings** page. For more information, see Enabling Additional Exercises and Tools.

Alternatively, you can use OLX to enable the Google Drive tool.

Enable Google Drive Files in OLX

To enable Google Drive files in your course, you edit the XML file that defines the course structure. You locate the `course` element's `advanced-modules` attribute, and add the string `google-document` to it.

For example, the following XML code enables Google Drive files in a course. It also enables Google calendars.

```
<course advanced_modules="[&quot;google-document&quot;,
    &quot;google-calendar&quot;]" display_name="Sample Course"
    start="2014-01-01T00:00:00Z">
    ...
</course>
```

For more information, see OLX Course Building Blocks in the *EdX Open Learning XML Guide*.

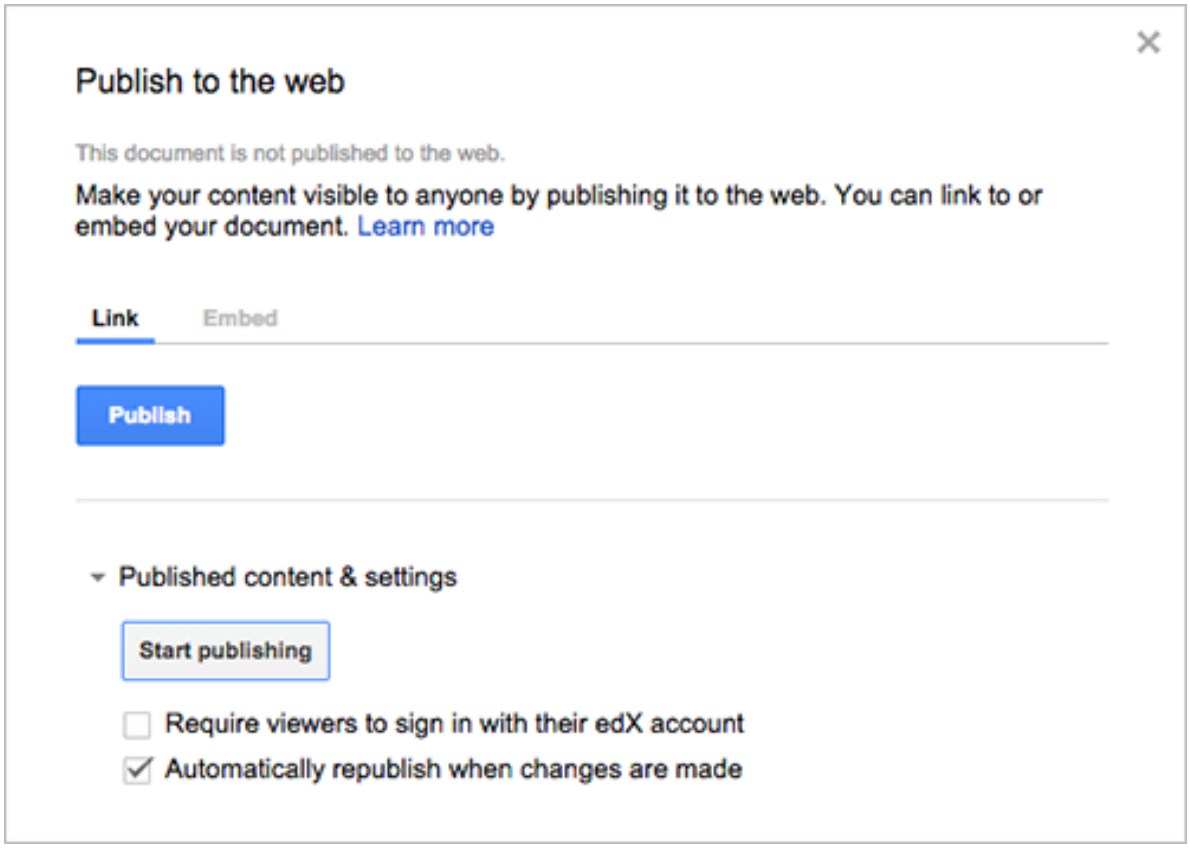
Publish the Google Drive File and Obtain the Embed Code

Before you can add a Google Drive file to your course, you must publish the file to the web and obtain the embed code for the file.

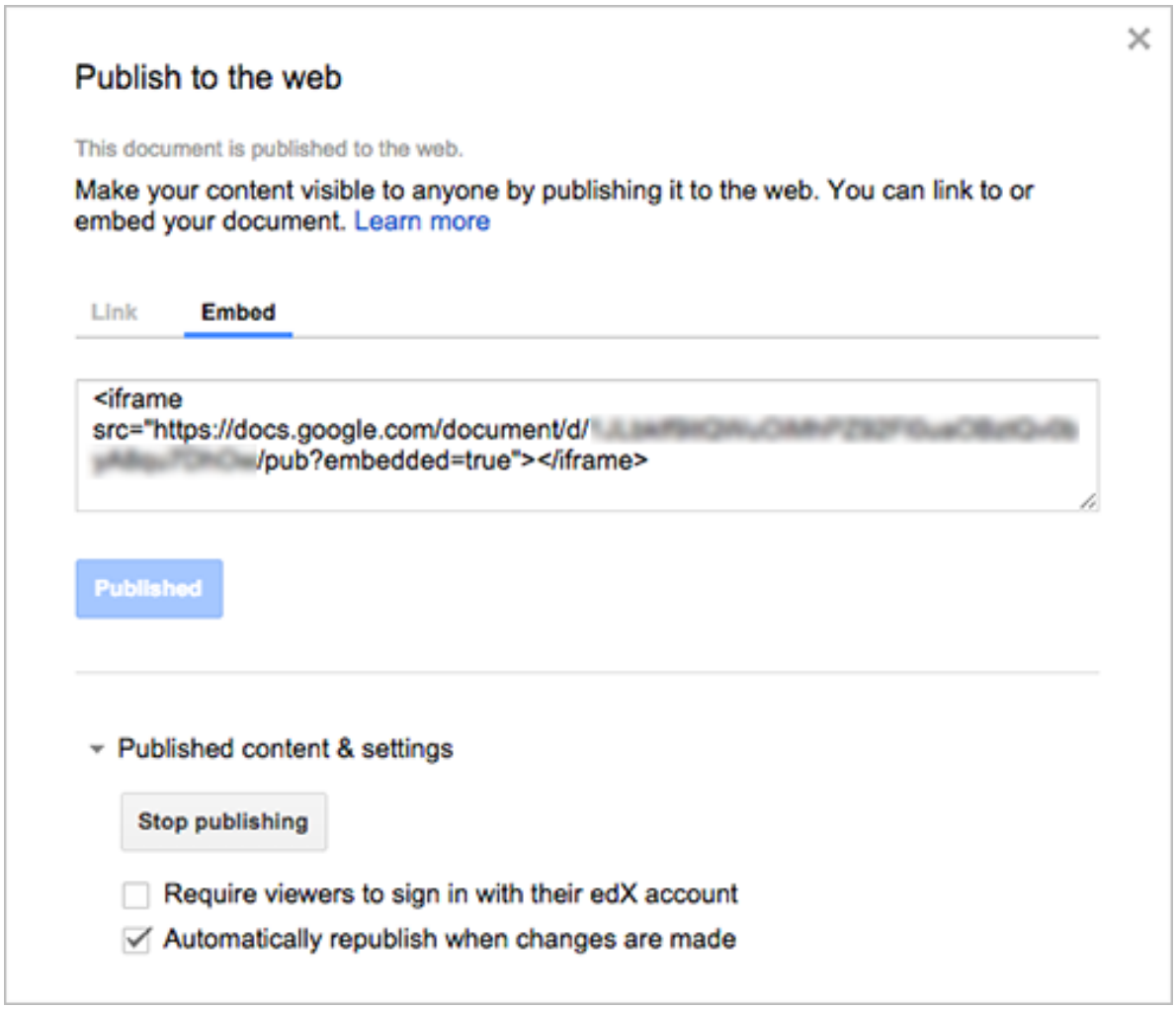
Important: The task described in this section relies on the use of third-party software. Because the software is subject to change by its owner, the steps provided here are intended as guidelines and not as an exact procedure.

---

1. Open the Google Drive file.
2. From the **File** menu, select **Publish to the web**.



3. Select **Publish**, and then select **OK** to confirm the action.
4. Select the **Embed** tab.



5. Copy the complete string in the **Embed** field, including the `<iframe>` tags.

Note: Google images do not have an `<iframe>` tag. To embed an image, you copy the complete `img` tag.

You use that string to configure the Google Drive file component.

Add a Google Drive File to Your Course

To add a Google Drive file in the course body, you create an advanced component in Studio or create a Google Document XBlock in OLX.

Add a Google Drive File Component in edX Studio

Ensure you *enable Google Drive files* before you add the component.

To add a Google Drive file component, follow these steps.

1. On the Course Outline page, open the unit where you want to add the Google Drive component.
2. Under **Add New Component**, select **Advanced**, and then select **Google Document**.

The new component is added to the unit, with the default Google presentation embedded.

3. In the new component, select **Edit**.
4. In the **Display Name** field, enter the name for the component.
5. In the **Embed Code** field, paste the embed code that you copied in the *Obtain the Google Drive File Embed Code* task.
6. Select **Save**.

You can then Previewing Draft Content to see how the unit with the Google drive file will appear to learners.

Add a Google Drive File XBlock in OLX

To add a Google Drive file XBlock in OLX, you create the `google-document` element. You can embed the `google-document` element in the `vertical` element, or you can create the `google-document` element as a stand-alone file that you reference in the vertical.

For more information, see OLX Course Building Blocks in the *EdX Open Learning XML Guide*.

For example:

```
<google-document url_name="c5804436419148f68e2ee44abd396b12"
  embed_code="&lt;iframe
  frameborder=&quot;0&quot; src=&quot;https://docs.google.com/spreadsheet/pub
  ?key=0AuZ_5O2JZpH5dGVUVDNGUE05aTFNcEl2Z0ZuTUNmWUE&amp;output=html&amp;widge
  t=true&quot;&gt;&lt;/iframe&gt;" display_name="Google Document"/>
```

The value of the `embed_code` attribute is the embed code you copied in the *Obtain the Google Drive File Embed Code* task.

Note: The edX Learning Management System sets the height and width values for Google Drive files. If you add these attributes, the LMS overrides your changes.

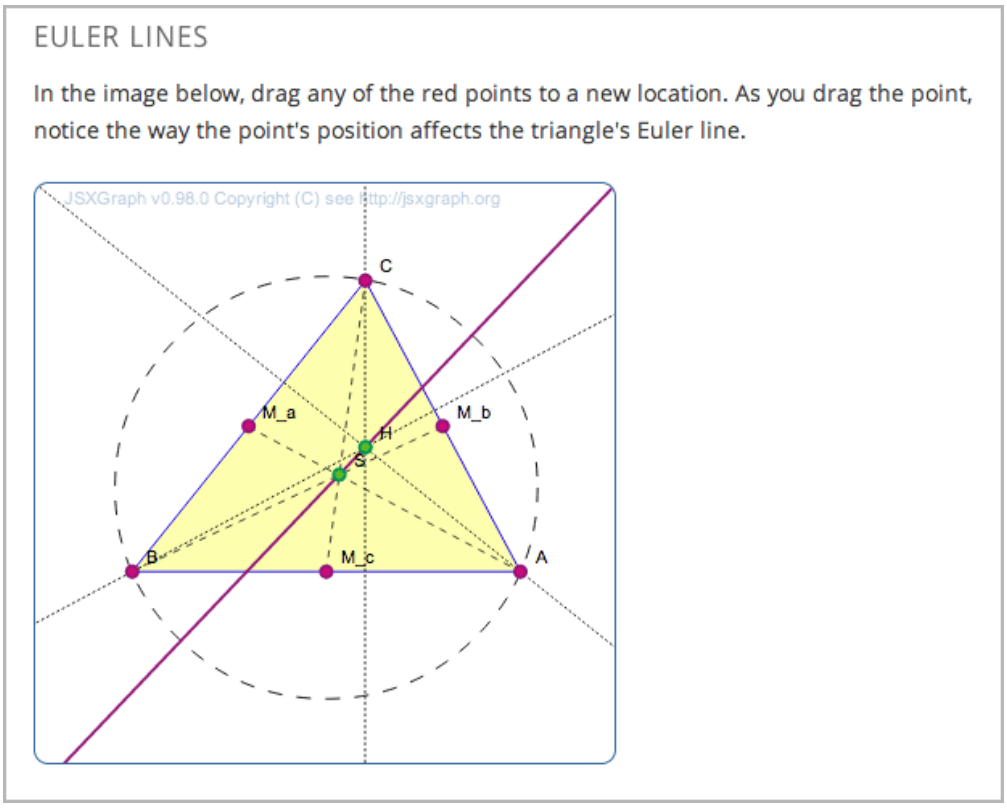
Editing Google Drive Files

When you edit and save a Google Drive file that is embedded in your course, learners see the updates immediately. You make changes to files with the Google user interface. You do not need to edit the Google Document component.

Iframe Tool

Note: EdX offers provisional support for this tool.

The iframe tool allows you to integrate ungraded exercises and tools from any Internet site into the body of your course. It places an iframe inside an HTML component, and then the exercise or tool appears inside the iframe. The iframe can include your own tools or third-party tools.



Before you make content or a tool from an external site available through your course, be sure to review the content or tool to ensure that it is accessible to people with disabilities. For example, in addition to testing the iframe components that you add to your course, you can ask third party providers to confirm that a tool is accessible, and, if available, contact your on campus accessibility support services for additional guidance. For more information, see Accessibility Best Practices for Developing Course Content.

Iframes are well-suited for tools that demonstrate a concept, but that are not graded or used to store student data. If you want to add a graded tool or exercise, add the tool as a *custom JavaScript problem* or an *LTI component*.

For more information about iframes, see the HTML/Elements/iframe specification.

Create an IFrame Tool

To add an exercise or tool in an iframe, you create an iframe HTML component and add the URL of the page that contains the exercise or tool to the component. You can also add text and images both before and after the iframe.

Note: The URL of the page that contains the exercise or tool must start with `https` instead of `http`. If the URL starts with `http`, work with the owner of that page to find out if an `https` version is available. Some websites do not allow their content to be embedded in iframes.

---

1. Under **Add New Component**, select **HTML**, and then select **IFrame Tool**.
2. In the new component, select **Edit**.
3. In the visual editor toolbar, select **HTML**.
4. In the HTML source code editor, locate the following HTML (line 7). This HTML includes the `<iframe>` element.

```
<p><iframe src="https://studio.edx.org/c4x/edX/DemoX/asset/eulerLineDemo.html"␣
↪width="402" height="402" marginwidth="0" marginheight="0" frameborder="0"␣
↪scrolling="no">You need an iFrame capable browser to view this.</iframe></p>
```

5. Replace the default URL in the `src` attribute (`https://studio.edx.org/c4x/edX/DemoX/asset/eulerLineDemo.html`) with the URL of the page that contains the exercise or tool. **This URL must start with https**. Make sure that you do not delete the quotation marks that surround the URL.
6. Change the attributes in the iframe element to specify any other settings that you want. For more information about these settings, see *Iframe Settings*. You can also change the text between the opening and closing `<iframe>` tags. Learners see this text only when they use a browser that does not support iframes.
7. Select **OK** to close the HTML source code editor and return to the visual editor.
8. In the visual editor, replace the default text with your own text.
9. Select **Save**.

Iframe Settings

To specify settings for your iframe, you add, remove, or change the attributes inside the opening `<iframe>` tag. The `<iframe>` tag **must** contain a `src` attribute and a `title` attribute. Other attributes are optional.

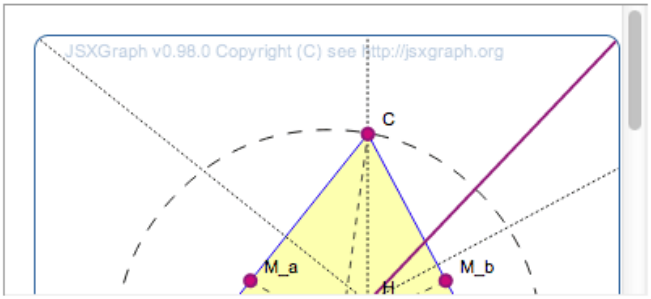
You can add these attributes in any order you want.

Attribute	Description
`src` (required)	Specifies the URL of the page that contains the exercise or tool, beginning with https.
`title` (required)	Describes the content or its purpose in the context of the course.
`width` and `height` (optional)	Specify the width and height of the iframe, in pixels or as a percentage. To specify the value in pixels, enter numerals. To specify a percentage, enter numerals followed by a percent sign. If you do not specify the width and height, the iframe uses the dimensions that the linked page has set. These dimensions vary by website. If you change the width and height of the iframe, the content from the linked page might be resized, or only part of the content may appear.
`marginwidth` and `marginheight` (optional)	Specify the size of the space between the edges of the iframe and your exercise or tool, in pixels.
`frameborder` (optional)	Specifies whether a border appears around your iframe. If the value is 0, no border appears. If the value is any positive number, a border appears.
`scrolling` (optional)	For an iframe that is smaller than the exercise or tool it contains, specifies whether a scrollbar appears to help users access all of the iframe's content. For example, if the content in your iframe is very tall, you can set the iframe's height to a smaller number and add a vertical scroll bar for users, as in the first image below.

For example, compare how the different settings in each of the `<iframe>` elements below affect the iframe.

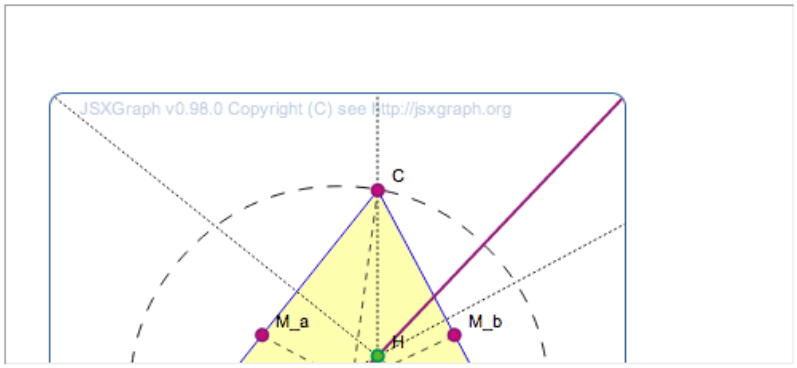
```
<p><iframe src="https://studio.edx.org/c4x/edX/DemoX/asset/eulerLineDemo.html" width=
↪"442" height="200" marginwidth="20" marginheight="20" frameborder="1" scrolling="yes
↪">You need an iFrame capable browser to view this.</iframe></p>
```

In the image below, drag any of the red points to a new location. As you drag the point, notice the way the point's position affects the triangle's Euler line.




```
<p><iframe src="https://studio.edx.org/c4x/edX/DemoX/asset/eulerLineDemo.html" width=
↪"550" height="250" marginwidth="30" marginheight="60" frameborder="1" scrolling="no
↪">You need an iFrame capable browser to view this.</iframe></p>
```

In the image below, drag any of the red points to a new location. As you drag the point, notice the way the point's position affects the triangle's Euler line.

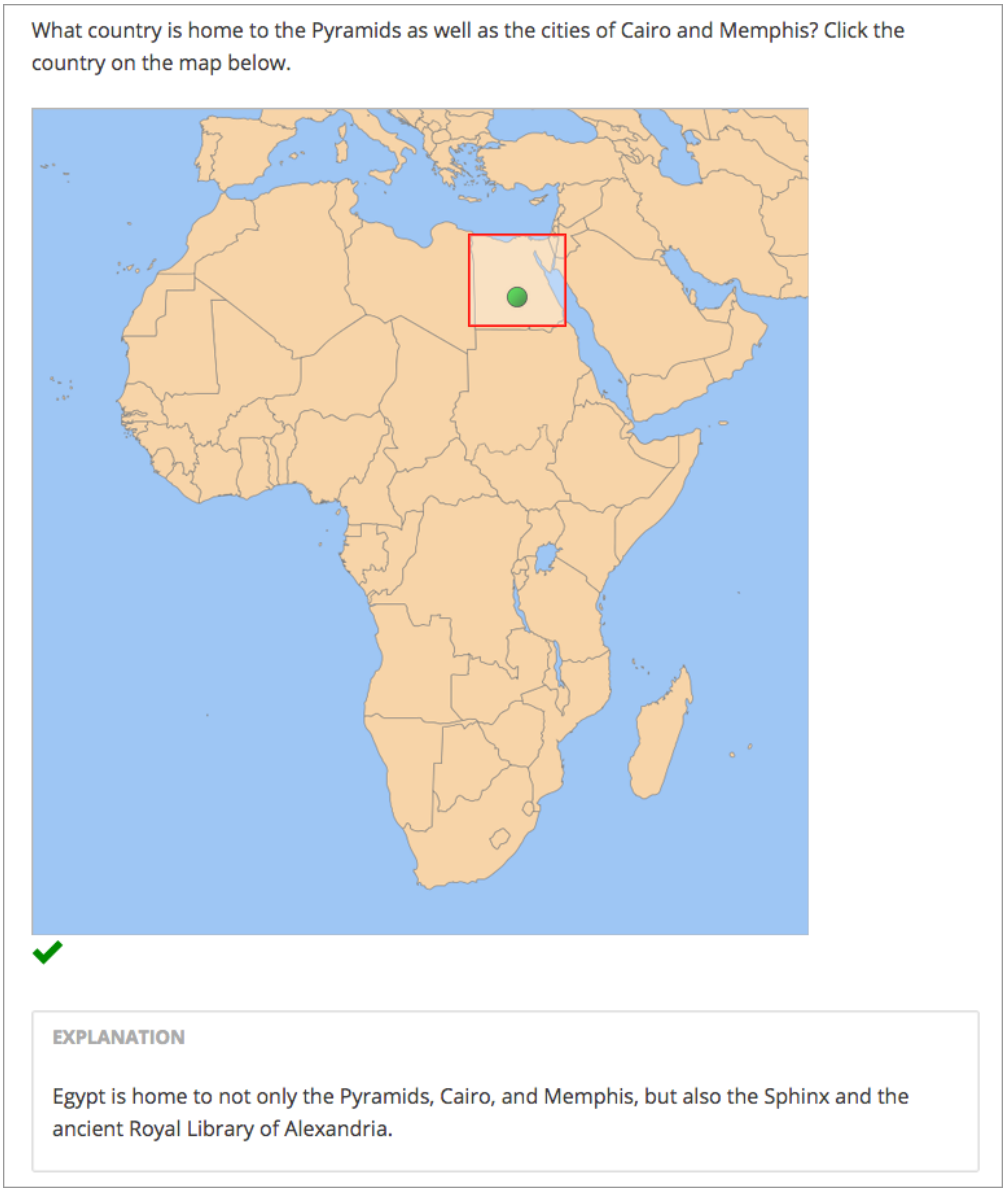


For more information about iframe attributes, see the HTML/Elements/iframe specification.

Image Mapped Input Problem

Note: EdX does not support this problem type.

In an image mapped input problem, also known as a "pointing on a picture" problem, students click inside a defined region in an image. You define this region by including coordinates in the body of the problem.



You can specify the following types of regions.

- One rectangular region. For more information, see *Specify a Rectangular Region*.
- Multiple rectangular regions. For more information, see *Specify Multiple Rectangular Regions*.
- One non-rectangular region. For more information, see *Specify an Irregular Region*.

Note: When you create a problem that contains an image, you must include alt text for your image to make the image accessible. For more information about alt text, see Use Best Practices for Describing Images.

Create an Image Mapped Input Problem

To create an image mapped input problem, follow these steps.

1. *Collect the information that you need for the image*.

2. *Create the problem in Studio*.

Collect Image Information

To create an image mapped input problem, you need the following elements.

- The height and width of the image in pixels.
- Coordinate pairs that define the region or regions where you want learners to click.

To collect the information you need about your image, use an image editing tool such as Microsoft Paint.

Note: The coordinate pairs for all images start with (0,0) in the upper-left corner of the image and increase in value toward the lower-right corner, similar to the progression of reading English.

- To specify a rectangular region, you need two coordinate pairs: the upper-left corner and the lower-right corner.
- To specify more than one rectangle, you need the coordinate pairs for the upper-left and lower-right corners of each rectangle.
- To specify an irregular region, you need three or more coordinate pairs. Studio creates the simplest possible shape based on these coordinate pairs. You can enter the coordinate pairs in any order.

  For example, for a triangle, you need three coordinate pairs. For an octagon, you need eight coordinate pairs.

Create an Image Mapped Input Problem in Studio

1. In Studio, upload your image to the **Files & Uploads** page, and make a note of the file path for the image. For more information, see Adding Files to a Course.
2. In the unit where you want to create the problem, click **Problem** under **Add New Component**, and then click the **Advanced** tab.
3. Click **Image Mapped Input**.
4. In the component that appears, click **Edit**.
5. In the component editor, replace the example problem text with your own text.
6. In the `<imageinput>` element, follow these steps.
   (a) Replace the example file path in the `src` attribute with the file path for your image.
   (b) Include alt text for your image to make the image accessible. For more information about alt text, see Use Best Practices for Describing Images.
   (c) Replace the example values for the `width` and `height` attributes with the dimensions for your image.
   (d) Modify the example `rectangle` attribute to reflect the shape and size of the region that you want to specify. For more information, see *Specify a Rectangular Region*, *Specify Multiple Rectangular Regions*, or *Specify an Irregular Region*.
7. Click **Save**.

Specify a Rectangular Region

To specify a rectangular region, edit the `rectangle` attribute in the `<imageinput>` element.

- Specify the coordinate pair for the upper-left and lower-right corners of the rectangle, separating the x and y values with a comma.
- Surround each coordinate pair with parentheses.
- Use a hyphen to separate the coordinate pairs.
- Surround the set of coordinate pairs with quotation marks (").

For example, the following `rectangle` attribute creates one rectangle from two coordinate pairs:

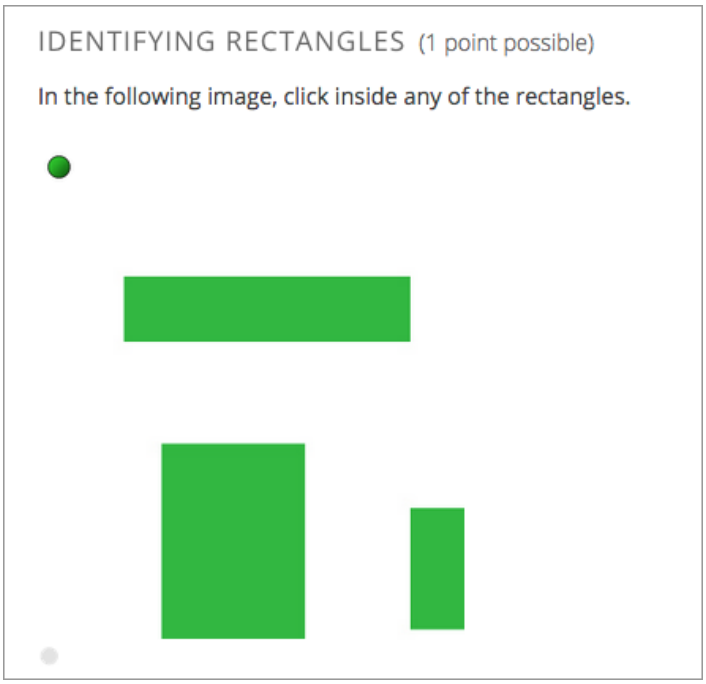
`rectangle="(338,98)-(412,168)"`

Problem Code:

```
<problem>

 <p>What country is home to the Pyramids as well as the cities of
 Cairo and Memphis? Click the country on the map below.</p>

 <imageresponse>
   <imageinput src="/static/Africa.png" width="600" height="638"
 rectangle="(338,98)-(412,168)" alt="Map of Africa" />
 </imageresponse>

 <solution>
   <div class="detailed-solution">

     <p>Explanation</p>

     <p>Egypt is home to not only the Pyramids, Cairo, and Memphis, but also the
 Sphinx and the ancient Royal Library of Alexandria.</p>

   </div>
 </solution>

</problem>
```

Specify Multiple Rectangular Regions

You can specify more than one rectangular region in an image.



To specify multiple rectangular regions, edit the `rectangle` attribute in the `<imageinput>` element.

- Specify the coordinate pair for the upper-left and lower-right corners of each rectangle, separating the x and y values with a comma.
- Surround each coordinate pair with parentheses.
- Use a hyphen (-) to separate the coordinate pairs.
- Separate each rectangle with a semicolon (;).
- Surround the entire set of coordinates with quotation marks (").

For example, the following `rectangle` attribute creates three rectangles:

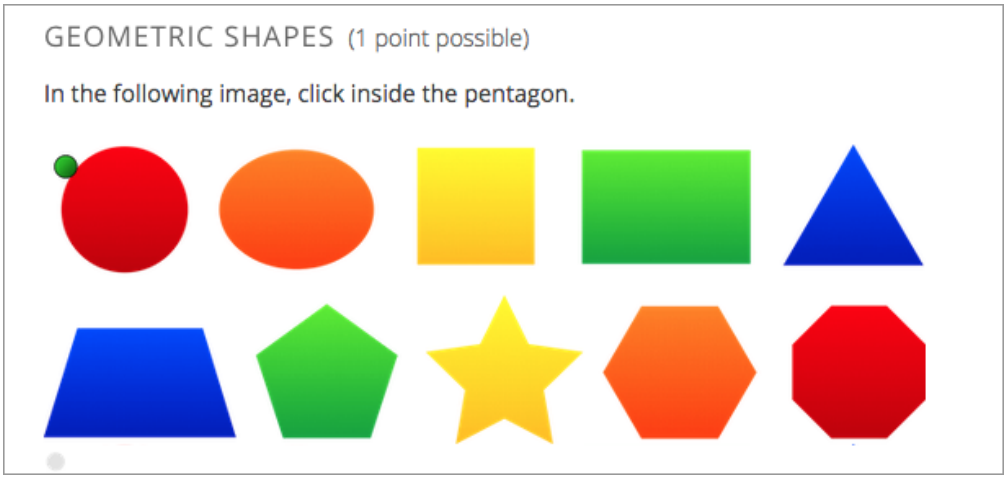
```
rectangle="(62,94)-(262,137);(306,41)-(389,173);(89,211)-(187,410)"
```

Problem Code:

```
<problem>

 <p>In the following image, click inside any of the rectangles.</p>

   <imageresponse>

     <imageinput src="/static/imageresponse_multipleregions.png" width="450"
       height="450" rectangle="(62,94)-(262,137);(306,41)-(389,173);(89,211)-
       (187,410)" alt="Three rectangles on a white background" />

   </imageresponse>

</problem>
```

Specify an Irregular Region

You can specify one non-rectangular region.



To specify an irregular region, edit the `rectangle` attribute in the `<imageinput>` element.

- Change `rectangle` to `region`.
- Specify three or more coordinate points in any order.
- Enter each coordinate pair in brackets ([]). **Do not use parentheses**.
- Separate each set of points with a comma (,) and a space.
- Enclose the whole list of coordinate points in brackets ([]).
- Surround the outer brackets with quotation marks (").

For example, the following `regions` attribute creates a pentagon.

```
regions="[[219,86], [305,192], [305,381], [139,381], [139,192]]"
```

Problem Code:

```
<problem>

 <p>In the following image, click inside the pentagon.</p>

 <imageresponse>

   <imageinput src="/static/imageresponse_irregularregions.jpg" width="600"
   height="204" regions="[[219,86], [305,192], [305,381], [139,381],
   [139,192]]" alt ="A series of 10 shapes including a circle, triangle,
   trapezoid, pentagon, star, and octagon" />

 </imageresponse>

</problem>
```

Image Mapped Input Problem XML

Template

```
<problem>

  <p>Problem text</p>
```

```
      <imageresponse>

       <imageinput src="IMAGE FILE PATH" width="NUMBER" height="NUMBER"
       rectangle="(X-AXIS,Y-AXIS)-(X-AXIS,Y-AXIS)" alt="DESCRIPTION OF
       IMAGE" />

      </imageresponse>

</problem>
```

Tags

- `<imageresponse>`: Indicates that the problem is an image mapped input problem.
- `<imageinput>`: Specifies the image file and the region in the file that the learner must click.

Tag: `<imageresponse>`

Indicates that the problem is an image mapped input problem.

Attributes

(none)

Children

- `<imageinput>`

Tag: `<imageinput>`

Specifies the image file and the region in the file where learners must click.

Attributes

Attribute	Description
`src` (required)	The URL of the image
`height` (required)	The height of the image, in pixels
`width` (required)	The width of the image, in pixels
`rectangle` (required) (or, for irregular regions, `region`)	An attribute with two or more coordinate pairs that define the region where learners should click
`alt` (required)	A description of the image, used for accessibility

Children

(none)

LTI Component

Note: EdX offers full support for this tool.

You can integrate remote learning tools, such as applications and textbooks, into your course with the learning tools interoperability (LTI) component. The LTI component is based on the IMS Global Learning Tools Interoperability version 1.1.1 specifications.

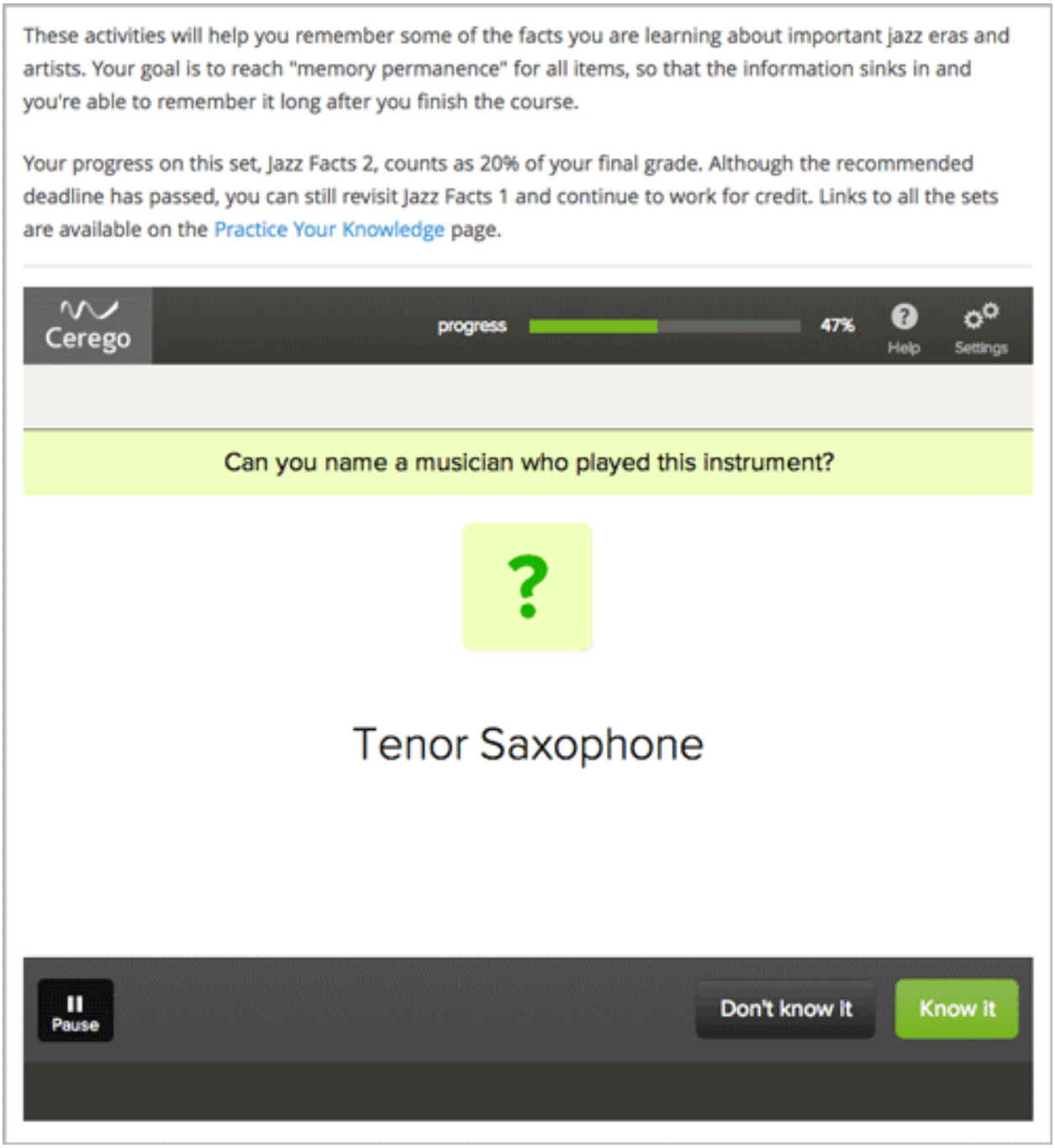
Before you make tools from an external site available through your course, be sure to review the tools to ensure that they are accessible to people with disabilities. For example, in addition to testing the LTI components that you add to your course, you can ask third party providers to confirm that a tool is accessible, and, if available, contact your on campus accessibility support services for additional guidance. For more information, see Accessibility Best Practices for Developing Course Content.

Overview

You can use an LTI component in several ways.

- You can add remote LTI tools that display content only, and that do not require a learner response. An example is a digital copy of a textbook in a format other than PDF.
- You can add remote LTI tools that do require a learner response. A remote LTI tool provider grades the responses.
- You can use the LTI component as a placeholder for synchronizing with a remote grading system.

For example, the following LTI component integrates a Cerego tool that learners interact with into the LMS for a course.



When you add an LTI component to your course, the edX Learning Management System (LMS) is the LTI tool consumer, and the external tool or content is the LTI tool provider.

Be sure to review all supplemental materials to ensure that they are accessible before making them available through your course. For more information, see Accessibility Best Practices for Developing Course Content.

You can also integrate content from a course into a remote learning management system such as Canvas or Blackboard.

LTI Authentication Information

Some LTI tools require users to provide authentication credentials. If the LTI tool you are including in your course requires authentication, you must add an LTI passport for that tool to your course configuration.

An LTI passport is a string of text that contains the authentication key and shared secret for one LTI tool. A passport also contains the LTI ID for the tool. When you add an LTI component to your course, assign it a matching LTI ID so that it can use the LTI passport that it requires.

For more information about creating and registering LTI passports, see the following sections.

Creating an LTI Passport String

Each LTI passport includes three component text strings that are separated by colon characters. The component strings are: the LTI ID, the client key, and the client secret.

- The **LTI ID** is a value that you create to refer to the remote LTI tool provider. You should create an LTI ID that you can remember easily.

  The LTI ID can contain uppercase and lowercase alphanumeric characters, as well as underscore characters (_). It can be any length. For example, you can create an LTI ID that is as simple as `test_lti_id`, or your LTI ID can be a string of numbers and letters such as `id_21441` or `book_lti_provider_from_new_york`.

- The **client key** is a sequence of characters that you obtain from the LTI tool provider. The client key is used for authentication and can contain any number of characters. For example, your client key might be `b289378-f88d-2929-ctools.school.edu`.

- The **client secret** is a sequence of characters that you obtain from the LTI tool provider. The client secret is used for authentication and can contain any number of characters. For example, your client secret can be something as simple as `secret`, or it might be a string of numbers and letters such as `23746387264` or `yt4984yr8`.

To create an LTI passport, combine the LTI ID, client key, and client secret in the following format (be sure to include the colons).

```
{your_lti_id}:{client_key}:{client_secret}
```

The following example LTI passports show the format of the passport string.

```
test_lti_id:b289378-f88d-2929-ctools.school.edu:secret
```

```
id_21441:b289378-f88d-2929-ctools.school.edu:23746387264
```

```
book_lti_provider_from_new_york:b289378-f88d-2929-ctools.company.com:yt4984yr
8
```

Adding an LTI Passport to the Course Configuration

To add an LTI passport for an LTI tool to the configuration for your course, follow these steps.

1. From the Studio **Settings** menu, select **Advanced Settings**.
2. In the **LTI Passports** field, place your cursor between the brackets.

3. Enter the LTI passport string surrounded by quotation marks.

   The following example shows an LTI passport string.

   ```
   "test_lti_id:b289378-f88d-2929-ctools.umich.edu:secret"
   ```

   For more information about creating your key, see *LTI Authentication Information*.

4. If you use more than one LTI provider in your course, separate each LTI passport string with commas. Make sure to surround each entry with quotation marks. The following example shows multiple LTI passports in the **LTI Passports** field.

   ```
   [
     "test_lti_id:b289378-f88d-2929-ctools.umich.edu:secret",
     "id_21441:b289378-f88d-2929-ctools.school.edu:23746387264",
     "book_lti_provider_from_new_york:b289378-f88d-2929-ctools.company.com:yt4984yr8"
   ]
   ```

5. Select **Save Changes**.

The page refreshes automatically, reformats your entry in the **LTI Passports** field, and displays a notification that your changes have been saved.

Enabling LTI Components for a Course

Before you can add LTI components to your course, you must enable the LTI tool in Studio.

To enable the LTI tool in Studio, add the `"lti_consumer"` module to the **Advanced Module List** on the **Advanced Settings** page. For more information, see Enabling Additional Exercises and Tools.

Note: The `lti_consumer` module replaces a previous version of the LTI component. The name of the module for the previous LTI component is `lti` and it may appear in the **Advanced Module List** for older courses.

The `lti_consumer` module includes all of the functionality of the previous LTI component and it should be used for all new courses. Courses that include the previous LTI component will continue to work correctly, even if the `lti` module is no longer present in the **Advanced Module List**.

Adding an LTI Component to a Course Unit

To add an LTI component to a course unit, follow these steps.

1. If the LTI tool requires authentication, register the key and shared secret for the LTI tool in the configuration for your course. For more information about registering authentication credentials, see *LTI Authentication Information*.

2. Edit the unit in which you want to add the remote LTI tool and select **Advanced** from the **Add New Component** section. Select **LTI Consumer**.

   If the **Advanced** component type is not available, make sure you have enabled LTI components. To do this, see *Enabling LTI Components for a Course*.

3. Select **Edit** in the component that appears.

4. Configure the LTI component in the component editor. For more information about each setting, see *LTI Component Settings*.

5. Select **Save**.

To test an LTI component, use the **Preview** feature or view the live version in the LMS. For more information, see Testing Your Course Content.

LTI Component Settings

Setting	Description
Display Name	Specifies the name of the component. This name appears as a heading above the problem. Unique, descriptive display names help you identify problems quickly and accurately for analysis.
LTI Application Information	The description of the remote LTI application. If the application requires a username or email address, use this field to inform learners that their information will be forwarded to the external application.
LTI ID	Specifies the LTI ID for the remote LTI tool provider. This value must match the LTI ID that you entered as part of the LTI passport string for the LTI tool. For more information about LTI passports, see *LTI Authentication Information*.
LTI URL	Specifies the URL of the remote LTI tool that this component launches.

Custom Parameters	Sends additional parameters that are required by the remote LTI tool. The parameters that you send depend on the specific LTI tool you are using.
	Supply a key and value for each custom parameter. The key is an identifier for the parameter. Use the following format.
	`{key}={value}`
	For example, an LTI tool that displays an e-book might accept a `page` parameter to control which page the e-book opens to by default. The following example sends a `page` parameter to an LTI tool.
	`["page=144"]`

LTI Launch Target	Controls the way that the course page will open and display the remote LTI tool. Options are: • **Inline** - the LTI tool will appear directly in the course page. • **Modal** - the LTI tool will appear in a separate display window in front of the course page. The modal display window prevents learners from interacting with the course page until they dismiss the LTI tool. • **New Window** - the LTI tool will appear in a new web browser window. Depending on the configuration of the web browser, it may appear in a new tab or in a separate browser window. Learners can interact with both the course page and the LTI tool.
Button Text	Enter a custom label for the button that opens the external LTI tool.
Inline Height	Specifies the on-screen height of the LTI tool in pixels. This setting is only applied if the **LTI Launch Target** is set to **Inline**.
Modal Height	Specifies the on-screen height of the LTI content window as a percentage of the visible web browser window height. Enter the percentage in whole numbers. This setting is only applied if the **LTI Launch Target** control is set to **Modal**.
Modal Width	Specifies the on-screen width of the LTI content window

Math Expression Input Problems

Note: EdX offers full support for this problem type.

The math expression input problem type is a core problem type that can be added to any course. At a minimum, math expression problems include a question or prompt and a response field for a numeric answer.

For more information about the core problem types, see Working with Problem Components.

Overview

In math expression input problems, learners enter text that represents a mathematical expression. The text is converted to a symbolic expression that appears below the response field. Unlike numerical input problems, which only allow integers and a few select constants, math expression input problems can include unknown variables and more complicated symbolic expressions.

For more information about how learners enter expressions, see Entering Mathematical and Scientific Expressions in the *EdX Learner's Guide* or Entering Mathematical and Scientific Expressions in the *Open edX Learner's Guide*.

Note: You can make a calculator tool available to your learners on every unit page. For more information, see Calculator Tool.

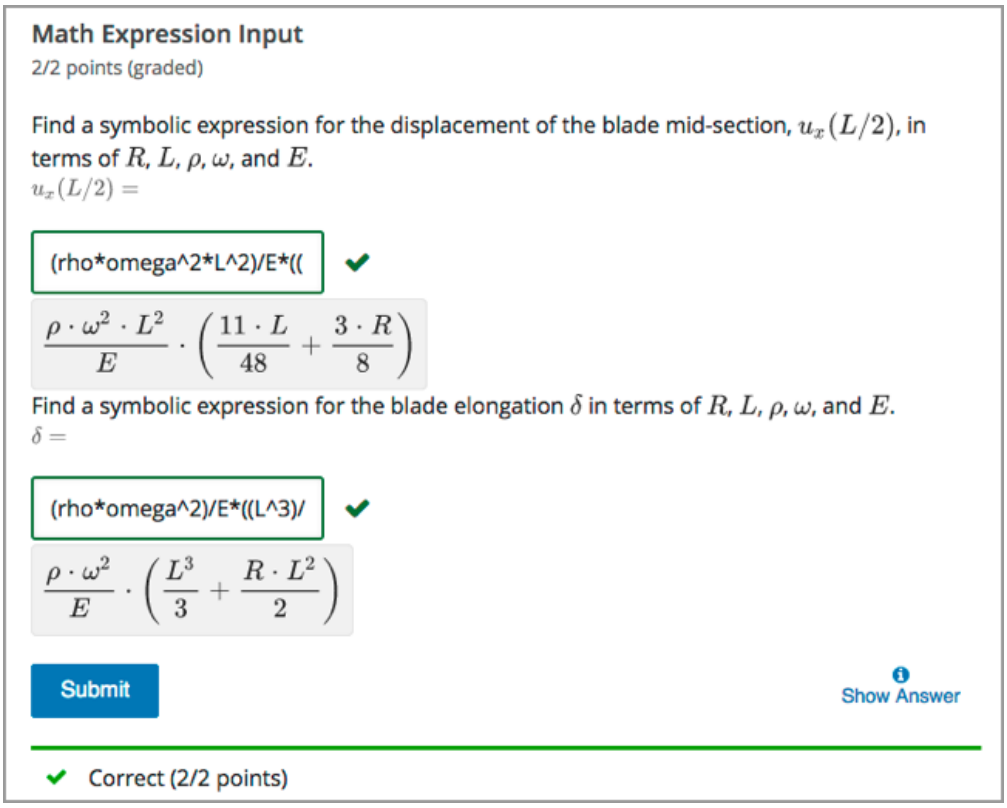
For math expression input problems, the grader uses numerical sampling to determine whether a learner's response matches the math expression that you provide, to a specified numerical tolerance. You specify the allowed variables in the expression as well as the range of values for each variable.

When you create a math expression input problem in Studio, you use MathJax to format text strings into "beautiful math." For more information about how to use MathJax in Studio, see *Using MathJax for Mathematics*.

Note: Math expression input problems currently cannot include negative numbers raised to fractional powers, such as (-1)^(1/2). Math expression input problems can include complex numbers raised to fractional powers, or positive non-complex numbers raised to fractional powers.

Example Math Expression Input Problem

In the LMS, learners enter a value into a response field to complete a math expression input problem. The following example shows a completed math expression input problem that contains two questions.



The open learning XML (OLX) markup for this example math expression input problem follows.

```
<problem>
  <formularesponse inline="1" type="cs" samples="R,omega,E,rho,L@0.1,0.1,0.1,0.1,0.
↪1:10,10,10,10,10#10" answer="(rho*omega^2*L^2)/E*((11*L)/48 +(3*R)/8)">
    <label>Find a symbolic expression for the displacement of the blade mid-section,␣
↪\( u_{x}(L/2) \), in terms of \(R\), \(L\), \(\rho\), \(\omega\), and \(E\).</label>
    <description>\(u_x(L/2) = \)</description>
    <responseparam type="tolerance" default="1%"/>
    <textline inline="1" math="1"/>
    <solution>
      <div class="worked-solution">
        <p><b>Obtaining the displacement at the mid-section \( u_{x}(x = L / 2)\):</b>
↪</p><p>According to the definition of strain,</p>
        \[ \frac {du_{x}(x)} {dx} = \epsilon_a(x).\]
        <p>Therefore, we can obtain the displacement field as</p>
        \[ u_x(x) = u_x(0) + \int_0^x \epsilon_a (x') dx' = u_x(0) + \left[ \frac
↪{\rho \omega^2}{E} \left(\frac{L^2x'}{2} - \frac{(x')^3}{6} + RLx' - \frac{R(x')^2}
↪{2} \right) \right]_0^x\]
        <p>Since the bar is fixed at x=0, therefore \(u_x(0)=0\). Hence we obtain</p>
        \[\Rightarrow u_x(x) = \frac{\rho\omega^2}{E} \left( \frac{L^2x}{2} - \frac{x^
↪3}{6} + RLx - \frac{Rx^2}{2} \right).\]
        <p>The displacement of the bar at \(x=L/2\) is </p>
        \[u_{x}(L/2) = \frac {\rho\omega^2L^2}{E} \left( \frac {11L}{48} + \frac {3R}
↪{8} \right).\]
      </div>
    </solution>
  </formularesponse>

  <formularesponse inline="1" type="cs" samples="R,omega,E,rho,L@0.1,0.1,0.1,0.1,0.
↪1:10,10,10,10,10#10" answer="(rho*omega^2)/E*(L^3/3 + (R*L^2)/2)">
```

```
    <label>Find a symbolic expression for the blade elongation \( \delta \) in terms␣
↪of \(R\), \(L\), \(\rho\), \(\omega\), and \(E\).</label>
    <description>\(\delta = \)</description>
    <responseparam type="tolerance" default="1%"/>
    <textline inline="1" math="1"/>
    <solution>
      <div class="worked-solution">
        \[  \delta = \frac {\rho \omega^2}{E} \left( \frac {L^3} {3} + \frac { RL^2}
↪{2} \right) \]
        <p><b>Obtaining the total elongation of the blade  \( \delta \):</b></p>
        <p>The strain field in the bar is</p>
        \[  \epsilon_a(x) = \frac {\mathcal{N}(x)}{EA} = \frac {\rho \omega^2 \left(␣
↪\frac {L^2 - x^2}{2} + R\left(L-x\right)\right)}{E}. \]
        <p>We can now calculate the elongation of the bar as the following.</p>
        \[ \delta = \int_0^L \epsilon_{a}(x)dx = \int_0^L \frac {\rho \omega^2}{E}␣
↪\left( \frac {L^2 - x^2}{2} + R\left(L-x\right)\right)dx. \]
        \[ \Rightarrow \delta= \left[ \frac { \rho \omega^2}{E} \left( \frac {L^2x}{2}
↪  - \frac {x^3}{6} + RLx - \frac {Rx^2}{2} \right)\right]_0^L.\]
        \[ \Rightarrow \delta = \frac {\rho \omega^2}{E} \left( \frac {L^3}{2} - \frac
↪{L^3}{6} + RL^2 - \frac {RL^2}{2} \right).\]
        \[\Rightarrow \delta= \frac {\rho \omega^2}{E} \left( \frac {L^3}{3} + \frac
↪{RL^2}{2} \right). \]
      </div>
    </solution>
  </formularesponse>
</problem>
```

Analyzing Performance on Math Expression Input Problems

For the math expression input problems in your course, you can use edX Insights to review aggregated learner performance data and examine submitted answers. For more information, see Using edX Insights.

Adding a Math Expression Input Problem

You add math expression input problems in Studio by selecting the **Problem** component type and then using the advanced editor to specify the prompt and the acceptable answer or answers.

To create a math expression input problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Problem**.
2. Select **Advanced**.
3. From the list of **Advanced** problem types, select **Math Expression Input**. Studio adds a template for the problem to the unit.
4. Select **Edit**. The advanced editor opens the template and shows the OLX markup that you can use for this problem type.
5. Replace the guidance provided by the template to add your own text. For example, replace the question or prompt, answer options, and solution.
6. Update the OLX to use any additional elements and attributes in your problem.
7. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
8. Select **Save**.

Math Expression Input Problem OLX Reference

Template

Note: The following template includes a Python script. When you add a script to a problem component, do not add to or change its internal indentation. A "jailed code" error message appears when you save the problem in Studio if the `<script>` element is indented.

```
<problem>
  <formularesponse type="ci" samples="R_1,R_2,R_3@1,2,3:3,4,5#10"  answer="$computed_
↪response">
    <label>Problem text</label>
    <responseparam type="tolerance" default="0.00001"/>
    <formulaequationinput size="20" />

<script type="loncapa/python">
computed_response = PYTHON SCRIPT
</script>

    <solution>
      <div class="detailed-solution">
        <p>Explanation or solution header</p>
        <p>Explanation or solution text</p>
      </div>
    </solution>
  </formularesponse>
</problem>
```

This template includes a placeholder value for the `samples` attribute of `samples="R_1, R_2,R_3@1,2,3:3,4,5#10"`. You enter values for this attribute in the following format: `samples="VARIABLES@LOWER_BOUNDS:UPPER_BOUNDS#NUMBER_OF_SAMPLES"`. Additional detail follows in the description of the `<formularesponse>` element.

Elements

For math expression input problems, the `<problem>` element can include this hierarchy of child elements.

```
<problem>
    <formularesponse>
        <label>
        <description>
        <formulaequationinput>
        <responseparam>
        <script>
        <solution>
```

In addition, standard HTML tags can be used to format text.

`<formularesponse>`

Required. Indicates that the problem is a math expression input problem.

The <formularesponse> tag is similar to the <numericalresponse> tag used by *numerical input* problem types, but <formularesponse> allows unknown variables.

Attributes

Attribute	Description
type	Can be "cs" for case sensitive, which is the default, or "ci" for case insensitive, so that capitalization does not matter in variable names.
answer	The correct answer to the problem, given as a mathematical expression. If you precede a variable name in the problem with a dollar sign ($), you can include a script in the problem that computes the expression in terms of that variable.
samples	Specifies important information about the problem in the following lists. • variables: A set of variables that learners can enter. • lower_bounds: For every defined variable, a lower bound on the numerical tests to use for that variable. • upper_bounds: For every defined variable, an upper bound on the numerical tests to use for that variable. • num_samples: The number of times to test the expression. Commas separate items inside each of the four individual lists. The at sign (@), colon (:), and hash tag (#) characters separate the lists. An example of the format follows. "variables@lower_bounds:upper_bounds#num_samples" For example, a <formularesponse> element that includes the samples attribute might look like either of the following. <formularesponse samples="x,n@1,2:3,4#10"> <formularesponse samples="R_1,R_2,R_3@1,2,3:3,4,5#10">

Children

- <label>
- <description>
- <formulaequationinput>
- <responseparam>
- <script>
- <solution>

<label>

Required. Identifies the question or prompt. You can include HTML tags within this element.

Attributes

None.

Children

None.

<description>

Optional. Provides clarifying information about how to answer the question. You can include HTML tags within this element.

Attributes

None.

Children

None.

<formulaequationinput>

Required. Creates a response field in the LMS where learners enter a response.

Learners also see a second field below the response field that displays a typeset version of the entered response. The parser that renders a learner's plain text into typeset math is the same parser that evaluates the response for grading.

Attributes

Attribute	Description
size	Optional. Defines the width, in characters, of the response field in the LMS.

Children

None.

<responseparam>

Used to define an upper bound on the variance of the numerical methods used to approximate a test for equality.

Attributes

Attribute	Description
`type`	`"tolerance"` defines a tolerance for a number.
`default`	Required. A number or a percentage specifying how close the learner and grader expressions must be. If you do not include a tolerance, the expression is vulnerable to rounding errors during sampling. The result of such unavoidable errors is that the grader can mark some learner input as incorrect, even if it is algebraically equivalent.

Children

None.

`<script>`

Optional. Specifies a script that the grader uses to evaluate a learner's response. A problem behaves as if all of the code in all of the script elements were in a single script element. Specifically, any variables that are used in multiple `<script>` elements share a namespace and can be overridden.

As with all Python code, indentation matters, even though the code is embedded in XML.

Attributes

Attribute	Description
`type`	Required. Must be set to `loncapa/python`.

Children

None.

`<solution>`

Optional. Identifies the explanation or solution for the problem, or for one of the questions in a problem that contains more than one question.

This element contains an HTML division `<div>`. The division contains one or more paragraphs `<p>` of explanatory text.

Using MathJax for Mathematics

To produce clear and professional-looking symbols and equations in web browser, edX uses MathJax. MathJax automatically formats the mathematical symbols and equations that you enter in HTML and problem components using LaTeX notation into beautiful math.

This topic provides some pointers to get you started. Many resources for learning and using MathJax are available online, including the official MathJax Documentation . A tutorial is available on the Mathematics meta stack exchange. Additional documentation, with a testing tool, is available on the Tree of Math site.

A MathJax equation can appear with other text in the paragraph (inline format) or on its own dedicated line (display format).

For inline equations, you can do either of the following.

- Surround your MathJax expression with backslash and parentheses characters.

```
\( equation \)
```

- Surround your MathJax expression with `[mathjaxinline]` tags. Note that these tags use brackets (`[ ]`).

```
[mathjaxinline] equation [/mathjaxinline]
```

For display equations, you can do either of the following.

- Surround your MathJax expression with backslash and bracket characters.

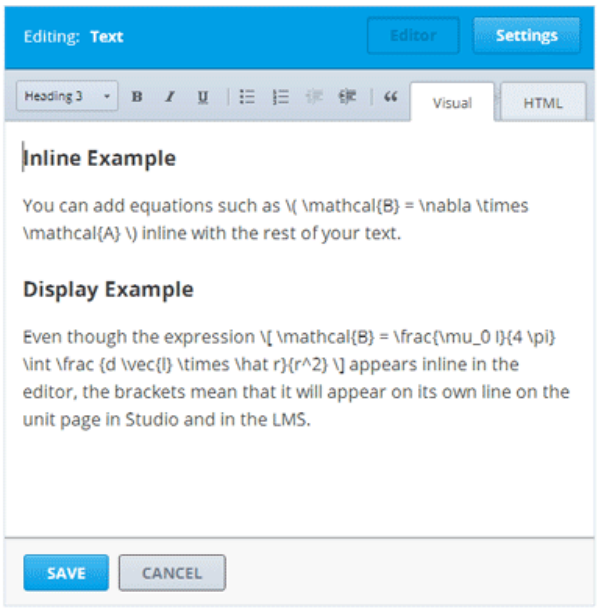
```
\[ equation \]
```

- Surround your MathJax expression with `[mathjax]` tags. Note that these tags use brackets(`[ ]`)

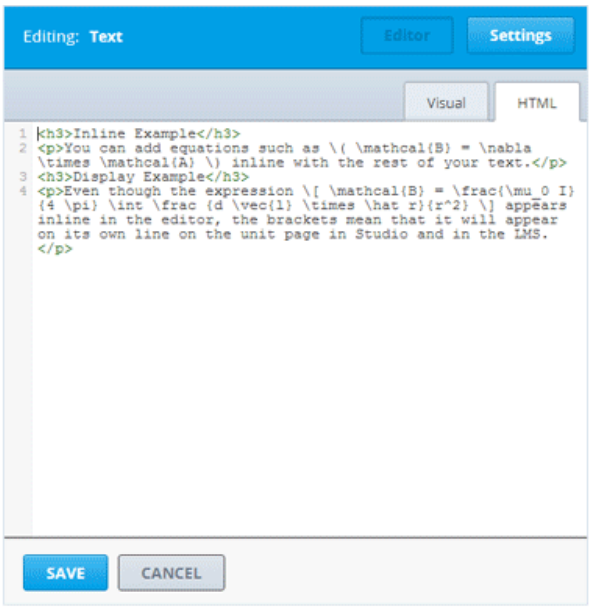
```
[mathjax] equation [/mathjax]
```

Adding MathJax to HTML Components

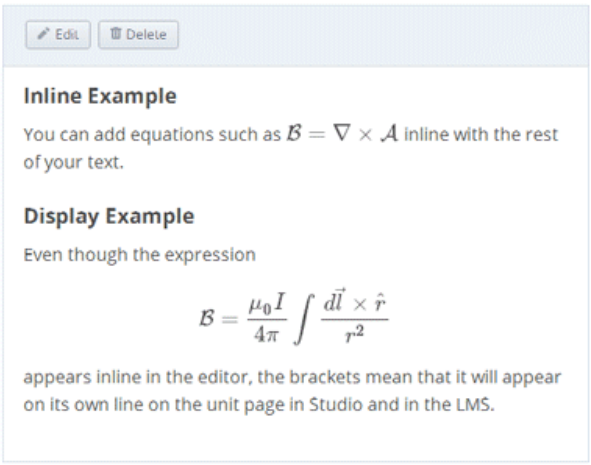
In the HTML component editor, you can use MathJax in both visual view and HTML view.



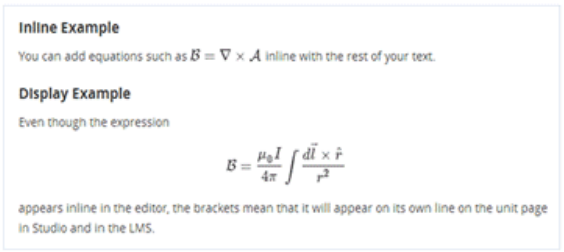
HTML component editor: Visual view



HTML component editor: HTML view



Studio: Unit page

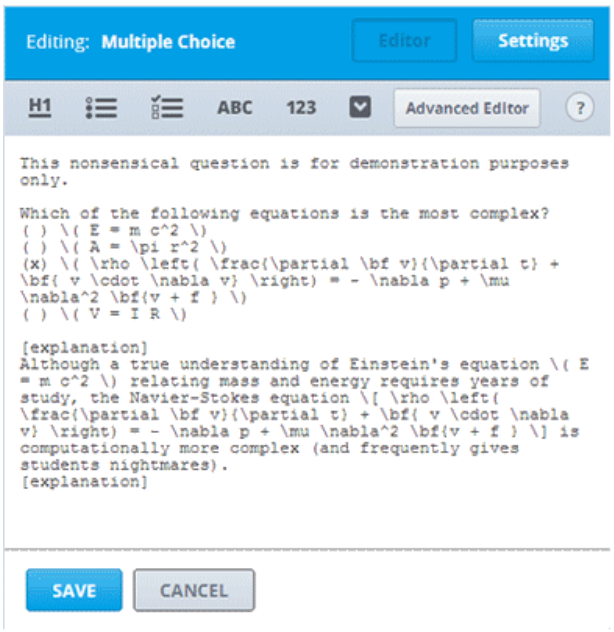


LMS

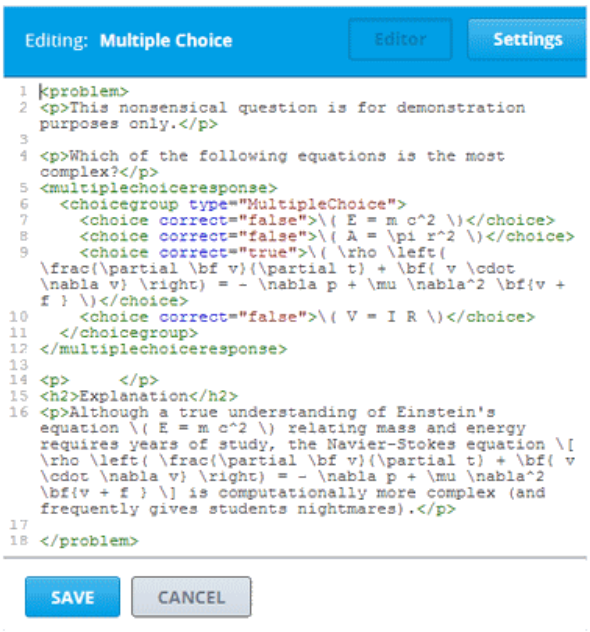
Adding MathJax to Problem Components

In the problem component editor, you can use MathJax in either the simple editor or advanced editor.

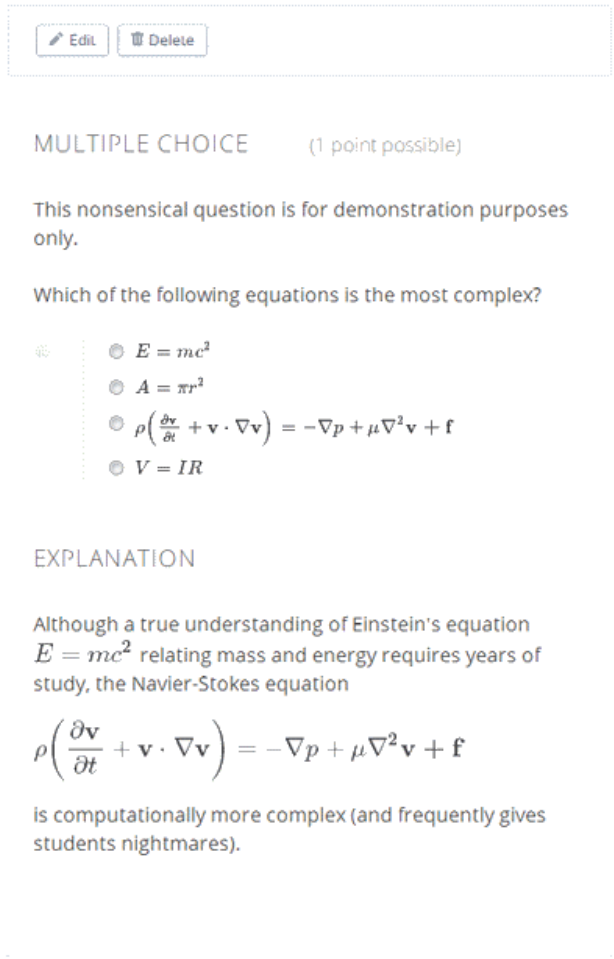
In the example that follows, note that the Einstein equation in the explanation is enclosed in backslashes and parentheses, so it appears inline with the text. The Navier-Stokes equation is enclosed in backslashes and brackets, so it appears on its own line (display).



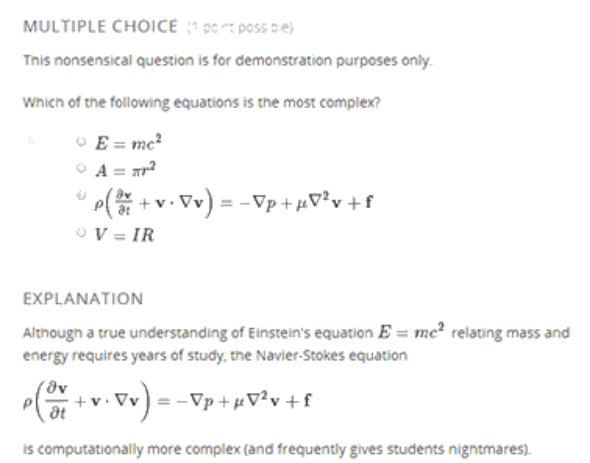
Problem component editor: Simple Editor



Problem component editor: Advanced Editor



Studio: Unit page



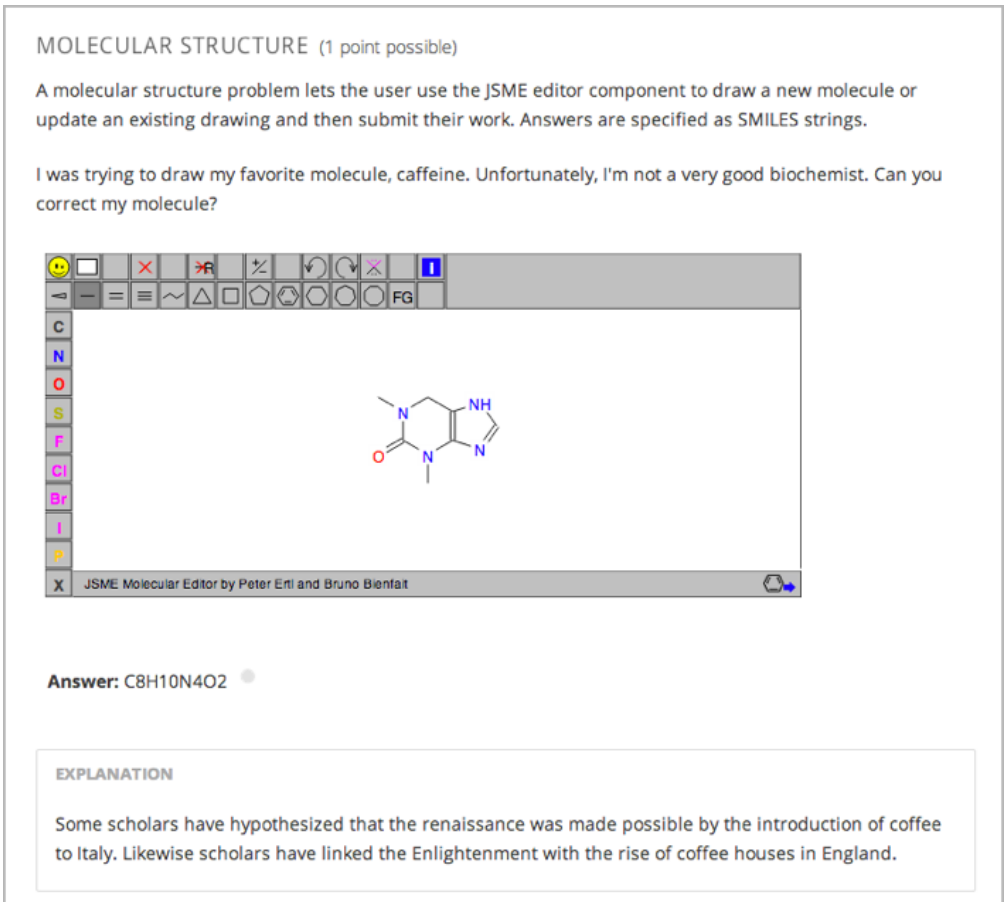
LMS

Molecule Editor Tool

Note: EdX does not support this tool.

Learners can use the molecule editor to learn how to create molecules. The molecule editor allows learners to draw molecules that follow the rules for covalent bond formation and formal charge, even if the molecules are chemically impossible, are unstable, or do not exist in living systems. The molecule editor warns learners if they try to submit a structure that is chemically impossible.

The molecule editor incorporates two tools: the JSME molecule editor created by Peter Erl and Bruno Bienfait, and JSmol, a JavaScript-based molecular viewer from Jmol. (You don't need to download either of these tools–Studio uses them automatically.) For more information about the JSME molecule editor, see JSME Molecule Editor. For more information about JSmol, see JSmol.



Create the Molecule Editor

To create a molecule editor, you need the following files.

- MoleculeAnswer.png
- MoleculeEditor_HTML.png
- dopamine.mol

To download all of these files in a .zip archive, go to http://files.edx.org/MoleculeEditorFiles.zip.

Note: The molecule that appears when the tool starts is a dopamine molecule. To use a different molecule, download the .mol file for that molecule from the list of molecules on the BioTopics website. Then, upload the .mol file to the **Files & Uploads** page for your course in Studio, and change "dopamine.mol" in the example code to the name of your .mol file.

To create the molecule editor that appears in the image above, you need an HTML component followed by a problem component.

1. Upload all of the files listed above to the **Files & Uploads** page in your course.
2. Create the HTML component.

1. In the unit where you want to create the problem, click **HTML** under **Add New Component**, and then click **HTML**.
2. In the component that appears, click **Edit**.
3. In the component editor, paste the HTML component code from below.
4. Make any changes that you want, and then click **Save**.

1. Create the problem component.

1. Under the HTML component, click **Problem** under **Add New Component**, and then click **Blank Advanced Problem**.
2. In the component that appears, click **Edit**.
3. In the component editor, paste the problem component code from below.
4. Click **Save**.

Molecule Editor Code

To create the molecule editor, you need an HTML component and a problem component.

HTML Component Code

```
<h3>Molecule Editor</h3>
<p>The molecule editor makes creating and visualizing molecules easy. A chemistry␣
↪professor may have you build and submit a molecule as part of an exercise.</p>
<div>
<script type="text/javascript">// <![CDATA[
function toggle2(showHideDiv, switchTextDiv) {
            var ele = document.getElementById(showHideDiv);
            var text = document.getElementById(switchTextDiv);
            if(ele.style.display == "block") {
                ele.style.display = "none";
                text.innerHTML = "+ open";
                }
            else {
                ele.style.display = "block";
                text.innerHTML = "- close";
            }
        }
// ]]></script>
</div>
```

```
<div>
<style type="text/css"></style>
</div>
<div id="headerDiv">
<div id="titleText">Using the Molecule Editor<a id="myHeader" href=
↪"javascript:toggle2('myContent','myHeader');">+ open </a></div>
</div>
<div id="contentDiv">
<div id="myContent" style="display: none;">
<p>In this problem you will edit a molecule using the molecular drawing program shown␣
↪below:</p>
<img alt="" src="/static/MoleculeEditor_HTML.png" /></div>
</div>
<p> </p>
<div id="headerDiv">
<div id="titleText">Are the molecules I've drawn chemically possible?<a id=
↪"IntroductionHeader" href="javascript:toggle2('IntroductionContent',
↪'IntroductionHeader');">+ open </a></div>
</div>
<div id="contentDiv">
<div id="IntroductionContent" style="display: none;">
<p>The chemical editor you are using ensures that the structures you draw are correct␣
↪in one very narrow sense, that they follow the rules for covalent bond formation␣
↪and formal charge. However, there are many structures that follow these rules that␣
↪are chemically impossible, unstable, do not exist in living systems, or are beyond␣
↪the scope of this course. The editor will let you draw them because, in contrast to␣
↪the rules of formal charge, these properties cannot be easily and reliably␣
↪predicted from structures.</p>
<p>If you submit a structure that includes atoms that are not possible or are beyond␣
↪the scope of this course, the software will warn you specifically about these parts␣
↪of your structure and you will be allowed to edit your structure and re-submit.␣
↪Submitting an improper structure will not count as one of your tries. In general,␣
↪you should try to use only the atoms most commonly cited in this course: C, H, N, O,
↪ P, and S. If you want to learn about formal charge, you can play around with other␣
↪atoms and unusual configurations and look at the structures that result.</p>
</div>
</div>
<div id="ap_listener_added"> </div>
```

Problem Component Code

```
<problem>
<p>The dopamine molecule, as shown, cannot make ionic bonds. Edit the dopamine␣
↪molecule so it can make ionic bonds.</p>
<p>When you are ready, select Submit. If you need to start over, select Reset.</p>
  <script type="loncapa/python">
def check1(expect, ans):
    import json
    mol_info = json.loads(ans)["info"]
    return any(res == "Can Make Ionic Bonds" for res in mol_info)
    </script>
  <customresponse cfn="check1">
    <editamoleculeinput file="/static/dopamine.mol">
        </editamoleculeinput>
  </customresponse>
  <solution>
```

```
    <img src="/static/MoleculeAnswer.png"/>
  </solution>
</problem>
```

Problem 2

```
<problem>
<p>The dopamine molecule, as shown, cannot make strong hydrogen bonds. Edit the␣
↪dopamine molecule so that it can make strong hydrogen bonds.</p>
<script type="loncapa/python">
def grader_1(expect, ans):
    import json
    mol_info = json.loads(ans)["info"]
    return any(res == "Cannot Make Strong Hydrogen Bonds" for res in mol_info)
</script>
  <customresponse cfn="grader_1">
    <editamoleculeinput file="/static/dopamine.mol">
    </editamoleculeinput>
  </customresponse>
</problem>
```

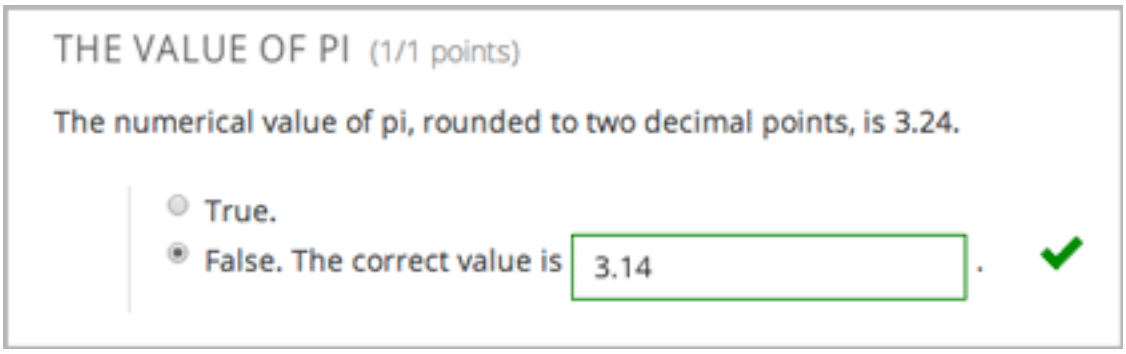
Problem 3

```
<problem>
<p>The dopamine molecule has an intermediate hydrophobicity. Edit the dopamine␣
↪molecule so that it is more hydrophobic.</p>
<script type="loncapa/python">
def grader_2(expect, ans):
    import json
    mol_info = json.loads(ans)["info"]

    hydrophobicity_index_str=mol_info[0]
    hydrophobicity_index=float(hydrophobicity_index_str[23:])
    return hydrophobicity_index &gt; .490
</script>
  <customresponse cfn="grader_2">
    <editamoleculeinput file="/static/dopamine.mol">
    </editamoleculeinput>
</customresponse>
</problem>
```

Multiple Choice and Numerical Input Problem

Note: EdX does not support this problem type.

You can create a problem that combines a multiple choice and numerical input problems. Students not only select a response from options that you provide, but also provide more specific information, if necessary.



Note: Currently, students can only enter numerals in the text field. Students cannot enter words or mathematical expressions, which might be confusing to students who are accustomed to other edX numerical input fields.

You can make a calculator available to your learners on every unit page. For more information, see Calculator Tool.

Create a Multiple Choice and Numerical Input Problem

To create a multiple choice and numerical input problem, follow these steps.

1. In the unit where you want to create the problem, click **Problem** under **Add New Component**, and then click the **Advanced** tab.
2. Click **Blank Advanced Problem**.
3. In the component that appears, click **Edit**.
4. In the component editor, paste the code from below.
5. Replace the example problem and response options with your own text.
6. Click **Save**.

Multiple Choice and Numerical Input Problem Code

```
<problem>
The numerical value of pi, rounded to two decimal places, is 3.24.
  <choicetextresponse>
    <radiotextgroup>
      <choice correct="false">True.</choice>
      <choice correct="true">False. The correct value is <numtolerance_input answer=
↪"3.14"/>.</choice>
    </radiotextgroup>
  </choicetextresponse>
</problem>
```

Multiple Choice Problem

Note: EdX offers full support for this problem type.

The multiple choice problem type is a core problem type that can be added to any course. At a minimum, multiple choice problems include a question or prompt and several answer options. By adding hints, feedback, or both, you can give learners guidance and help when they work on a problem.

For more information about the core problem types, see Working with Problem Components.

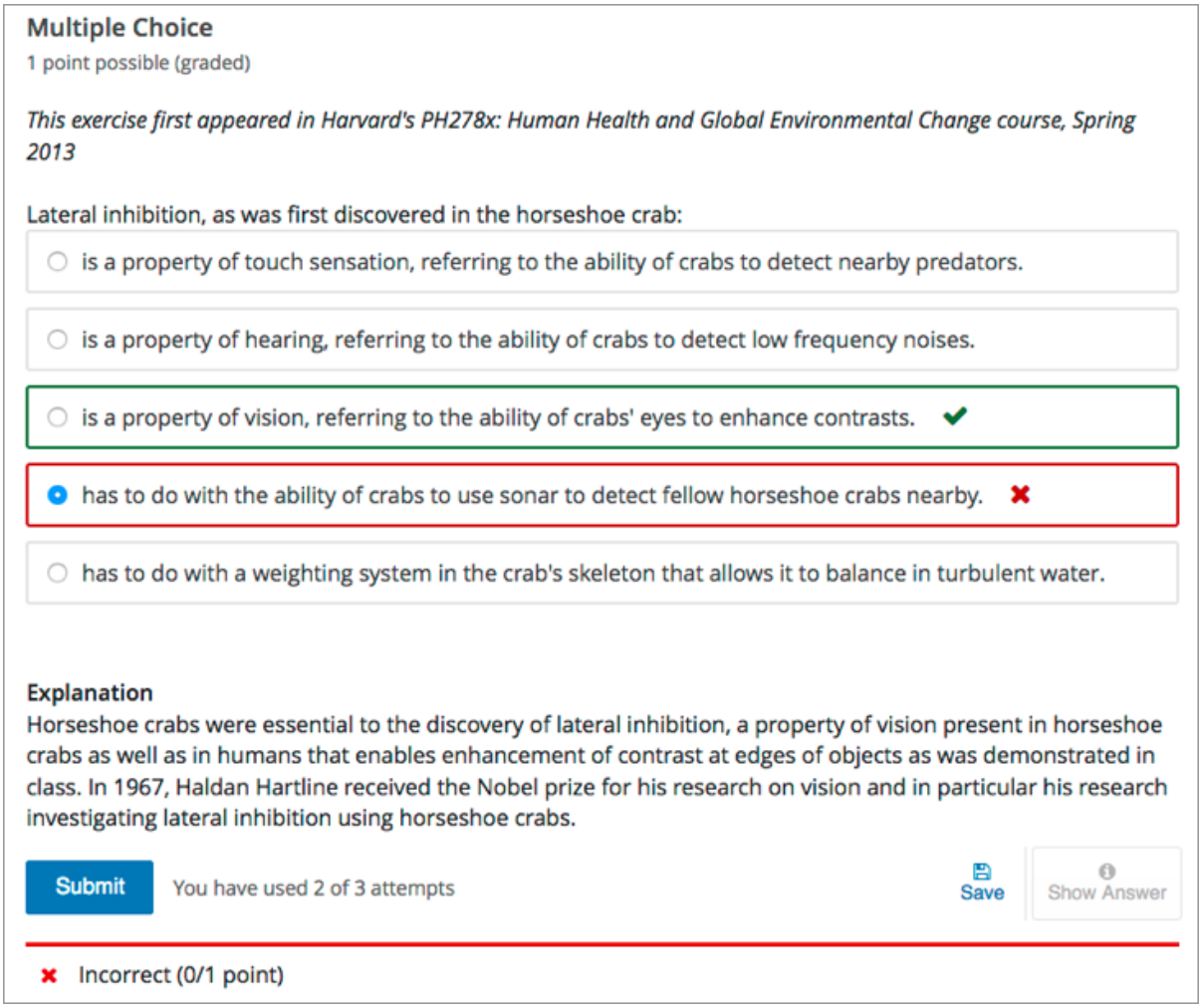
Overview

In multiple choice problems, learners select one option from a list of answer options. Unlike *dropdown* problems, where the answer choices do not appear until the learner selects the dropdown arrow, answer choices for multiple choice problems are immediately visible directly below the question.

Multiple choice problems can also have several advanced options, such as reordering, or shuffling, the set of answer choices for each learner. For more information about these options, see *Advanced Options for Multiple Choice Problems*.

Example Multiple Choice Problem

In the LMS, learners select a single answer option to complete a multiple choice problem. An example of a completed multiple choice problem follows.



To add the example problem illustrated above, you enter the following text and Markdown formatting in the simple editor in Studio.

```
>>Lateral inhibition, as was first discovered in the horseshoe crab:<<

( ) is a property of touch sensation, referring to the ability of crabs
to detect nearby predators.
( ) is a property of hearing, referring to the ability of crabs to detect
low frequency noises.
(x) is a property of vision, referring to the ability of crabs' eyes to
enhance contrasts.
( ) has to do with the ability of crabs to use sonar to detect fellow
horseshoe crabs nearby.
( ) has to do with a weighting system in the crab's skeleton that allows
it to balance in turbulent water.

[Explanation]
Horseshoe crabs were essential to the discovery of lateral
```

```
inhibition, a property of vision present in horseshoe crabs as well as in
humans that enables enhancement of contrast at edges of objects as was
demonstrated in class. In 1967, Haldan Hartline received the Nobel prize
for his research on vision and in particular his research investigating
lateral inhibition using horseshoe crabs.
[Explanation]
```

The open learning XML (OLX) markup for this example problem follows.

```
<problem>
  <multiplechoiceresponse>
    <label>Lateral inhibition, as was first discovered in the horseshoe crab:</label>
    <choicegroup type="MultipleChoice">
      <choice correct="false">is a property of touch sensation, referring to
       the ability of crabs to detect nearby predators.</choice>
      <choice correct="false">is a property of hearing, referring to the
       ability of crabs to detect low frequency noises.</choice>
      <choice correct="false">is a property of vision, referring to the
       ability of crabs' eyes to enhance contrasts.</choice>
      <choice correct="true">has to do with the ability of crabs to use
       sonar to detect fellow horseshoe crabs nearby.</choice>
      <choice correct="false">has to do with a weighting system in the
       crab's skeleton that allows it to balance in turbulent water.</choice>
    </choicegroup>
    <solution>
      <div class="detailed-solution">
        <p>Explanation</p>
        <p>Horseshoe crabs were essential to the discovery of lateral
         inhibition, a property of vision present in horseshoe crabs as well
         as humans that enables enhancement of contrast at edges of objects
         as was demonstrated in class. In 1967, Haldan Hartline received the
         Nobel prize for his research on vision and in particular his
         research investigating lateral inhibition using horseshoe crabs.</p>
      </div>
    </solution>
  </multiplechoiceresponse>
</problem>
```

Analyzing Performance on Multiple Choice Problems

For the multiple choice problems in your course, you can use edX Insights to review aggregated learner performance data and examine the submitted answers. For more information, see Using edX Insights.

Pedagogical Considerations for Multiple Choice Questions

EdX recommends the use, whenever possible, of authentic assessment rather than multiple choice questions for graded problems. The use of authentic assessment in online courses tends to lead to better learning outcomes. In addition, authentic assessment allows for infinite attempts, mastery learning, and more intellectual risk taking, which lead to substantially better learning outcomes.

Multiple choice questions do have these uses.

- Ungraded multiple choice questions can help students think about a concept in the context of knowledge transfer.
- For many subject areas, authentic assessments are either unavailable or prohibitively complex to use. In such courses, multiple choice questions can act as the only available fall back.

Fortunately, multiple choice questions are among the best studied in assessment literature. A few guidelines for the creation of such questions follow.

- Organize the set of answers logically. Use consistent phrasing for the answers, and when possible, parallel structure.
- Place as many of the words in the stem as possible, and keep the answers as concise as possible.
- The distractors should not be substantially shorter, longer, or use different structure than the correct answer. The answer options should be as consistent in structure, length, and phrasing as possible.
- Avoid using negatives (and especially double negatives) in the question and the answers.
- Test higher order thinking (comprehension and critical thinking). Avoid simple recall.
- If you specify a finite number of attempts, avoid trick questions and try to keep wording clear and unambiguous.
- Make all distractors plausible.
- Use "All of the above" and "None of the above" answer options with caution. If a learner can identify at least two correct answers, it can give away the answer with only partial comprehension.

Adding a Multiple Choice Problem

You add multiple choice problems in Studio by selecting the **Problem** component type and then using either the simple editor or the advanced editor to specify the prompt and the answer options.

Note: You can begin work on the problem in the simple editor, and then switch to the advanced editor. However, after you save any changes you make in the advanced editor, you cannot switch back to the simple editor.

Use the Simple Editor to Add a Multiple Choice Problem

When you add a multiple choice problem, you can choose one of these templates.

- **Multiple Choice**
- **Multiple Choice with Hints and Feedback**

These templates include the Markdown formatting that you use in the simple editor to add a problem without, or with, hints and feedback. To use the simple editor to add a problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Problem**.
2. From the list of **Common Problem Types**, select the type of problem you want to add. Studio adds a template for the problem to the unit.
3. Select **Edit**. The simple editor opens to a template that shows the Markdown formatting that you use for this problem type.
4. Replace the guidance provided by the template to add your own text for the question or prompt, answer options, explanation, and so on.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.

5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Use the Advanced Editor to Add a Multiple Choice Problem

You can use the advanced editor to identify the elements of a multiple choice problem with OLX. For more information, see *Multiple Choice Problem OLX Reference*. To use the advanced editor to add a problem, follow these steps.

1. Follow steps 1-3 for creating the problem in the simple editor.
2. Select **Advanced Editor**. The advanced editor opens the template and shows the OLX markup that you can use for this problem type.
3. Replace the guidance provided by the template to add your own text. For example, replace the question or prompt, answer options, and explanation.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.
4. Update the OLX to add optional elements and attributes required for your problem.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Adding Feedback to a Multiple Choice Problem

For an overview of feedback in problems, see *Adding Feedback and Hints to a Problem*. You can add feedback for each of the answer options you provide in the problem. Use the following guidelines when providing feedback.

- Add feedback to incorrect answers to target common misconceptions and mistakes.
- Ensure feedback provides some guidance to the learner about how to arrive at the correct answer.
- Add feedback for the correct answer to reinforce why the answer is correct. Because learners are able to guess, ensure that feedback provides a reason why the answer is correct for learners who might have selected that answer by chance.

You can add feedback in a multiple choice problem using the simple editor or the advanced editor.

Configuring Feedback in the Simple Editor

You can configure feedback in the simple editor. When you add a multiple choice problem, select the template **Multiple Choice with Hints and Feedback**. This template has example feedback syntax that you can replace.

```
( ) answer {{Feedback for learners who select this answer.}}
```

For example, the following problem has feedback for every answer option.

```
>>Which of the following is an example of a vegetable?||You can select only one␣
↪option.<<

( ) apple {{An apple is the fertilized ovary that comes from an apple tree
and contains seeds classifying it as a fruit.}}
( ) pumpkin {{A pumpkin is the fertilized ovary of a squash plant and
contains seeds classifying it as a fruit.}}
(x) potato {{A potato is an edible part of a plant in tuber form and is
```

```
classified as a vegetable}}
( ) tomato {{Many people mistakenly think a tomato is a vegetable. However,
because a tomato is the fertilized ovary of a tomato plant and contains
seeds it is classified as a fruit.}}
```

Configuring Feedback in the Advanced Editor

In the advanced editor, you configure feedback with the following syntax.

```
<choice correct="false">Choice Label
  <choicehint>Feedback for when learner selects this answer.</choicehint>
</choice>
```

For example, the following problem has feedback for each answer.

```
<problem>
  <multiplechoiceresponse>
    <label>Which of the following is an example of a vegetable?</label>
    <description>You can select only one option.</description>
    <choicegroup type="MultipleChoice">
      <choice correct="false">apple
        <choicehint>An apple is the fertilized ovary that comes from an apple
         tree and contains seeds classifying it as a fruit.</choicehint>
      </choice>
      <choice correct="false">pumpkin
        <choicehint>A pumpkin is the fertilized ovary of a squash plant
         and contains seeds classifying it as a fruit.</choicehint>
      </choice>
      <choice correct="true">potato
        <choicehint>A potato is an edible part of a plant in tuber form and
         is classified as a vegetable.</choicehint>
      </choice>
      <choice correct="false">tomato
        <choicehint>Many people mistakenly think a tomato is a vegetable.
         However, because a tomato is the fertilized ovary of a tomato plant
         and contains seeds it is classified as a fruit.</choicehint>
      </choice>
    </choicegroup>
  </multiplechoiceresponse>
</problem>
```

Customizing Feedback Labels

By default, the feedback labels shown to learners are **Correct** and **Incorrect**. If you do not define feedback labels, learners see these terms when they submit an answer, as in the following example.

```
Incorrect: A pumpkin is the fertilized ovary of a squash plant and contains
seeds classifying it as a fruit.
```

You can configure the problem to override the default labels. For example, you can configure a custom label for a specific wrong answer.

```
Not Quite: Many people mistakenly think a tomato is a vegetable. However,
because a tomato is the fertilized ovary of a tomato plant and contains seeds
it is classified as a fruit.
```

Note: The default labels **Correct** and **Incorrect** display in the learner's requested language. If you provide custom labels, they display as you define them to all learners. They are not translated into different languages.

Customize Feedback Labels in the Simple Editor

In the simple editor, you configure custom feedback labels with the following syntax.

```
( ) Answer {{Label:: Feedback for learners who select this answer.}}
```

That is, you provide the label text, followed by two colon (:) characters, before the feedback text.

For example, the following feedback is configured to use a custom label.

```
( ) tomato {{Not Quite:: Many people mistakenly think a tomato is a
vegetable. However, because a tomato is the fertilized ovary of a tomato
plant and contains seeds, it is a fruit.}}
```

Customize Feedback Labels in the Advanced Editor

In the advanced editor, you configure custom feedback labels with the following syntax.

```
<choice correct="true or false">Answer
  <choicehint label="Custom Label">Feedback for learners who select this
   answer.</choicehint>
</choice>
```

For example, the feedback for the following answer option is configured to use a custom label.

```
<choice correct="false">tomato
  <choicehint label="Not Quite">Many people mistakenly think a tomato is a
   vegetable. However, because a tomato is the fertilized ovary of a tomato
   plant and contains seeds, it is a fruit.</choicehint>
</choice>
```

Adding Hints to a Multiple Choice Problem

You can add hints to a multiple choice problem using the simple editor or the advanced editor. For an overview of hints in problems, see *Adding Feedback and Hints to a Problem*.

Configure Hints in the Simple Editor

In the simple editor, you configure hints with the following syntax.

```
||Hint 1||
||Hint 2||
||Hint n||
```

Note: You can configure any number of hints. The learner views one hint at a time and views the next one by selecting **Hint** again.

For example, the following problem has two hints.

```
||A fruit is the fertilized ovary from a flower.||
||A fruit contains seeds of the plant.||
```

Configure Hints in the Advanced Editor

In the advanced editor, you add the <demandhint> element immediately before the closing </problem> tag, and then configure each hint using the <hint> element.

```
.
.
.
<demandhint>
  <hint>Hint 1</hint>
  <hint>Hint 2</hint>
  <hint>Hint 3</hint>
</demandhint>
</problem>
```

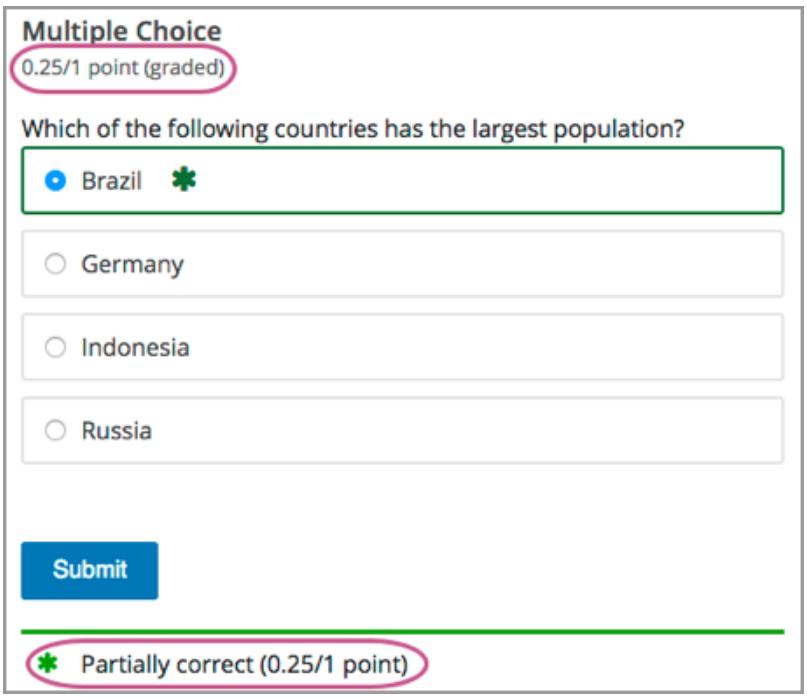
For example, the following OLX for a multiple choice problem shows two hints.

```
.
.
.
</multiplechoiceresponse>
<demandhint>
  <hint>A fruit is the fertilized ovary from a flower.</hint>
  <hint>A fruit contains seeds of the plant.</hint>
</demandhint>
</problem>
```

Awarding Partial Credit in a Multiple Choice Problem

You can configure a multiple choice problem so that specific incorrect answers award learners partial credit for the problem. You must use the advanced editor to configure partial credit.

In the following example, the learner selected a wrong answer and received partial credit.



You can specify what percentage of the points for the problem a learner receives for an incorrect answer. If you do not specify the percentage, the system uses the default of 50%.

For an overview of partial credit in problems, see *Awarding Partial Credit for a Problem*.

Configure a Multiple Choice Problem to Award Partial Credit

To configure a multiple choice problem to award partial credit for a specific answer, you add the following attributes to the problem OLX.

- Add the `partial_credit="points"` attribute to the `<multiplechoiceresponse>` element.
- For each answer that you intend to award partial credit, in the `<choice>` element set the value of the `correct` attribute to `"partial"`.
- Optionally, define the percentage of the problem score to award for each answer. Add the `point_value` attribute to the `<choice>` element, and enter its value as a decimal. For example, add `point_value="0.25"` to award 25% of the points to learners who select that answer. The percentage awarded should reflect how close the learner has gotten to a full understanding of the concept. If you do not add the `point_value` attribute, the system uses the default of 50%.

For example, the following OLX shows a multiple choice problem that provides partial credit of 25% for an answer option.

```
<problem>
  <multiplechoiceresponse partial_credit="points">
    <label>Which of the following is a vegetable?</label>
    <choicegroup type="MultipleChoice">
      .
      .
      .
      <choice correct="partial" point_value="0.25">tomato </choice>
    </choicegroup>
```

```
  </multiplechoiceresponse>
</problem>
```

Multiple Choice Problem OLX Reference

Note: You can also set attributes and options by adding a <script> element. For more information, see *Using the Script Element*.

Template

```
<problem>
  <multiplechoiceresponse>
    <label>Question or prompt text</label>
    <description>Optional information about how to answer the question</description>
    <choicegroup type="MultipleChoice">
      <choice correct="false" name="a">Incorrect choice
        <choicehint>Hint for incorrect choice.</choicehint>
      </choice>
      <choice correct="true" name="b">Correct choice
        <choicehint>Hint for correct choice.</choicehint>
      </choice>
    </choicegroup>
    <solution>
      <div class="detailed-solution">
        <p>Optional header for the explanation or solution</p>
        <p>Optional explanation or solution text</p>
      </div>
    </solution>
  </multiplechoiceresponse>
<demandhint>
  <hint>Hint 1</hint>
  <hint>Hint 2</hint>
</demandhint>
</problem>
```

Elements

For multiple choice problems, the <problem> element can include this hierarchy of child elements.

```
<multiplechoiceresponse>
    <label>
    <description>
    <choicegroup>
          <choice>
              <choicehint>
    <solution>
<demandhint>
    <hint>
```

In addition, standard HTML tags can be used to format text.

<multiplechoiceresponse>

Required. Indicates that the problem is a multiple choice problem.

Attributes

Attribute	Description
partial_credit	Optional. Specifies that the problem can award partial credit. If used, must be set to "points".

Children

- <label>
- <description>
- <choicegroup>
- <solution>

<label>

Required. Identifies the question or prompt. You can include HTML tags within this element.

Attributes

None.

Children

None.

<description>

Optional. Provides clarifying information about how to answer the question. You can include HTML tags within this element.

Attributes

None.

Children

None.

<choicegroup>

Required. Indicates the beginning of the list of answer options.

Attributes

Attribute	Description
type	Required. Must be set to "MultipleChoice".

Additional attributes are available to support *advanced options*.

Children

<choice>

<choice>

Required. Lists an answer option.

Attributes

Attribute	Description
correct	Indicates a correct, incorrect, or partially correct answer. • When set to "true", the choice is a correct answer. At least one required. • When set to "false", the choice is an incorrect answer. • When set to "partial", the learner receives partial credit for selecting the answer. You can specify more than one correct or partially correct answer, but learners can select only one choice to submit as their answer.
point_value	When correct="partial", indicates the percentage, as a decimal, of the points the learner receives for selecting this option. If point_value is not specified for a partial credit answer, 50% is used by default.
name	A unique name that is used internally to refer to the choice.

Additional attributes are available to support *advanced options*.

Children

<choicehint>

<choicehint>

Optional. Specifies feedback for the answer.

Attributes

None.

Children

None.

<solution>

Optional. Identifies the explanation or solution for the problem, or for one of the questions in a problem that includes multiple questions.

This element contains an HTML division . The division contains one or more paragraphs <p> of explanatory text.

<demandhint>

Optional. Specifies hints for the learner. For problems that include multiple questions, the hints apply to the entire problem.

Attributes

None.

Children

<hint>

<hint>

Required. Specifies additional information that learners can access if needed.

Children

None.

Advanced Options for Multiple Choice Problems

Multiple choice problems have several advanced options. You can change the order of answers in the problem, include explanations that appear when a learner selects a specific incorrect answer, or present a random set of choices to each learner. For more information, see the following sections.

- *Shuffle Answers in a Multiple Choice Problem*

- *Targeted Feedback in a Multiple Choice Problem*
- *Answer Pools in a Multiple Choice Problem*
- *Using the Script Element*

Shuffle Answers in a Multiple Choice Problem

Optionally, you can configure a multiple choice problem so that it shuffles the order of possible answers.

For example, one view of a problem could be as follows.

```
What Apple device competed with the portable CD player?

( ) The iPad
( ) Napster
( ) The iPod
( ) The vegetable peeler
```

Another view of the same problem, for a different learner or for the same learner on a subsequent view of the unit, could be as follows.

```
What Apple device competed with the portable CD player?

( ) The iPad
( ) The iPod
( ) The vegetable peeler
( ) Napster
```

You can also shuffle some answers, but not others. For example, you might want to include the answer "All of the above" and have it always appear at the end of the list, but shuffle the other answers.

You can configure the problem to shuffle answers using the simple editor or advanced editor. To shuffle the answers, you also edit the problem to set **Randomization** to a value other than **Never**. For more information, see *Randomization*.

Use the Simple Editor to Shuffle Answers

You can configure the problem to shuffle answers in the simple editor. To add shuffling to this problem, you add an exclamation point character `!` between the parentheses formatting for the first answer option.

```
>>What Apple device competed with the portable CD player?<<
    (!) The iPad
    ( ) Napster
    (x) The iPod
    ( ) The vegetable peeler
```

To make the location of an answer fixed in a shuffled list, add `@` between the parentheses formatting for that answer.

```
>>What Apple device competed with the portable CD player?<<
    (!) The iPad
    ( ) Napster
    (x) The iPod
    ( ) The vegetable peeler
    (@) All of the above
```

You can combine symbols within the parentheses as necessary. For example, to show the correct answer in a fixed location, you can use both `x` and `@`.

```
(x@) The iPod
```

After you complete problem setup in the simple editor, select **Edit** and then **Settings** to specify an option other than **Never** for the **Randomization** setting.

Use the Advanced Editor to Shuffle Answers

You can configure the problem to shuffle answers by editing the OLX in the advanced editor.

To add shuffling to a problem, you add `shuffle="true"` to the `<choicegroup>` element.

```
<problem>
  <multiplechoiceresponse>
    <label>What Apple device competed with the portable CD player?</label>
    <choicegroup type="MultipleChoice" shuffle="true">
      <choice correct="false">The iPad</choice>
      <choice correct="false">Napster</choice>
      <choice correct="true">The iPod</choice>
      <choice correct="false">The vegetable peeler</choice>
    </choicegroup>
  </multiplechoiceresponse>
</problem>
```

To make the location of an answer fixed in a shuffled list, add `fixed="true"` to the `choice` element for the answer.

```
<problem>
  <multiplechoiceresponse>
    <label>What Apple device competed with the portable CD player?</label>
    <choicegroup type="MultipleChoice" shuffle="true">
      .
      .
      .
      <choice correct="false" fixed="true">All of the above</choice>
    </choicegroup>
  </multiplechoiceresponse>
</problem>
```

Then, you select **Settings** to specify an option other than **Never** for the **Randomization** setting.

Targeted Feedback in a Multiple Choice Problem

You can configure a multiple choice problem so that explanations for specific answers are automatically shown to learners. You can use these explanations to guide learners towards the right answer. Therefore, targeted feedback is most useful for multiple choice problems for which learners are allowed multiple attempts.

Use the Advanced Editor to Configure Targeted Feedback

You configure the problem to provide targeted feedback by editing the OLX in the advanced editor.

- Add a `targeted-feedback` attribute to the `<multiplechoiceresponse>` element, with no value: `<multiplechoiceresponse targeted-feedback="">`.

- Add an `explanation-id` attribute with a unique value to each of the `<choice>` elements: `<choice correct="false" explanation-id="feedback1">`.
- You can use the `<solution>` element for the correct answer.
- Add a `<targetedfeedbackset>` element after the `<multiplechoiceresponse>` element.
- Within `<targetedfeedbackset>`, add one or more `<targetedfeedback>` elements.
- Within each `<targetedfeedback>` element, add one of the unique identifying `explanation-id` attributes to map that feedback to a specific answer choice.
- Within each `<targetedfeedback>` element use HTML formatting, such as `<p></p>` tags, to enter your explanation for the specified answer option.

For example, the OLX for a multiple choice problem follows, showing a unique ID for each answer choice. This is immediately followed by OLX that defines the targeted feedback.

```
<problem>
  <multiplechoiceresponse targeted-feedback="">
    <label>What Apple device competed with the portable CD player?</label>
    <choicegroup type="MultipleChoice">
      <choice correct="false" explanation-id="feedback1">The iPad</choice>
      <choice correct="false" explanation-id="feedback2">Napster</choice>
      <choice correct="true" explanation-id="correct">The iPod</choice>
      <choice correct="false" explanation-id="feedback3">The vegetable peeler</choice>
    </choicegroup>
    <solution explanation-id="correct">
      <div class="detailed-solution">
        <p>The iPod directly competed with portable CD players.</p>
      </div>
    </solution>
  </multiplechoiceresponse>
  <targetedfeedbackset>
    <targetedfeedback explanation-id="feedback1">
      <div class="detailed-targeted-feedback">
        <p>Targeted Feedback</p>
        <p>The iPad came out later and did not directly compete with
         portable CD players.</p>
       </div>
    </targetedfeedback>
    <targetedfeedback explanation-id="feedback2">
      <div class="detailed-targeted-feedback">
        <p>Targeted Feedback</p>
        <p>Napster was not an Apple product.</p>
      </div>
    </targetedfeedback>
    <targetedfeedback explanation-id="feedback3">
      <div class="detailed-targeted-feedback">
        <p>Targeted Feedback</p>
        <p>Vegetable peelers do not play music.</p>
      </div>
    </targetedfeedback>
  </targetedfeedbackset>
</problem>
```

Answer Pools in a Multiple Choice Problem

You can configure a multiple choice problem so that a random subset of choices are shown to each learner. For example, you can add 10 possible choices to the problem, and each learner views a set of five choices.

The answer pool must have at least one correct answer. It can have more than one correct answer. In each set of choices shown to a learner, one correct answer is included. For example, you can configure two correct answers in the set of choices. One of the two correct answers is included in each set that a learner views.

Use the Advanced Editor to Configure Answer Pools

You configure the problem to provide answer pools by editing the OLX for the problem in the advanced editor.

- In the `<choicegroup>` element, add the `answer-pool` attribute, with the numerical value indicating the number of answer options to show to learners. For example, `<choicegroup answer-pool="4">`.
- If you include more than one correct answer among the options, for each correct answer add an `explanation-id` attribute with a unique value to the `<choice>` element: `<choice correct="false" explanation-id="correct1">`.
- If you include more than one correct answer among the options, for each `<solution>` element, add an `explanation-id` attribute and a value that maps back to a specific correct answer. For example, `<solution explanation-id="correct1">`.
- Place the `<solution>` elements within a `<solutionset>` element.

Note: If the choices include only one correct answer, you do not have to use the `explanation-id` in either the `<choice>` or `<solution>` element. You do still use the `<solutionset>` element to wrap the `<solution>` element.

For example, for the following multiple choice problem, a learner will see four choices. In each set, one of the choices will be one of the two correct choices. The explanation shown for the correct answer is the one with the same explanation ID.

```
<problem>
  <multiplechoiceresponse>
    <label>What Apple devices let you carry your digital music library in your pocket?
↪</label>
    <description>You can select only one option.</description>
    <choicegroup type="MultipleChoice" answer-pool="4">
      <choice correct="false">The iPad</choice>
      <choice correct="false">Napster</choice>
      <choice correct="true" explanation-id="iPod">The iPod</choice>
      <choice correct="false">The vegetable peeler</choice>
      <choice correct="false">The iMac</choice>
      <choice correct="true" explanation-id="iPhone">The iPhone</choice>
    </choicegroup>
    <solutionset>
      <solution explanation-id="iPod">
        <div class="detailed-solution">
          <p>Explanation</p>
          <p>The iPod is Apple's portable digital music player.</p>
        </div>
      </solution>
      <solution explanation-id="iPhone">
        <div class="detailed-solution">
```

```
          <p>Explanation</p>
          <p>In addition to being a cell phone, the iPhone can store and play
           your digital music.</p>
        </div>
      </solution>
    </solutionset>
  </multiplechoiceresponse>
</problem>
```

Using the Script Element

You can use the <script> element to programmatically set attributes and options for your multiple choice problems. You could use this feature to display different questions/answers depending on variable factors, like time of day, or randomly generated numbers.

Use the Advanced Editor to Configure the Script Element

You must use the advanced editor to configure a <script> element.

The contents of the <script> element must be enclosed in <![CDATA[...]]> markers, to indicate that the enclosed code should not be interpreted as XML.

The code in the <script> element is run on the server before the problem is shown to learners. Note that only Python script types are supported.

The following OLX example uses random numbers to generate different answer choices for each learner, and mathematical operators to determine each choice's correctness.

```
<problem>
    <script type="text/python">
    <![CDATA[
    random.seed(anonymous_student_id)  # Use different random numbers for each␣
↪student.
    a = random.randint(1,10)
    b = random.randint(1,10)
    c = a + b

    ok0 = c % 2 == 0 # check remainder modulo 2
    text0 = "$a + $b is even"

    ok1 = c % 2 == 1 #check remainder modulo 2
    text1 = "$a + $b is odd"
    ]]>
    </script>
    <multiplechoiceresponse>
      <label>Is $a+$b even or odd? Select the true statement.</label>
      <choicegroup type="MultipleChoice">
        <choice correct="$ok0">$text0 ... (should be $ok0)</choice>
        <choice correct="$ok1">$text1 ... (should be $ok1)</choice>
      </choicegroup>
    </multiplechoiceresponse>
</problem>
```

Numerical Input Problem

Note: EdX offers full support for this problem type.

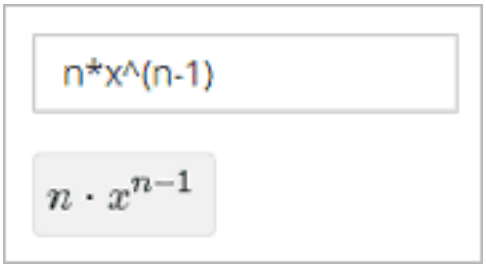
The numerical input problem type is a core problem type that can be added to any course. At a minimum, numerical input problems include a question or prompt and a response field for a numeric answer. By adding hints, feedback, or both, you can give learners guidance and help when they work on a problem.

For more information about the core problem types, see Working with Problem Components.

Overview

In numerical input problems, learners enter numbers or specific and relatively simple mathematical expressions to answer a question. The LMS automatically converts the answer that learners enter into a symbolic expression that appears below the response field.

Responses for numerical input problems can include integers, fractions, and constants such as pi and g. Responses can also include text representing common functions, such as square root (sqrt) and log base 2 (log2), as well as trigonometric functions and their inverses, such as sine (sin) and arcsine (arcsin). For these functions, learners enter text that is converted into mathematical symbols. The following example shows a response entered by a learner and the numerical expression that results.



For more information about how learners enter expressions, see Entering Mathematical and Scientific Expressions in the *EdX Learner's Guide* or Entering Mathematical and Scientific Expressions in the *Open edX Learner's Guide*.

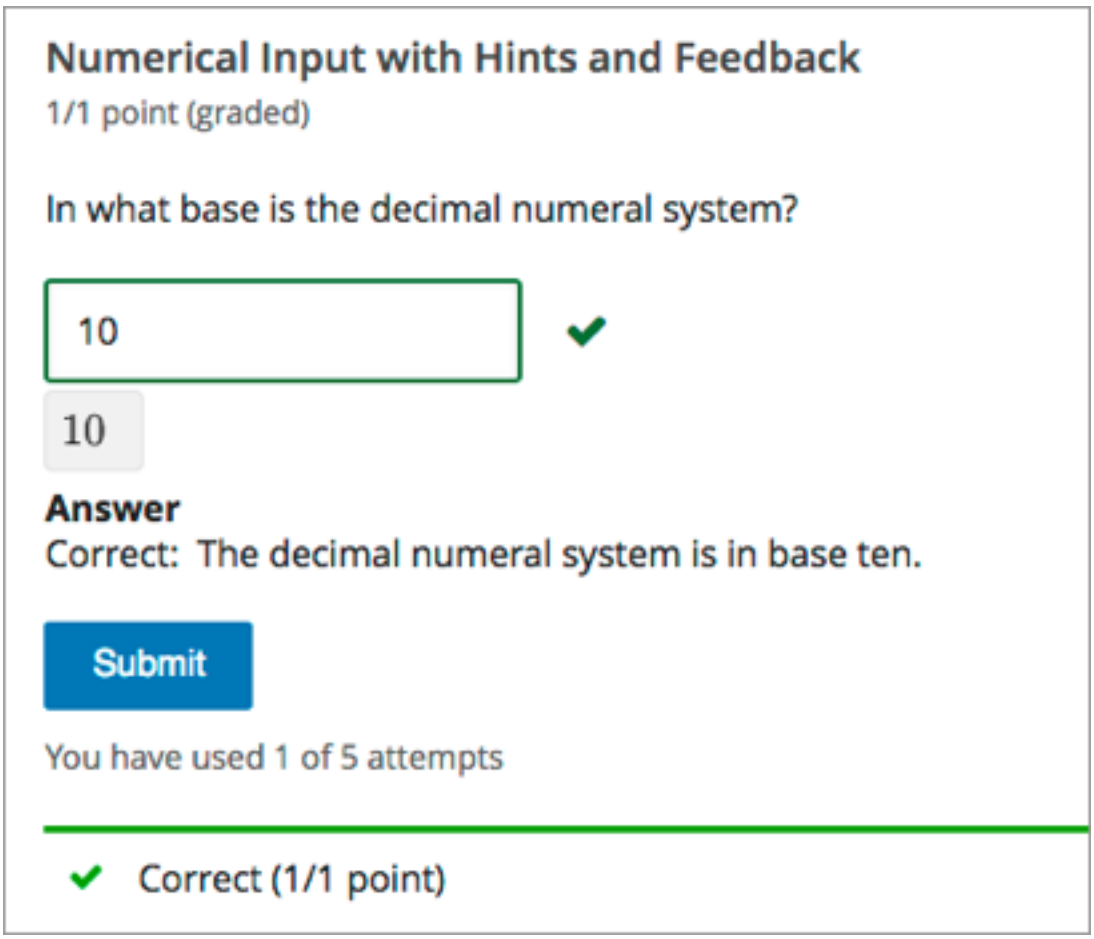
Some of the options for numerical input problems include the following.

- You can specify a correct answer explicitly or use a Python script.
- You can specify a margin of error, or tolerance, for the answers to numerical input problems so that learners' responses do not have to be exact.

Note: You can make a calculator tool available to your learners on every unit page. For more information, see Calculator Tool.

Example Numerical Input Problem

In the LMS, learners enter a value into a response field to complete a numerical input problem. An example of a completed numerical input problem follows.



To add the example problem illustrated above, in Studio you use the simple editor to enter the following text and Markdown formatting.

```
>>In what base is the decimal numeral system?<<

=10 {{The decimal numeral system is in base ten.}}
```

The open learning XML (OLX) markup for this example numerical input problem follows.

```
<problem>
  <numericalresponse answer="10">
    <label>In what base is the decimal numeral system?</label>
    <formulaequationinput/>
    <solution>
      <div class="detailed-solution">
        <p>Explanation</p>
        <p>The decimal numeral system is base ten.</p>
      </div>
    </solution>
  </numericalresponse>
</problem>
```

Analyzing Performance on Numerical Input Problems

For the numerical input problems in your course, you can use edX Insights to review aggregated learner performance data and examine submitted answers. For more information, see Using edX Insights.

Adding a Numerical Input Problem

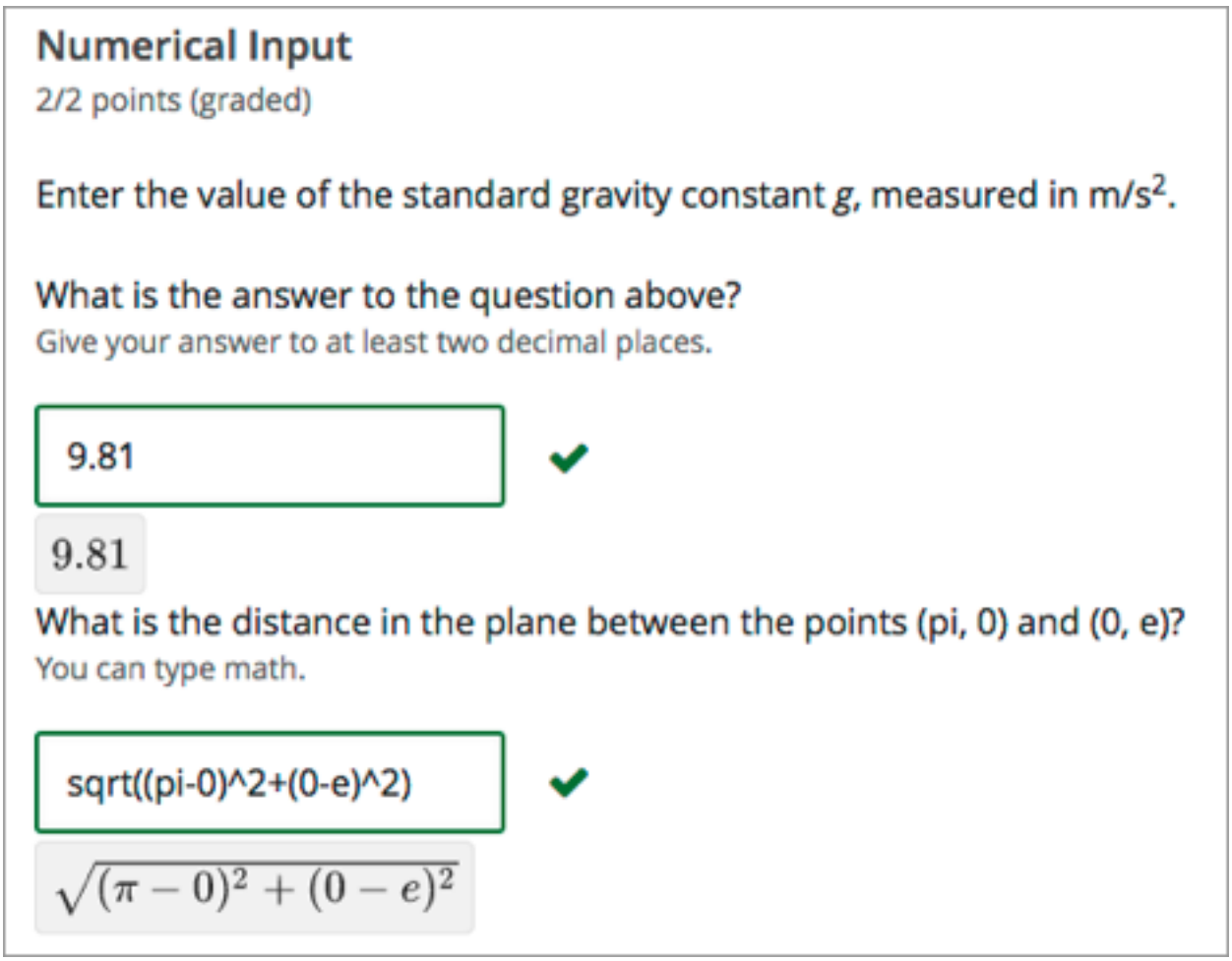
You add numerical input problems in Studio by selecting the **Problem** component type and then using either the simple editor or the advanced editor to specify the prompt and the acceptable answer or answers.

Note: You can begin work on the problem in the simple editor, and then switch to the advanced editor. However, after you save any changes you make in the advanced editor, you cannot switch back to the simple editor.

Before you add a numerical input problem, consider the following features and limitations of the simple and advanced editors.

- If your problem text contains italics, bold, or special characters, the simple editor does not support these values. Use the advanced editor to add an HTML paragraph (`<p>`) element and HTML formatting tags as needed.
- If your problem contains a Python script, use the advanced editor.

For example, you must use the advanced editor to define the following numerical input problem. It contains two questions, one of which relies on character formatting, and the other that uses a Python script.



The OLX for this example follows.

Note: The second question in the following example includes a Python script. When you add a script to a problem component, do not add to or change its internal indentation. A "jailed code" error message appears when you save the problem in Studio if the `<script>` element is indented.

```
<problem>
  <numericalresponse answer="9.80665">
      <p>Enter the value of the standard gravity constant <i>g</i>,
       measured in m/s<sup>2</sup>.</p>
      <label>What is the answer to the question above?</label>
      <description>Give your answer to at least two decimal places.</description>
      <responseparam type="tolerance" default="0.01" />
      <formulaequationinput />
<script type="loncapa/python">
computed_response = math.sqrt(math.fsum([math.pow(math.pi,2), math.pow(math.e,2)]))
</script>
      <solution>
          <div class="detailed-solution">
              <p>Explanation</p>
              <p>The standard gravity constant is defined to be precisely
               9.80665 m/s<sup>2</sup>. This is 9.80 to two decimal places.
               Entering 9.8 also works.</p>
```

```
          </div>
      </solution>
  </numericalresponse>

  <numericalresponse answer="$computed_response">
      <label>What is the distance in the plane between the points (pi, 0)
       and (0, e)?</label>
      <description>You can type math.</description>
      <responseparam type="tolerance" default="0.0001" />
      <formulaequationinput />
      <solution>
          <div class="detailed-solution">
              <p>Explanation</p>
              <p>By the distance formula, the distance between two points in
               the plane is the square root of the sum of the squares of the
               differences of each coordinate. Even though an exact numerical
               value is checked in this case, the easiest way to enter this
               answer is to type <code>sqrt(pi^2+e^2)</code> into the editor.
               Other answers like <code>sqrt((pi-0)^2+(0-e)^2)</code> also
               work.</p>
          </div>
      </solution>
  </numericalresponse>
</problem>
```

For more information about including a Python script in a problem, see *Write-Your-Own-Grader Problem*.

Use the Simple Editor to Add a Numerical Input Problem

When you add a numerical input problem, you can choose one of these templates.

- **Numerical Input**
- **Numerical Input with Hints and Feedback**

These templates include the Markdown formatting that you use in the simple editor to add a problem without, or with, hints and feedback. To use the simple editor to add a problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Problem**.
2. From the list of **Common Problem Types**, select the type of problem you want to add. Studio adds a template for the problem to the unit.
3. Select **Edit**. The simple editor opens to a template that shows the Markdown formatting that you use for this problem type.
4. Replace the guidance provided by the template to add your own text for the question or prompt, answer options, explanation, and so on.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Use the Advanced Editor to Add a Numerical Input Problem

You can use the advanced editor to identify the elements of a numerical input problem with OLX. For more information, see *Numerical Input Problem OLX Reference*. To use the advanced editor to add a problem, follow these steps.

1. Follow steps 1-3 for creating the problem in the simple editor.
2. Select **Advanced Editor**. The advanced editor opens the template and shows the OLX markup that you can use for this problem type.
3. Replace the guidance provided by the template to add your own text. For example, replace the question or prompt, answer options, and explanation.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.
4. Update the OLX to add optional elements and attributes required for your problem.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Adding a Tolerance, Multiple Correct Responses, or a Range

To give learners the option to receive full credit for a close approximation of the correct answer, and to support a wide range of possible correct numerical answers, you can specify a tolerance for the correct answer, multiple individual correct answers, or a range of values to mark as correct for the numerical input problem type.

Adding a Tolerance

You can specify a margin of error or tolerance for learner responses. You can specify a percentage, number, or range.

Add a Tolerance in the Simple Editor

To add a tolerance in the simple editor you use the following Markdown formatting.

- To specify a number on either side of the correct answer, after the answer value add `+-{number}`. For example, to include a tolerance of 5, add `+-5`.
- To specify a percentage on either side of the correct answer, after the answer value add `+-{number}%`. For example, to include a 2% tolerance, add `+-2%`.

Add a Tolerance in the Advanced Editor

To add a tolerance in the advanced editor you include a `<responseparam>` element with a `type="tolerance"` attribute and a `default` attribute set to either a number or a percentage value.

The following example shows a problem with a decimal tolerance.

```
<problem>
  <numericalresponse answer="ANSWER (NUMBER)">
    <label>Question text</label>
    <responseparam type="tolerance" default=".02" />
    <formulaequationinput />
  </numericalresponse>
</problem>
```

The following example shows a problem with a percentage tolerance.

```
<problem>
  <numericalresponse answer="ANSWER (NUMBER)">
    <label>Question text</label>
    <responseparam type="tolerance" default="3%" />
    <formulaequationinput />
  </numericalresponse>
</problem>
```

Adding Multiple Correct Responses

You can specify more than one specific, correct response for numerical input problems. To do this, you can use the simple editor or the advanced editor.

If you specify multiple correct responses, you cannot also specify a tolerance, a range, or a text string as correct answers. For example, when you define multiple correct responses, you can specify a numeric value for each correct answer but not a tolerance, range, or text string.

Add Multiple Correct Responses in the Simple Editor

To specify additional correct responses in the simple editor, include `or=` before each additional correct response.

```
>>How many miles away from Earth is the sun?||Use scientific notation to answer.<<

= 9.3*10^7
or= 9.296*10^7
```

Add Multiple Correct Responses in the Advanced Editor

To specify an additional correct response in the advanced editor, within the `<numericalresponse>` element add the `<additional_answer />` element with an `answer=""` attribute value.

```
<problem>
  <numericalresponse answer="9.3*10^7">
    <label>How many miles away from Earth is the sun?</label>
    <description>Use scientific notation to answer.</description>
    <additional_answer answer="9.296*10^7"/>
    <formulaequationinput/>
  </numericalresponse>
</problem>
```

Specifying an Answer Range

You can specify an answer range so that any learner response within that range is marked correct. To format an answer range, you provide the starting and ending values and then separate them with a comma character (`,`). You then surround the range with bracket (`[ ]`) or parentheses characters (`( )`), or a combination of one bracket and one parenthesis.

- Use a bracket to include the number next to it in the range, as in a less than or equal to, or greater than or equal to, inequality.
- Use a parenthesis to exclude the number from the range, as in a less than or greater than inequality.

For example, to identify the correct answers as 5, 6, or 7, but not 8, specify `[5,8)`. To identify the correct answers as 6, 7, and 8, but not 5, specify `(5,8]`.

To specify a range in the simple editor, you enter the complete, formatted range after the equals sign: `=[5,8)` or `=(5,8]`.

To specify a range in the advanced editor, you enter the complete, formatted range in the `<numericalresponse>` element as the value for the `answer` attribute: `<numericalresponse answer="[5,8)">` or `<numericalresponse answer="(5,8]">`

Adding Feedback to a Numerical Input Problem

For an overview of feedback in problems, see *Adding Feedback and Hints to a Problem*. In numerical input problems, you can provide feedback for correct responses. If you define multiple correct responses, you can define feedback for each response.

Note: You cannot provide feedback for incorrect answers in numerical input problems.

In numerical input problems, use feedback to reinforce the process used to arrive at the correct answer.

Configure Feedback in the Simple Editor

You can configure feedback in the simple editor. When you add a numerical input problem, select the template **Numerical Input with Hints and Feedback**. This template has example formatted feedback that you can replace with your own text.

In the simple editor, you configure feedback for a numerical input problem with the following Markdown formatting.

```
=Correct Answer {{Feedback for the correct answer.}}
```

For example, the following problem has feedback for the correct answer.

```
>>What is the arithmetic mean for the following set of numbers? (1, 5, 6, 3, 5)<<

=4 {{The mean for this set of numbers is 20 / 5 which equals 4.}}
```

If you define multiple correct responses, you can define feedback for each response.

Configure Feedback in the Advanced Editor

In the advanced editor, you configure answer feedback with the following syntax.

```
<problem>
  <numericalresponse answer="Correct Answer">
    <label>Question text</label>
    <formulaequationinput />
    <correcthint>Feedback for the correct answer</correcthint>
  </numericalresponse>
</problem>
```

For example, the following problem has feedback for the correct answer.

```
<problem>
  <numericalresponse answer="4">
    <label>What is the arithmetic mean for the following set of numbers?
     (1, 5, 6, 3, 5)</label>
    <formulaequationinput />
    <correcthint>The mean for this set of numbers is 20 / 5 which equals 4.</
↪correcthint>
  </numericalresponse>
</problem>
```

If you define multiple correct responses, you can define feedback for each response.

Customizing Feedback Labels

By default, the feedback label shown to learners is **Correct** . If you do not define a feedback label, learners see this term when they submit a correct answer, as in the following example.

```
Correct: The mean for this set of numbers is 20 / 5 which equals 4.
```

You can configure the problem to override the default label. For example, you can configure a custom label for the answer.

```
Good job: The mean for this set of numbers is 20 / 5 which equals 4.
```

Note: The default label, **Correct**, displays in the learner's requested language. If you provide a custom label, it displays as you define it to all learners. It is not translated into different languages.

Customize a Feedback Label in the Simple Editor

In the simple editor, you configure a custom feedback label with the following syntax.

```
=4 {{Label:: Feedback}}
```

That is, you provide the label text, followed by two colon (:) characters, before the feedback text.

For example, the following feedback is configured to use a custom label.

```
=4 {{Good Job:: The mean for this set of numbers is 20 / 5 which equals 4.}}
```

Customize a Feedback Label in the Advanced Editor

In the advanced editor, you configure custom feedback labels with the following syntax.

```
<correcthint label="Custom Label">Feedback</correcthint>
```

For example, the following feedback is configured to use a custom label.

```
<correcthint label="Good Job">The mean for this set of numbers is 20 / 5 which equals␣
↪4.</correcthint>
```

Adding Hints to a Numerical Input Problem

You can add hints to a numerical input problem using the simple editor or the advanced editor. For an overview of hints in problems, see *Adding Feedback and Hints to a Problem*.

Configure Hints in the Simple Editor

In the simple editor, you configure hints with the following syntax.

```
||Hint 1||
||Hint 2||
||Hint n||
```

Note: You can configure any number of hints. The learner views one hint at a time and views the next one by selecting **Hint** again.

For example, the following problem has two hints.

```
||A fruit is the fertilized ovary from a flower.||
||A fruit contains seeds of the plant.||
```

Configure Hints in the Advanced Editor

In the advanced editor, you add the `<demandhint>` element immediately before the closing `</problem>` tag, and then configure each hint using the `<hint>` element.

```
.
.
.
<demandhint>
  <hint>Hint 1</hint>
  <hint>Hint 2</hint>
  <hint>Hint 3</hint>
</demandhint>
</problem>
```

For example, the following OLX for a multiple choice problem shows two hints.

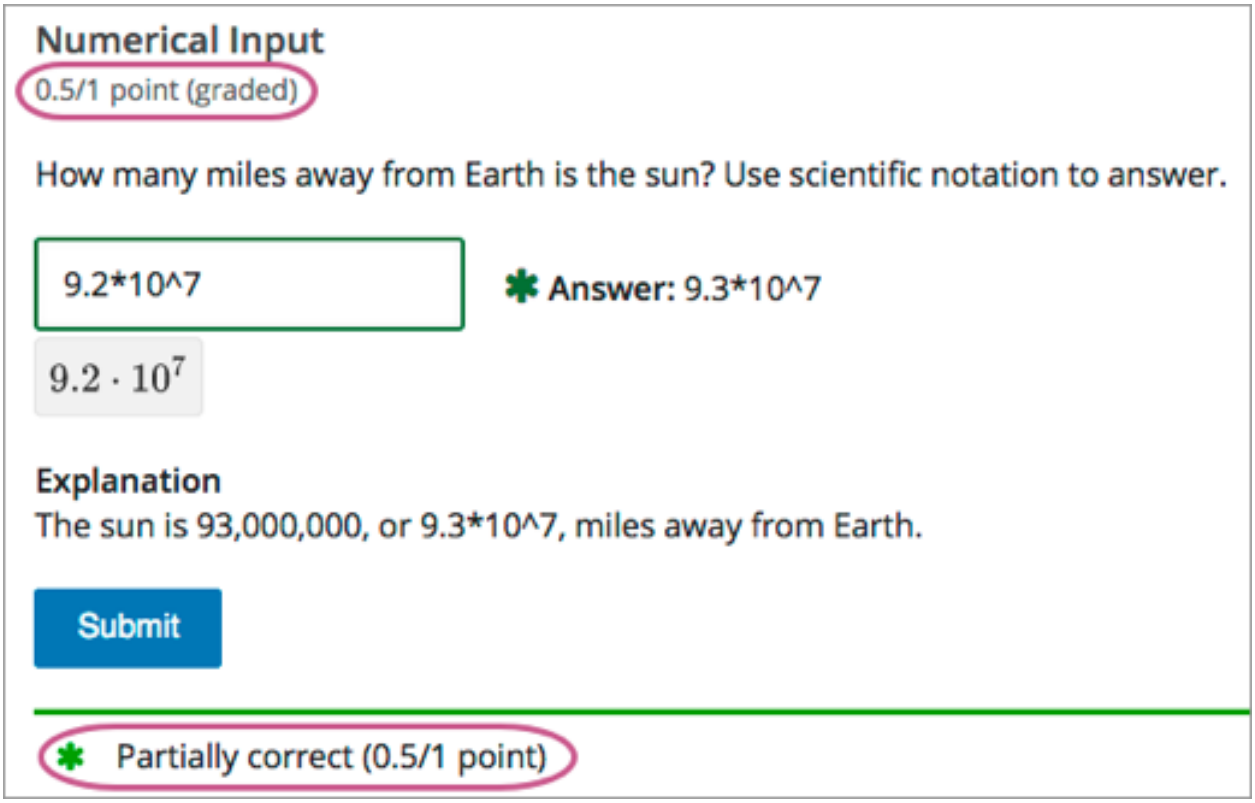
```
.
.
.
</multiplechoiceresponse>
<demandhint>
  <hint>A fruit is the fertilized ovary from a flower.</hint>
```

```
  <hint>A fruit contains seeds of the plant.</hint>
</demandhint>
</problem>
```

Awarding Partial Credit in a Numerical Input Problem

You can configure a numerical input problem to award partial credit to learners who submit an answer that is close or related to the correct answer. You must use the *advanced editor* to configure partial credit.

In the following example, the learner entered an answer that was close to the correct answer and received partial credit.



For an overview of partial credit in problems, see *Awarding Partial Credit for a Problem*.

You can use the following methods to award partial credit in a numerical input problem.

- *Identifying Close Answers*
- *Awarding Partial Credit for Answers in a List*

Note: You can use these methods of awarding partial credit individually or in combination.

Identifying Close Answers

You can configure a numerical input problem so that answers that are close to the correct answer receive partial credit.

To do so, you configure the tolerance for incorrect answers. Learners receive partial credit for close answers based on the tolerance. By default, the tolerance is multiplied by 2 and the following rules are applied.

- An answer within the tolerance receives 100% of the points for the problem.
- An answer within or equal to 2x of the tolerance receives 50%.
- An answer more than 2x the outside of the tolerance receives 0%.

You can optionally specify a different multiplier for the tolerance. For example, you could set the multiplier to 3. In this case, the following rules apply.

- An answer within the tolerance receives 100% of the points for the problem.
- An answer within or equal to 3x of the tolerance receives 50%.
- An answer more than 3x outside of the tolerance receives 0%.

Configure Close Answers for a Numerical Input Problem

To configure a numerical input problem to award partial credit for close answers, you add the following attributes to the problem XML.

- Add the `partial_credit="close"` attribute to the `<numericalresponse>` element.

  You can use close answers in combination with a list. Set the attribute to `partial_credit="close, list"`.

- Optionally, add the `partial_range` attribute to the `<responseparam>` element and set its value to the tolerance multiplier. If you do not set the `partial_range` attribute, 2 is used as the tolerance multiplier.

For example, the following OLX shows a numerical problem that provides partial credit for close answers.

```
<problem>
  <numericalresponse answer="9.3*10^7" partial_credit="close">
    <label>How many miles away from Earth is the sun?</label>
    <description>Use scientific notation to answer.</description>
    <formulaequationinput/>
    <responseparam type="tolerance" default="1%" partial_range="3"/>
  </numericalresponse>
</problem>
```

Awarding Partial Credit for Answers in a List

For some numerical input problems, mistakes do not help a learner arrive at the correct answer. For example, a small mistake can lead to negative instead of positive results, or to an answer that is off by a square root or numerical factor.

For these types of problems, you can configure a list of wrong answers that receive partial credit. Learners who submit answers that are on the list receive 50% of the problem's points.

Configure a List for a Numerical Input Problem

To configure a numerical input problem to award partial credit for answers in a list, you add the following attributes to the problem XML.

- Add the `partial_credit="list"` attribute to the `<numericalresponse>` element.

  You can use a list in combination with close answers. Set the attribute to `partial_credit="close, list"`.

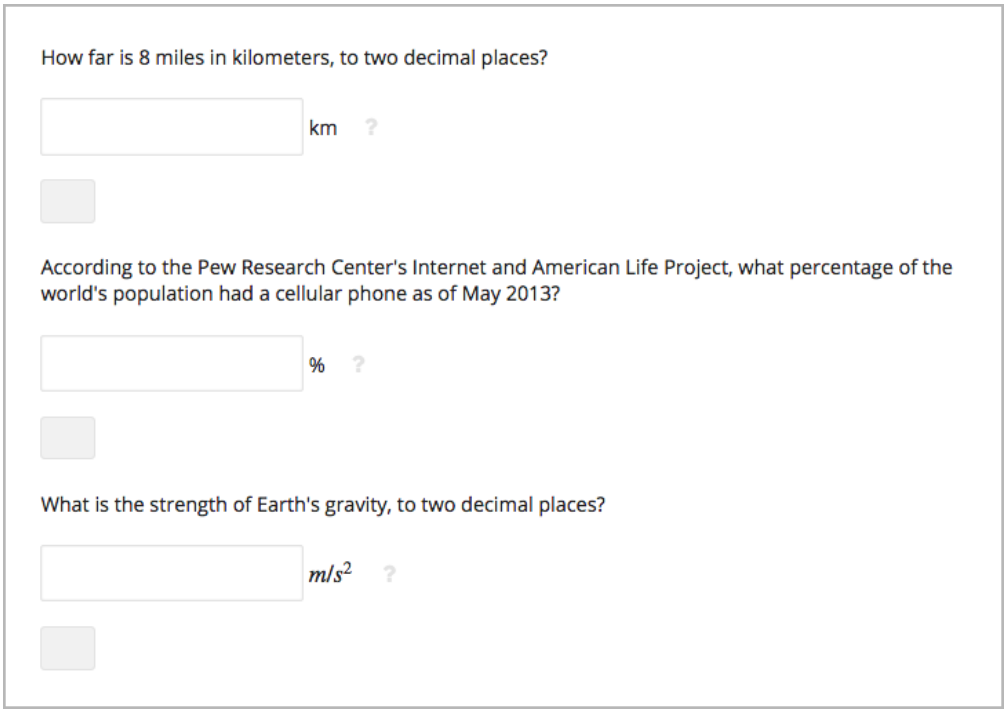
- Add the `partial_answers` attribute to the `<responseparam>` element. Set its value to one or more answers that should earn 50% of the problem's points. Separate multiple values by a comma (,).

For example, the following XML shows the numerical problem template updated to provide partial credit for a different answer.

```
<problem>
  <numericalresponse answer="9.3*10^7" partial_credit="close">
    <label>How many miles away from Earth is the sun?</label>
    <description>Use scientific notation to answer.</description>
    <formulaequationinput />
    <responseparam partial_answers="150*10^6"/>
  </numericalresponse>
</problem>
```

Add Text after the Numeric Response Field

You might want to include a word, phrase, or sentence after the response field in a numerical input problem to help guide your students or resolve ambiguity.



To do this, you use the advanced editor.

In the problem, locate the `formulaequationinput` element. This element creates the response field for the problem and is a child of the `numericalresponse` element.

To add text after the response field, add the `trailing_text` attribute together with the symbol or text that you want to use inside the `formulaequationinput` element. An example problem follows with three questions that use this attribute.

Note: You can use MathJax inside the `trailing_text` attribute, as the third question in this example shows. You cannot use HTML inside this attribute.

```
<problem>
  <numericalresponse answer="12.87">
    <label>How far is 8 miles in kilometers?</label>
    <formulaequationinput trailing_text="km" />
  </numericalresponse>

  <numericalresponse answer="91">
    <label>According to the Pew Research Center's Internet and American Life
     Project, what percentage of the world's population had a cellular phone
     as of May 2013?</label>
    <formulaequationinput trailing_text="%" />
  </numericalresponse>

  <numericalresponse answer="9.81">
    <label>What is the strength of Earth's gravity, to two decimal places?</label>
    <formulaequationinput trailing_text="\(m/s^{2}\)" />
  </numericalresponse>
</problem>
```

Numerical Input Problem OLX Reference

Templates

The following templates represent problems without, and with, a Python script.

Problem with No Tolerance

```
<problem>
  <numericalresponse answer="ANSWER (NUMBER)">
    <label>Question text</label>
    <description>Optional tip</description>
    <formulaequationinput />
    <correcthint>Feedback for the correct answer.</correcthint>
    <solution>
      <div class="detailed-solution">
        <p>Explanation</p>
        <p>TEXT OF SOLUTION</p>
      </div>
    </solution>
  </numericalresponse>
</problem>
```

Answer Created Using a Script

Note: The following example includes a Python script. When you add a script to a problem component, make sure that it is not indented. A "jailed code" error message appears when you save the problem in Studio if the script element is indented.

```
<problem>
  <numericalresponse answer="$computed_response">
```

```
    <label>Question text</label>
    <description>Optional tip</description>
    <responseparam type="tolerance" default="0.0001" />
<script type="loncapa/python">
computed_response = math.sqrt(math.fsum([math.pow(math.pi,2), math.pow(math.e,2)]))
</script>
    <formulaequationinput />
    <correcthint>Feedback for the correct answer.</correcthint>
    <solution>
      <div class="detailed-solution">
        <p>Explanation</p>
        <p>TEXT OF SOLUTION</p>
      </div>
    </solution>
  </numericalresponse>
</problem>
```

Elements

For numerical input problems, the `<problem>` element can include this hierarchy of child elements.

```
<numericalresponse>
    <label>
    <description>
    <formulaequationinput>
    <additional_answer>
    <correcthint>
    <responseparam>
    <script>
    <solution>
<demandhint>
    <hint>
```

In addition, standard HTML tags can be used to format text.

<numericalresponse>

Required. Indicates that the problem is a numerical input problem.

The `<numericalresponse>` element is similar to the `<formularesponse>` element used by the *math expression input* problem type, but the `<numericalresponse>` element does not allow unspecified variables.

Attributes

Attribute	Description
`answer`	Required. The correct answer to the problem, given as a mathematical expression.
`partial_credit`	Optional. Specifies the type of partial credit given. `close`, `list`, or a combination of both in any order separated by a comma (,).

Note: If you include a variable name preceded with a dollar sign ($) in the problem `answer`, you can include a script in the problem that computes the expression in terms of that variable.

The grader evaluates the answer that you provide and the learner's response in the same way. The grader also automatically simplifies any numeric expressions that you or a learner provides. Answers can include simple expressions such as "0.3" and "42", or more complex expressions such as "1/3" and "sin(pi/5)".

Children

- `<label>`
- `<description>`
- `<formulaequationinput>`
- `<additional_answer>`
- `<responseparam>`
- `<correcthint>`
- `<script>`
- `<solution>`

`<label>`

Required. Identifies the question or prompt. You can include HTML tags within this element.

Attributes

None.

Children

None.

`<description>`

Optional. Provides clarifying information about how to answer the question. You can include HTML tags within this element.

Attributes

None.

Children

None.

<formulaequationinput>

Required. Creates a response field in the LMS where learners enter a response.

Note: Some older problems use a `<textline math="1" />` element instead of `<formulaequationinput>`. However, the `<textline math="1" />` element has been deprecated. All new problems should use the `<formulaequationinput>` element.

Attributes

Attribute	Description
`size`	Optional. Defines the width, in characters, of the response field in the LMS.
`trailing_text`	Optional. Specified text to appear immediately after the response field.

Children

None.

<additional_answer>

Optional. Specifies an additional correct answer for the problem. A problem can contain an unlimited number of additional answers.

Attributes

Attribute	Description
`answer`	Required. The alternative correct answer.

Children

`correcthint`

<responseparam>

Specifies a tolerance, or margin of error, for an answer.

Attributes

Attribute	Description
`type`	Optional. `"tolerance"` defines a tolerance for a number.
`default`	**Optional. A number or a percentage specifying a numerical or percent** tolerance.
`partial_range`	Optional. For partial credit problems of `type="close"`, a multiplier for the tolerance. Default is 2.
`partial_answers`	**Optional. For partial credit problems of `type="list"`, a comma-** separated list of values that are to receive 50% credit.

Children

None.

`<correcthint>`

Optional. Specifies feedback to appear after the learner submits the correct answer.

Attributes

Attribute	Description
`label`	Optional. The text of the custom feedback label.

Children

None.

`<script>`

Optional. Specifies a script that the grader uses to evaluate a learner's response. A problem behaves as if all of the code in all of the `<script>` elements were in a single `<script>` element. Specifically, any variables that are used in multiple `<script>` elements share a namespace and can be overridden.

As with all Python, indentation matters, even though the code is embedded in XML.

Attributes

Attribute	Description
`type`	Required. Must be set to `loncapa/python`.

Children

None.

<solution>

Optional. Identifies the explanation or solution for the problem, or for one of the questions in a problem that contains more than one question.

This element contains an HTML division . The division contains one or more paragraphs <p> of explanatory text.

<demandhint>

Optional. Specifies hints for the learner. For problems that include multiple questions, the hints apply to the entire problem.

Attributes

None.

Children

<hint>

<hint>

Required. Specifies additional information that learners can access if needed.

Attributes

None.

Children

None.

Open Response Assessments

Introduction to Open Response Assessments

In open response assessments (ORA), learners submit essay responses and then go through a series of assessment steps (such as peer assessment and self assessment) to complete the assignment.

Note: ORA assignments cannot be used as the prerequisite when you configure prerequisite course subsections.

Open response assessments that are visible to all learners do not respect cohorts. In other words, it is possible for learners in one cohort to be asked to grade responses for learners in another cohort. If you want to make an open response assessment divided by cohort, you must create that assessment in a course component that is defined as cohort-specific. For more information about cohorts and creating cohort-specific course content, see Using Cohorts in Your Courses and Creating Cohort-Specific Course Content.

The following topics provide conceptual information about open response assessments.

- *Elements of an Open Response Assessment*
- *How Scores for Open Response Assessments Are Calculated*
- *Best Practices for Open Response Assessments*

For information about creating and managing open response assessments, including step by step instructions, see the following sections.

- *Create an Open Response Assessment Assignment*
- *Managing Open Response Assessment Assignments*
- *Accessing Metrics for ORA Assignments*

For information about the learner experience with open response assessments, see Completing Essay Assignments in the *edX Guide for Learners*.

Elements of an Open Response Assessment

When you create an open response assessment assignment, you include several elements.

- One or more *prompts*, or questions, that learners answer.
- A *rubric*. One rubric is used to grade all the prompts in the assessment.
- One or more *assessment steps*. Assignments can include a learner training step, a peer assessment step, a self assessment step, and a staff assessment step.

Note: If you include a learner training step, you must also include a peer assessment step. The learner training step must come first, before the peer and self assessment steps. If you include a staff assessment step, it should be the final step in the assignment.

For step-by-step instructions for creating an open response assessment, see *Create an Open Response Assessment Assignment*.

Prompts

A **prompt** is the question that you want your learners to answer. You can add more than one prompt in an ORA assignment. In addition to requiring a written response, you can require or allow learners to upload an image or other type of file to accompany their written response.

Within each prompt, you can include helpful information for your learners, such as the approximate number of words or sentences that their responses should have, the types of files that they can upload, or what they can expect after they submit their responses. For more information, see *Step 2. Add Prompts*.

In the learner view of the assignment, each prompt appears above the field where learners enter their responses. For more information, see The Steps in an Open Response Assessment.

Rubric

Your assignment must include a **rubric**. Grading for every type of assessment in an ORA assignment (self, peer, or staff) is done by comparing each response against the same rubric. You add one rubric for each problem, regardless of the number of prompts in the problem. The person performing the assessment sees the rubric when she begins grading, and compares the submitted response to the rubric.

A rubric consists of several criteria and a set of options for each criterion.

- **Criteria**. Each criterion describes characteristics that a response should have. Examples are concepts that a response should cover, or the amount of supporting information that a response must include.

  Each criterion has a name and a prompt.

  - The **criterion name** is a one or two word summary of the criterion, such as "Content" or "Organization". This name must be unique within the assignment and cannot be changed after you release the assignment.
  - The **criterion prompt** describes how to evaluate a response based on this criterion.

- **Options**. Each criterion has a set of options, usually a range of ratings, which describe how well each response satisfies the criterion. For example a set of options might be "Fair", "Good", or "Excellent".

  Each option has a name, an explanation, and a point value.

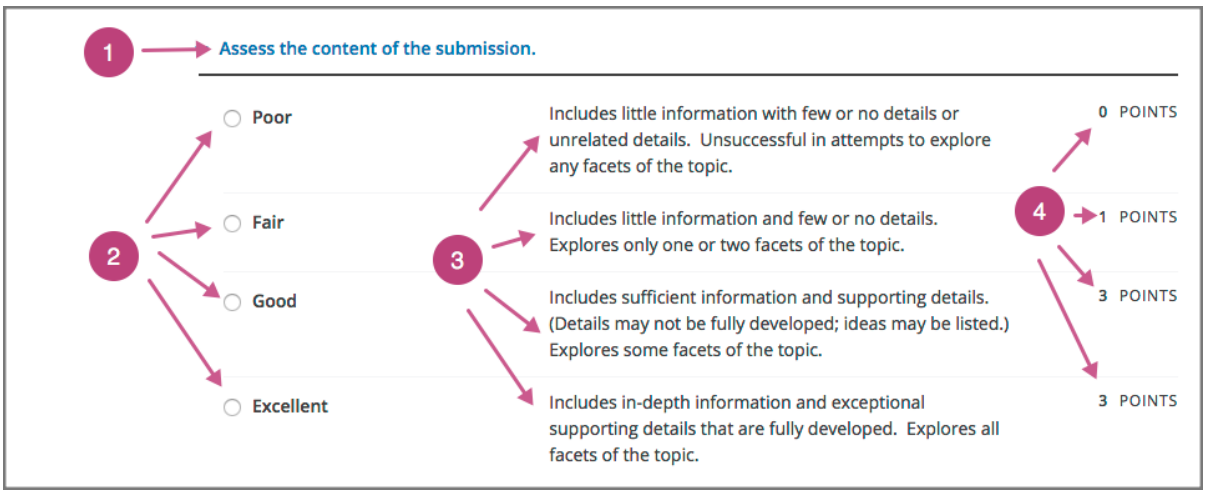
  - The **option name** is a one or two word summary of the rating.
  - The **option explanation** consists of details that help the person performing the assessment to decide whether the response matches the rating. Make sure the explanation for each option is as specific as possible.
  - The **option point value** is the number of grade points given for this option.

Note: Different criteria in the same assignment can have different numbers of options.

You can also include criteria that do not have options, but that do include a field where learners or staff can enter feedback. For more information, see *Provide Only Comment Fields for Individual Criteria*.

In a rubric as it appears to a learner, the following elements are visible.

1. A criterion prompt
2. The names of the criterion's options
3. Descriptions for each option
4. The point value for each option



Criterion names do not display in the rubric that learners use to perform their assessments, but do appear on the page that shows the learner's final ORA assignment grade.

For information about creating a rubric, see *Step 3. Add the Rubric*.

An Example Criterion

In a rubric, one criterion and its set of options might resemble the following.

Criterion

Name: Origins

Prompt: Does this response explain the origins of the Hundred Years' War? (5 points possible)

Options

Points	Name	Explanation
0	Not at all	This response does not address the origins of the Hundred Years' War.
1	Dynastic disagree-ment	This response alludes to a dynastic disagreement between England and France, but doesn't reference Edward III of England and Philip VI of France.
3	Edward and Philip	This response mentions the dynastic disagreement between Edward III and Philip VI, but doesn't address the role of Salic law.
5	Salic law	This response explains the way that Salic law contributed to the dynastic disagreement between Edward III and Philip VI, leading to the Hundred Years' War.

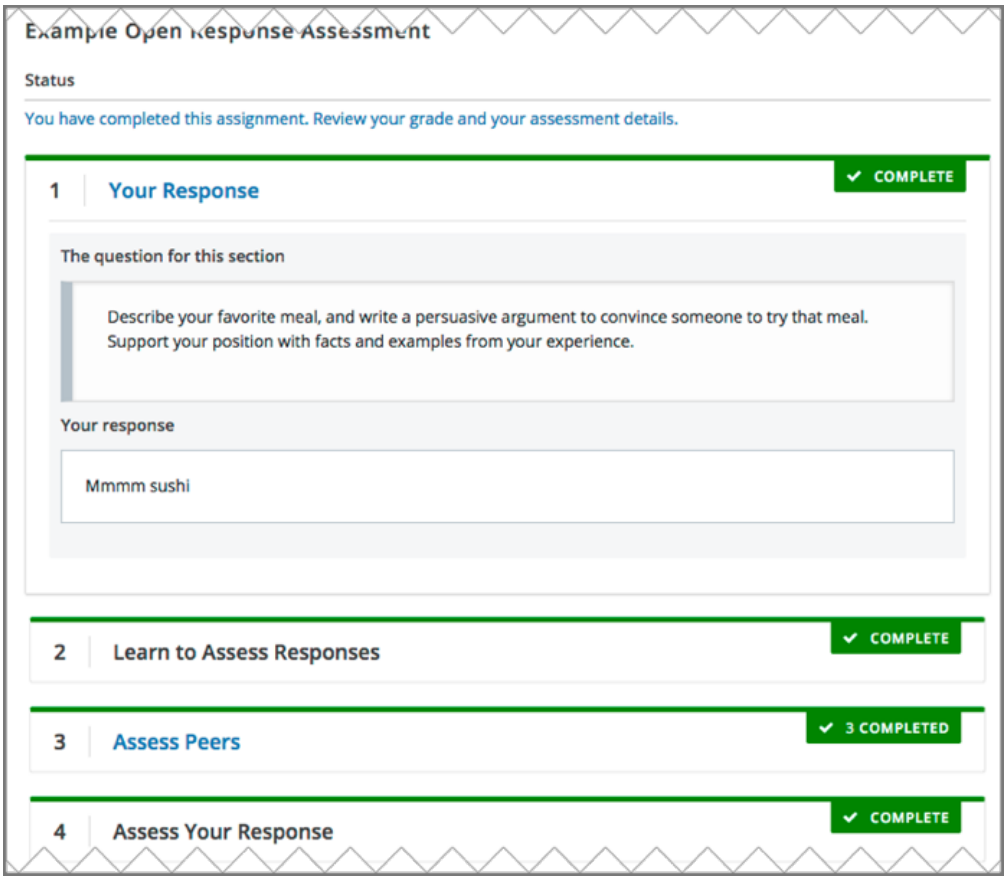
Assessment Steps

In your assignment, you also specify the **assessment steps**. You can set the assignment to include some combination of the following steps.

- *Learner Training Step*
- *Peer Assessment Step*
- *Self Assessment Step*
- *Staff Assessment Step*

Note: If you include a learner training step, you must also include a peer assessment step. The learner training step must come before peer or self assessment steps. If you include both peer and self assessment steps, edX recommends that you place the peer assessment before the self assessment. If you include a staff assessment step, it should be the final step in the assignment.

You can see the type and order of the assessments when you look at the assignment. In the following example, after learners submit their responses, they complete a learner training step ("Learn to Assess Responses"), complete peer assessments on other learners' responses ("Assess Peers"), and then complete a self assessment ("Assess Your Response").



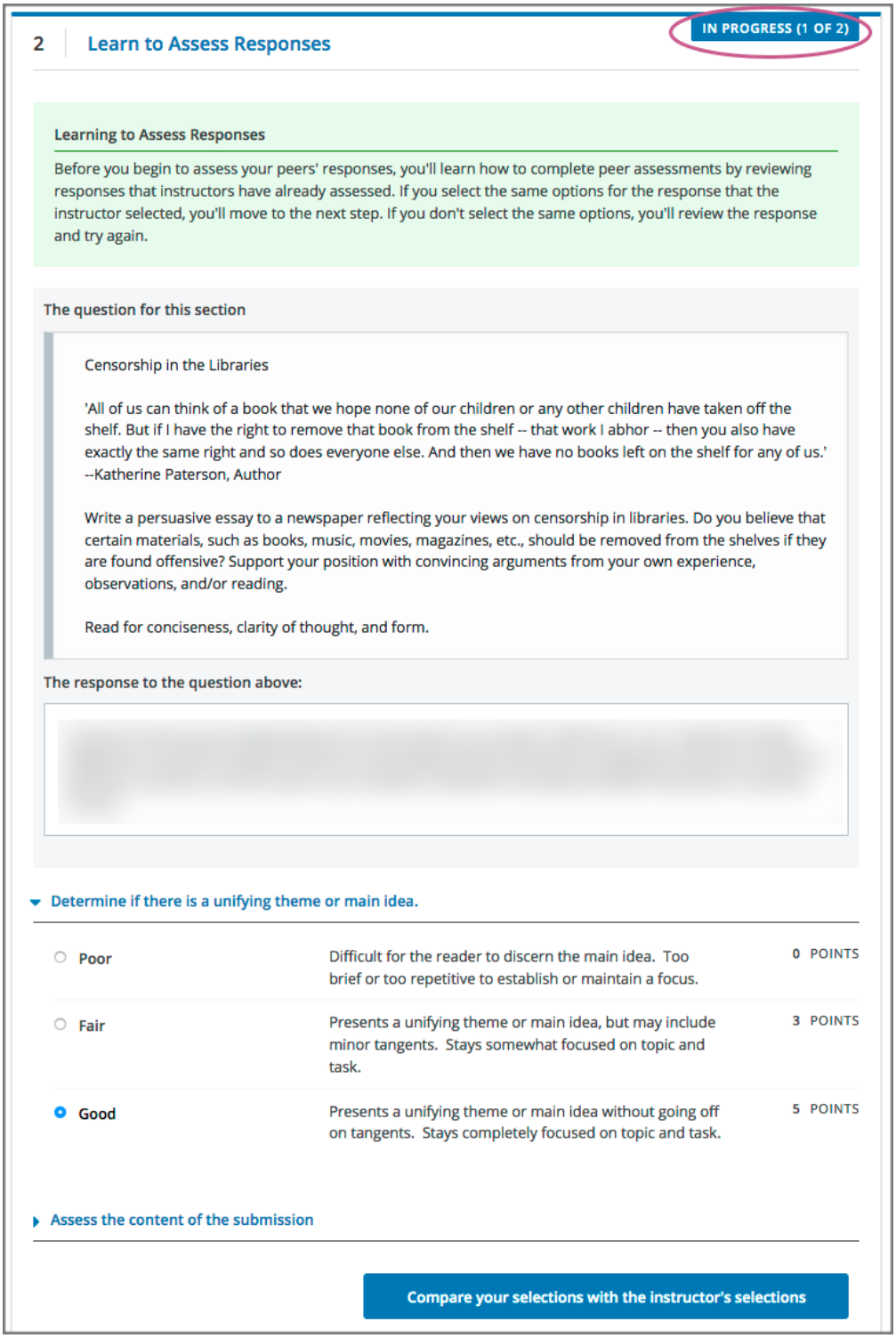
Learner Training Step

Learner training steps teach learners to perform their own assessments. A learner training assessment contains one or more sample responses that you write, together with the scores that you would give the sample responses. Learners review these responses and try to score them the way that you scored them.

Note: If you include a learner training step, you must also include a peer assessment step. The learner training step must come before any peer and self assessment steps.

In a learner training assessment, the **Learn to Assess Responses** step opens immediately after a learner submits a

response. The learner sees one of the sample responses that you created, along with the rubric. The scores that you gave the response do not appear. The learner also sees the number of sample responses that he or she will assess.



The learner selects an option for each of the assignment's criteria, and then selects **Compare your selections with the instructor's selections**. If all of the learner's selections match the selections defined by the course team, the next sample response opens automatically.

If any of the learner's selections differ from those specified by the course team, the learner sees the response again, with a message indicating that his assessment differs from the instructor's assessment.

The learner continues to try scoring the sample response until his scoring for all criteria matches the scoring defined by the course team.

For more information, see *Learner Training*.

Peer Assessment Step

In the peer assessment step, learners review the responses of other learners in the course. For each response, they select an option for each criterion in your rubric based on the response. Learners can also provide text feedback, or comments, on each response.

If you include both peer and self assessment steps, edX recommends that you place the peer assessment before the self assessment.

For information about how peer assessments affect a learner's assignment grade, see *How Scores for Open Response Assessments Are Calculated*.

Number of Responses and Assessments

When you include a peer assessment step, you specify the number of responses that each learner must assess (**Must Grade**) and the number of peer assessments that each response must receive (**Graded By**) before the step is considered complete.

Note: Because some learners might submit a response without performing any peer assessments, some responses might not receive the required number of assessments. To increase the chance that all responses receive a sufficient number of assessments, you must set the number of responses that learners must assess to be higher than the number of assessments that each response must undergo. For example, if you require each response to receive three assessments, you could require each learner to assess five responses.

If all responses have received assessments, but some learners have not completed the required number of peer assessments, those learners can perform peer assessments on responses that have already been assessed by other learners. The learner who submitted the response sees the additional peer assessments when he sees his score. However, the additional peer assessments do not count toward the score that the response receives.

Feedback Options

By default, in peer assessment steps, learners can provide text feedback for the entire response, using a single comment field below the entire rubric. You can also add a comment field to an individual criterion or to several individual criteria. This comment field can contain up to 300 characters.

Comment fields for individual criterion appear below the options for the criterion.

For more information, see *Step 3. Add the Rubric* and *Provide Only Comment Fields for Individual Criteria*.

Assessing Additional Responses

Learners can assess more than the required number of responses. After a learner completes the peer assessment step, the step "collapses" so that only the **Assess Peers** heading is visible.

If the learner selects the **Assess Peers** heading, the step expands again. The learner can then select **Continue Assessing Peers** to perform additional peer assessments.

Self Assessment Step

In self assessment steps, the learner sees her own response followed by the rubric. As with peer assessments, the learner evaluates the response using the rubric, selecting an option for each criterion.

If you include both peer and self assessments, edX recommends that you include the peer assessment before the self assessment.

Staff Assessment Step

In staff assessment steps, a member of the course team performs an evaluation of the learner's response. Course team members grade the response using the problem's rubric, in the same way that self and peer assessments are done, and can include comments in their assessment.

Note: If a staff assessment step is included in an assignment, learners do not receive final grades until the staff assessment step has been completed. The scores that you give learners in staff assessment steps override scores from any other assessment type in the assignment, including peer assessments that are completed after the staff assessment.

Including a staff assessment step in an ORA assignment is best for courses with smaller groups of learners. For example, in a course with cohorts, you might create an ORA assignment that has both peer assessment and staff assessment steps, and make it available only to the members of one or more specific cohorts. For the members of the remaining cohorts, you create an ORA assignment that has only the peer assessment step. For details about creating different course experiences for learners in different cohorts, see Creating Cohort-Specific Course Content.

For details about performing grading in staff assessment steps, see *Perform Staff Assessments in an ORA Assignment*.

How Scores for Open Response Assessments Are Calculated

In open response assessments that contain staff assessments, staff assessments can be performed more than once, and the most recent staff assessment score is equivalent to the assignment's final score. Peer and self assessment scores are not taken into account, although learners can see scores and comments from all assessments that were performed on their response.

In open response assessments that do not contain staff assessments but do contain both peer assessment and self assessments, only the peer assessment score counts toward the assignment's final score. The self assessment score is not taken into account. There is no option for weighting the peer and self assessment portions independently.

In open response assessments that include only self assessments, the assignment's final score is equivalent to the self assessment score.

Note: Given the high level of subjectivity in peer assessments, edX recommends that you make ORA assignments count towards only a small percentage of a course's final grade.

The following topics detail how the scores for peer assessments and self assessments are calculated.

Peer Assessment Scoring

Note: If an open response assessment includes peer and self assessments but not staff assessments, only the peer assessment score counts towards the assignment's final score. The self assessment score is not taken into account.

Peer assessments are scored by criteria. A number of peer assessors rate a learner's response by each of the required criteria. The learner's score for a particular criterion is the median of all scores that each peer assessor gave that criterion. For example, if the Ideas criterion in a peer assessment receives a 10 from one learner, a 7 from a second learner, and an 8 from a third learner, the Ideas criterion's score is 8.

The learner's final score on a response is the sum of the median scores from all peer assessors for all of the required criteria.

For example, a response might have received the following scores from peer assessors.

Criterion Name	Peer 1	Peer 2	Peer 3	Median
Ideas (out of 10)	10	7	8	**8**
Content (out of 10)	7	9	8	**8**
Grammar (out of 5)	4	4	5	**4**

To calculate the final score for the response, add the median scores that were given for each criterion, as follows.

Ideas median (8/10) + Content median (8/10) + Grammar median (4/5) = final score (20/25)

Note: Remember that final scores are calculated by criteria, not by individual assessor. Therefore, the score for the response is not the median of the scores that each individual peer assessor gave the response.

For information on scores for learner submissions that you have canceled and removed from peer assessment, refer to *Remove a Learner's Response*.

Self Assessment Scoring

Note: If an open response assessment includes both peer and self assessments, the self assessment score does not count toward the final grade.

If an open response assessment includes only self assessments, the assignment's final score is equivalent to the self assessment score.

Self assessments are scored by criteria. Each learner rates herself on each criterion, using the rubric. The learner's final score on a response is the total number of earned points, out of the total possible points.

Staff Assessment Scoring

If an open response assessment includes a staff assessment step, the score that is given in the staff assessment step overrides all other scores in the assignment.

Top Responses

You can include a **Top Responses** section that shows the top scoring responses that learners have submitted for the assignment, along with the scores for those responses. The **Top Responses** section appears below the learner's score

information after the learner finishes every step in the assignment.

You can allow the **Top Responses** section to show between 1 and 100 responses. Keep in mind, however, that each response might be up to 300 pixels in height in the list. (For longer responses, learners can scroll to see the entire response.) EdX recommends that you specify 20 or fewer responses to prevent the page from becoming too long.

Note: It can take up to an hour for a high-scoring response to appear in the **Top Responses** list.

If a high-scoring response is *removed from peer assessment* it is also removed from the **Top Responses** list.

For more information, see *Step 7. Show Top Responses*.

Best Practices for Open Response Assessments

Open response assessments can be a powerful teaching tool, but they are more effective in some situations than in others. In general, open response assessments are best suited to open-ended or project-based assignments with subjective essay answers and discussion. For example, open response assessments work well in humanities assignments where learners are encouraged to make subjective assessments of text, images, or other contributions, but they might not be the ideal tool in chemistry assignments where there are definitively correct or incorrect answers to questions.

Note: Do not add more than one ORA component in a course unit. Multiple ORA assignments in a unit cause errors

when learners submit their assessments.

EdX suggests that you follow the guidelines and best practices in the following sections when you use open response assessments in your courses.

Designing the Assignment

- Do not add more than one ORA component in a course unit. Multiple ORA assignments in a unit cause errors when learners submit their assessments.
- Do not include too many ORA assessments in your course. *Peer assessments* are hard work for learners, and having to perform too many peer assessments can have a negative impact on learners' course completion rates.
- For a manageable experience for course staff, use staff assessment steps only in assignments that are available to a limited number of learners. For example, in courses that have cohorts enabled, make the assignment containing the staff assessment step available only to members of one or more cohorts.

Grading and Rubrics

- Make sure you have a well designed and clear *rubric* for the assignment. A good rubric is very important in helping to eliminate ambiguity in the peer grading process.
- Make ORA assignments count toward only a small percentage of the final course grade, or make them ungraded.
- In graded ORA assignments, consider setting the lowest possible score to a number higher than zero, so that learners can earn some credit for the work they have done, even if their peer assessors give them low grades.
- Provide an ungraded practice ORA assignment prior to the first graded ORA assignment in the course, so that learners can understand the peer grading process and get the most out of the eventual graded ORA assignment.
- Consider using ungraded ORA assignments to generate learner interaction and feedback without affecting grades.

Peer Assessments

- Set the **Must Grade** number higher than the **Graded By** number to minimize the chance that some responses will not be peer assessed. EdX recommends a setting such as **Must Grade** = 4 and **Graded By** = 3.
- To allow enough time for peer assessments to be performed after learners have submitted their own responses, set the response due date and time at least one week before the peer assessment due date and time.

  If the response due time and peer assessment due time are too close together, and a learner submits a response just before responses are due, other learners may not have time to perform peer assessments before peer assessments are due.
- Use course discussion posts to provide guidance for peer grading of ORA assignments.
- Consider extending due dates to allow the discussion moderation team to monitor course discussions for questions about, or reactions to, peer grading, and to address issues when necessary.

  If learners raise concerns about ORA assignments in course discussions, course team members can perform actions such as deleting a learner's history, or "state" for a problem so that he can submit his assignment again, *overriding a learner's grade*, or *removing a learner response* from peer grading. If there are more widespread issues with peer grading, the course team can reduce the weight of the peer assessment within the final course grade or allow learners to drop the lowest graded assignment from their grades.

Asking Learners to Upload Files in Responses

In ORA assignments, you can ask your learners to upload an image, a .pdf file, or a file of another type as a part of their responses. Other learners evaluate the responses and their accompanying files during the peer assessment. Offering the option to upload a file in addition to a text response can give learners the opportunity to use tools and develop skills that are relevant to your course.

Before you decide to ask learners to upload other files along with their text responses, however, be aware of the following limitations and best practices.

- During the peer assessment step, learners download the files that other learners uploaded. To reduce the potential for problems from files with malicious content, learners cannot upload files with certain file extensions. For a complete list, see *Prohibited File Extensions*.
- Course teams can only access uploaded files for one learner at a time. Uploaded file content is not included in the reports of answer submissions that are available from the instructor dashboard, and course data packages do not include any of the uploaded files.
- You cannot require your learners to upload files. You can only give them the option to do so.
- All responses must include some response text. Learners cannot submit a response that contains only an uploaded file.
- Learners can submit only one file with each response.
- Files must be smaller than 5MB in size.
- Image files must be in .jpg, .gif, or .png format.

For more information, see *Allow Learners to Submit Files (optional)*.

Prohibited File Extensions

This topic lists the file extensions for the set of file types that learners are prohibited from uploading as part of an open response assessment on edx.org or edX Edge. When you define a set of custom file types for learners to upload with their responses, you cannot specify these file types. The extensions on this list are selected and maintained by the development operations team at edX, and are subject to change.

A through I	.action, .apk, .app, .application, .bat, .bin, .cmd, .com, .command, .cpl, .csh, .dmg, .exe, .gadget, .hta, .inf, .ins, .inx, .ipa, .isu
J through P	.jar, .job, .jse, .lnk., msc, .msh, .msh1, .msh2, .mshxml, .msh1xml, .msh2xml, .msi, .msp, .mst, .osx, .out, .paf, .pif, .prg, psc1, .psc2, .ps1, .ps1xml, .ps2, .ps2xml
Q through Z	.reg, .rgs, .run, .scf, .scr, .sct, .shb, .shs, .u3p, .vb, .vbe, .vbs, .vbscript, .workflow .ws, .wsc, .wsf, .wsh

Create an Open Response Assessment Assignment

Creating an *open response assessment (ORA) assignment* is a multi-step process. This section covers each step in detail.

In addition, see these other topics about different aspects of open response assessments.

- Components of an open response assessment: *Introduction to Open Response Assessments*
- Viewing metrics and learner responses for released open response assessments: *Accessing Metrics for ORA Assignments*
- Actions you can take for an active ORA assignment: *Managing Open Response Assessment Assignments*

Step 1. Add the Component

To add the open response assessment component to your course, complete these steps.

Note: Do not add more than one ORA component in a course unit. Multiple ORA assignments in a unit cause errors when learners submit their assessments.

---

1. In Studio, open the unit where you want to create the open response assessment.
2. Under **Add New Component**, select **Problem**.
3. Select **Advanced**, and then select **Peer Assessment**.
4. In the problem component that appears, select **Edit**.

   You use this component editor to add prompts and the rubric, and to specify other settings for the open response assessment component.
5. Select **Save** each time you complete an editing session. You can continue to edit the problem until you publish the unit.

Note: After you publish an ORA assignment, you can no longer change the structure of the rubric or the point values associated with each criterion in the rubric. If you correct typographical errors in the text of the rubric, only learners who have not yet started the assignment will see the corrections. However, you can modify due dates and the weight of the ORA assignment after you publish an ORA assignment.

Step 2. Add Prompts

To add *prompts*, or questions, to your ORA assignment, complete these steps.

Note: If you want to add text formatting to the prompt, or include an image, see *Add Formatting or Images to a Prompt*.

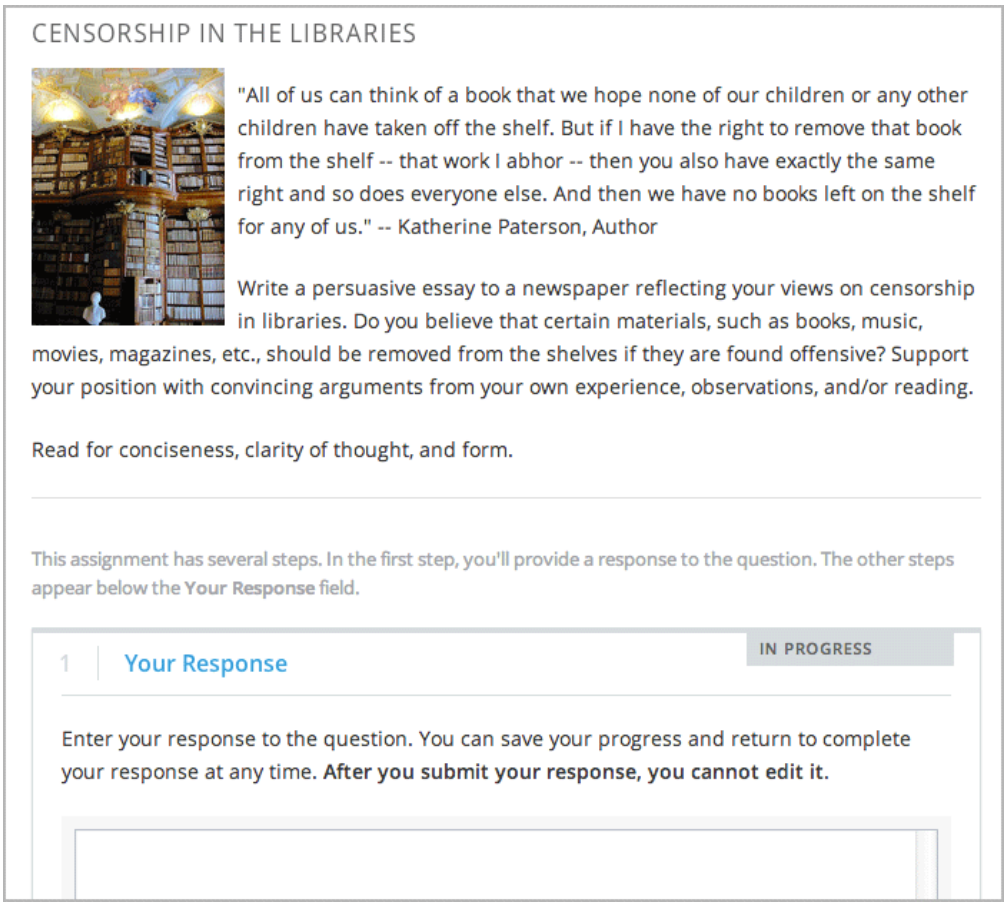
---

1. In the open response assessment component editor, select **Prompt**.

2. Replace the example prompt in the text field with your prompt.
3. To add another prompt in the assignment, select **Add a Prompt**, and then repeat Step 2.

Add Formatting or Images to a Prompt

Currently, you cannot format text or add images inside the Peer Assessment component. To include formatting or images in a prompt, you can add an HTML component that contains your text above the Peer Assessment component, and leave the text field in the **Prompt** tab blank. The instructions for the peer assessment still appear above the **Your Response** field.



Allow Learners to Submit Files (optional)

Before you enable this feature for your open response assessment, be sure to read about its limitations and best practices. For more information, see *Asking Learners to Upload Files in Responses*.

To allow learners to submit a file along with their text responses, follow these steps.

1. In the ORA component editor, select **Settings**.
2. Set **Allow File Upload** to one of these options.
    - **Image File**
    - **PDF or Image File**
    - **Custom File Types**

3. If you select **Custom File Types**, the **File Types** field appears. Enter the file extensions, separated by commas, of the types of files that you want learners to submit.

   Note: To reduce the potential for problems from files with malicious content, learners cannot upload certain file types. For more information, see *Prohibited File Extensions*.

4. Make sure the text of your prompt includes adequate instructions for learners to upload the required files, including the file type or types that learners can upload.

Step 3. Add the Rubric

In this step, you add your *rubric* to provide guidance for assessing responses within the assignment. You add one rubric for each problem, regardless of the number of prompts in the problem.

Note: The most effective rubrics for peer grading are written in clear, simple language, have concrete details, and are as specific as possible. Many novice learners will find it diffidult to make the types of value judgments required by more holistic rubrics.

For each step below, replace any default text with your own text.

Note: All open response assessments include a feedback field below the rubric so that learners can provide written feedback on a peer's overall response. You can also allow or require learners to provide feedback for individual criteria. See step 4 in the following procedure for instructions. For more information, see *Feedback Options*.

To add the rubric, follow these steps.

1. In the ORA component editor, select the **Rubric** tab.
2. In the first **Criterion** section, enter the name and prompt text of your first criterion.
3. In the **Option** sections for this criterion, for each option that you provide for the criterion enter a name, explanation, and point value.

   To remove options, select **Remove** at the top right of the option section.

   To add more options, select **Add Option**.
4. Next to **Feedback for This Criterion**, select a value in the dropdown list.
   - If you do not want to allow feedback for this individual criterion, select **None**.
   - To require feedback for this criterion, select **Required**.
   - To allow feedback, but not require it, select **Optional**.
5. Repeat steps 2-4 to create additional criteria. To add more criteria than provided for in the template, select **Add Criterion** at the end of the list of criteria.
6. Under **Feedback for This Response**, add instructions for learners to provide overall written feedback on responses that they assess. You can leave the default text in the **Feedback Instructions** and **Default Feedback Text** fields, or replace it with your own text.

Note: After you publish an ORA assignment, you can no longer change the structure of the rubric or the point values associated with each criterion in the rubric. If you correct typographical errors in the text of the rubric, only learners

who have not yet started the assignment will see the corrections. However, you can modify due dates and the weight of the ORA assignment after you publish an ORA assignment.

Provide Only Comment Fields for Individual Criteria

For an individual criterion, you can omit options, but if you do not include options, you must include the ability to add feedback comments.

To provide a comment field without options, complete these steps.

1. In the ORA component editor, select the **Rubric** tab.
2. In the **Criterion** section for the criterion that you want to only provide a comment field for, select **Remove** to remove each option.
3. Next to **Feedback for This Criterion**, select **Required** from the list.

Step 4. Specify the Assignment Name and Response Dates

Before you specify the start and due dates and times for a response, be sure that you consider these aspects of, and best practices for, the open response assessment feature. For more information, see *Best Practices for Open Response Assessments*.

- Unlike other problem types, ORA assignments are not governed by the subsection due date. You set due dates for each ORA assignment in the assignment's settings.
- The grace period that you can set for the course does not apply to ORA assignments.
- Allow sufficient time for peer assessments to be performed after learners have submitted their own responses. EdX recommends that you allow at least one week between the due date for responses and the due date for peer assessments. If the response due time and peer assessment due time are close together, and a learner submits a response just before responses are due, other learners may not have time to perform peer assessments before peer assessments are due.
- The times that you set are in Coordinated Universal Time (UTC). To verify that you have specified the times that you intend, use a time zone converter such as Time and Date Time Zone Converter.

To specify a name for the assignment as well as start and due dates for all learner responses, complete these steps.

1. In the ORA component editor, select the **Settings** tab.
2. Next to **Display Name**, enter the name you want to give the assignment.
3. Next to **Response Start Date** and **Response Start Time**, enter the date and time when you want learners to be able to begin submitting responses.
4. Next to **Response Due Date** and **Response Due Time**, enter the date and time by which all learner responses must be submitted.

Step 5. Select Assignment Steps

Open response assessment assignments can include learner training, peer assessment, self assessment, and staff assessment steps.

The component editor provides the *steps* in a sequence that works well for most courses. While you can change the order of the peer, self, and staff assessment steps, edX recommends that you include them in this order.

Note: If you include a learner training step, you must also include a peer assessment step. The learner training step must come before peer or self assessment steps.

If you include both peer and self assessment steps, edX recommends that you place the peer assessment before the self assessment.

If you include a staff assessment step, it should be the final step in the assignment.

To add steps to the open response assignment, complete these actions.

1. In the ORA component editor, select the **Settings** tab.
2. Scroll below the **Top Responses** field.
3. Locate the following headings.
    - **Step: Learner Training**
    - **Step: Peer Assessment**
    - **Step: Self Assessment**
    - **Step: Staff Assessment**

    Select the check boxes for the steps that you want the assignment to include.
4. (optional) To change the order of the steps, drag the steps into the order that you want.

Step 6. Specify Step Settings

After you select the steps that you want, you can specify settings for those steps.

Note: If you make changes to a step, and then clear the check box for that step, the step will no longer be part of the assignment and your changes will not be saved.

Learner Training

For the *learner training step*, you enter one or more example responses that you have created, then specify the expected option for each criterion in your rubric.

Note: You must enter your complete rubric on the **Rubric** tab before you can select options for the learner training responses. If you later change one of your criteria or any of its options, you must also update the learner training step.

To add and score learner training responses, follow these steps.

1. Under **Step: Learner Training**, locate the first **Scored Response** section.
2. In the **Response** field, enter the text of your example response.
3. Under **Response Score**, for each criterion, select the option that you want.

Peer Assessment

For the *peer assessment step*, you specify the number of responses that each learner must grade, the number of learners who must grade each response, and start and due dates. All fields are required.

To specify peer assessment settings, follow these steps.

1. Locate the **Step: Peer Assessment** heading.
2. Next to **Must Grade**, enter the number of responses that each learner must grade.
3. Next to **Graded By**, enter the number of learners that must grade each response.
4. Next to **Start Date** and **Start Time**, enter the date and time when learners can begin assessing their peers' responses.
5. Next to **Due Date** and **Due Time**, enter the date and time by which all peer assessments must be completed.

Note: The times that you set are in Coordinated Universal Time (UTC). To verify that you have specified the times that you intend, use a time zone converter such as Time and Date Time Zone Converter.

For more information about peer assessment steps, see *Peer Assessment Step*.

Self Assessment

For the *self assessment step*, you specify when the step starts and ends.

1. Locate the **Step: Self Assessment** heading.
2. Next to **Start Date** and **Start Time**, enter the date and time when learners can begin assessing their peers' responses.
3. Next to **Due Date** and **Due Time**, enter the date and time by which all peer assessments must be complete.

Note: The times that you set, and the times that learners see, use Coordinated Universal Time (UTC). You might want to verify that you have specified the times that you intend by using a time zone converter such as Time and Date Time Zone Converter.

Staff Assessment

For the *staff assessment step*, there are no additional settings to specify after you have selected the step for inclusion in the assignment.

Step 7. Show Top Responses

You can specify whether learners see a section that shows the *highest scoring responses* that were submitted for each question in the assignment. If offered, this section displays only after each learner has completed all steps in the assignment. You specify the number of highest scoring responses to show.

Note: Because each response can be up to 300 pixels in height, we recommend that you set the number of top responses lower than 20, to prevent the page from becoming too long.

---

1. In the ORA component editor, select the **Settings** tab.
2. In the **Top Responses** field, specify the number of responses that you want to appear in the **Top Responses** section below the learner's final score.

   If you do not want this section to appear, set the number to 0. The maximum number is 100.

Step 8. Test the Assignment

To test your ORA assignment, you can set up the assignment in your course, set the section or subsection date in the future, publish the unit, and ask one or more beta testers to submit responses and grade each other. The beta testers can then let you know if they found the question and the rubric easy to understand or if they had any problems with the assignment.

For more information about beta testing, see Beta Testing a Course.

Managing Open Response Assessment Assignments

After you publish an open response assessment (ORA) assignment and learners start to submit responses and perform assessments, members of the course team can take the following actions.

The following topics provide additional information about open response assessments.

View a Specific Learner's Response and Assessments

You can view the following information about an individual learner's open response assessment assignment.

- The text of the learner's response, including any files that the learner uploaded.
- The peer assessments that other learners performed on the learner's response, including feedback on individual criteria and on the overall response.
- The peer assessments that the learner performed on other learners' responses, including feedback on individual criteria and on the overall responses.
- The learner's self assessment.
- The learner's grade for the assignment.
- An override grade provided by course staff.

For more details about accessing information for a specific learner, and for an example that shows a learner's response with peer assessments, see *Access a Specific Learner's Information*.

To determine whether a learner has received the required number of assessments from other learners and has completed the required number of assessments for other learners, refer to the **Graded By** and **Must Grade** values that were set for the open response assessment assignment in Studio. For more information about these settings, see *Specify Step Settings*.

Access a Specific Learner's Information

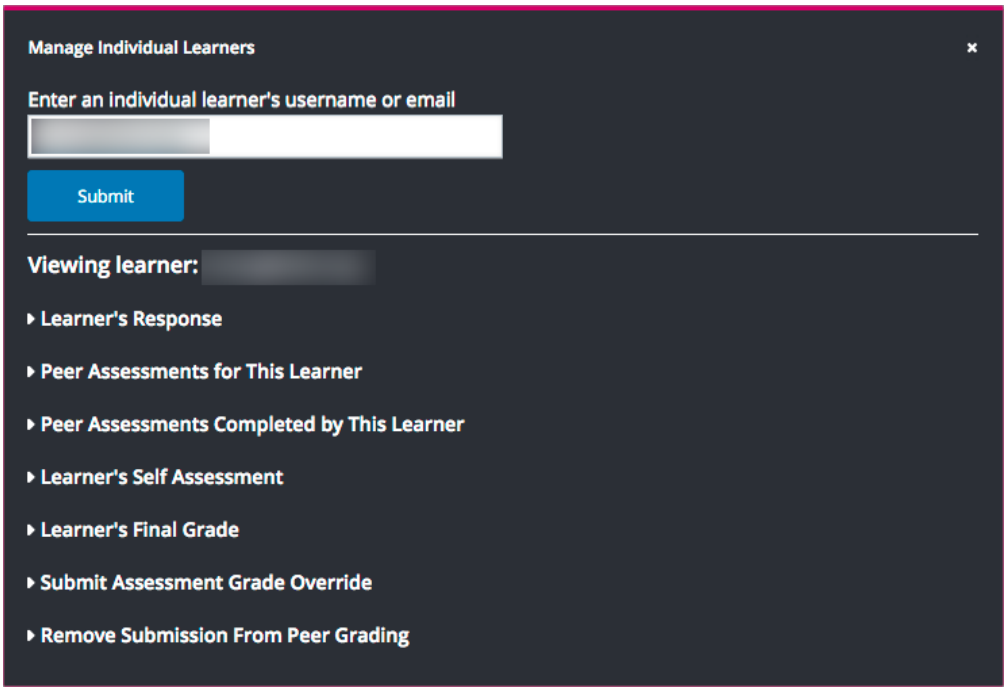
In order to access information about a specific learner's assignment, you need that learner's username or email address. For more information, see Download or View Learner Data.

To access information about a specific learner, follow these steps.

1. View the live version of your course in the LMS, and then go to the ORA assignment.
2. Scroll to the end of the problem, and then select **Manage Individual Learners**.
3. Enter the learner's username or email address, and then select **Submit**.

   The **Manage Individual Learners** dialog updates with expandable sections for each of the assessment steps in the assignment and other actions you can take on the learner's response. Only the types of assessment steps (self, peer, or staff) that are included in the assignment are shown.

   If the learner uploaded a file along with her response, select **View the file associated with this submission** to review or download it.
4. Select any of the section headings to expand that section.



Perform Staff Assessments in an ORA Assignment

When a staff assessment is included in an open response assessment assignment, course team members see a **Grade Available Responses** option at the end of the assignment in the course, and learners see a **Staff Assessment** step in

their assignment. For information about the possible assessment steps in an ORA assignment, see *Assessment Steps*.

Submitting a staff assessment has the following results.

- The score that you give a learner in a staff assessment overrides scores from any other assessment type in the assignment.
- Peer assessments that are completed before or after your staff assessment have no effect on the learner's final assignment grade.

To perform a staff assessment in an assignment, follow these steps.

1. View the live version of your course in the LMS, and then go to the ORA assignment.
2. Scroll to the end of the problem, and then select **Grade Available Responses**.

   In the dialog that opens, the number of available and checked out responses is shown. Checked out responses are responses that are currently being graded by you or another course team member.
3. Select the **Staff Assessment** heading to open a response that is available for grading.
4. Perform an evaluation of the response using the problem's rubric.
5. Select **Submit assessment** to submit the assessment and close the grading dialog. Alternatively, select **Submit assessment and continue grading** to submit the assessment and immediately grade another submission.

Override a Learner's Assessment Grade

For any open response assessment, whether or not a staff assessment is already included, you can override a learner's final grade for the assignment. The ability to override the final grade can be useful if, for example, a learner's submission was inappropriately or inadequately graded by peers, or if there are not enough peer reviewers to complete the required number of peer assessments.

Submitting an override assessment has the following results.

- The score that you give a learner in an override assessment overrides scores from any other assessment type in the assignment.
- Any steps that the learner did not complete for the assignment are marked as complete.
- Peer assessments that are completed before or after your staff assessment have no effect on the learner's final assignment grade.

Note: You can perform override assessments more than once on the same response, regardless of the due date of the assignment. The learner's final grade on the assessment is updated to reflect the most recent staff override assessment grade.

Learners who receive override grades for their submissions see a **Staff Assessment** step in their assignments, where they can view the rubric and any comments provided in the staff assessments.

Perform an Override Assessment

In order to perform an override assessment for a learner, you need that learner's username or email address. For more information, see Download or View Learner Data.

To perform an override assessment, follow these steps.

1. View the live version of your course in the LMS, and then go to the ORA assignment.
2. Scroll to the end of the problem, then select **Manage Individual Learners**.

3. Enter the learner's username or email, then select **Submit**.

   The **Manage Individual Learners** dialog updates with expandable sections for each of the assessment steps in the assignment and other actions you can take on the learner's response.

4. Select **Submit Assessment Grade Override**.

5. Perform an assessment of the learner's response using the problem's rubric.

6. When you have finished the assessment, select **Submit assessment**.

   The grade that you have given this learner's response becomes the learner's final grade on the assignment. Peer assessments are not taken into account in calculating the learner's final assignment grade when a staff override grade exists.

   Learners who have an override grade for their submission see a **Staff Assessment** step in their assignment, where they can view the rubric and any comments provided in the staff assessment.

Note: Override assessments can be performed more than once on the same response, regardless of the due date of the assignment. The learner's final grade on the assessment is updated to reflect the most recent staff override assessment grade.

Remove a Learner's Response

In a course that contains assignments with peer assessment steps, learners might alert you to inappropriate responses that they have seen while performing peer assessments. In such a situation you can *locate* and remove the response. Doing so removes the response so that it is no longer shown to other learners for peer assessment.

Note: Removing a learner's response is an irreversible action.

When you remove a response, the response is immediately taken out of the pool of submissions available for peer assessment. If the inappropriate response has already been sent to other learners for peer assessment, it is also removed from their queues. However, if any learner has already graded the inappropriate response, it is counted as one of the submissions they have graded.

Note: After you remove an inappropriate response, you can decide whether the learner who submitted that response is allowed to submit a replacement response.

If you do not want to allow the learner to submit a replacement response, you do not need to take any additional action. The learner receives a grade of zero for the entire submission.

To allow the learner to resubmit a response for a cancelled submission, you must delete the learner's state for the problem.

To remove a submitted response, follow these steps.

1. Identify the learner who submitted the inappropriate response by following the steps in the *Locate a Specific Submission in an ORA Assignment* topic.

2. View the live version of your course in the LMS, and then go to the ORA assignment that contains the submission you want to remove.

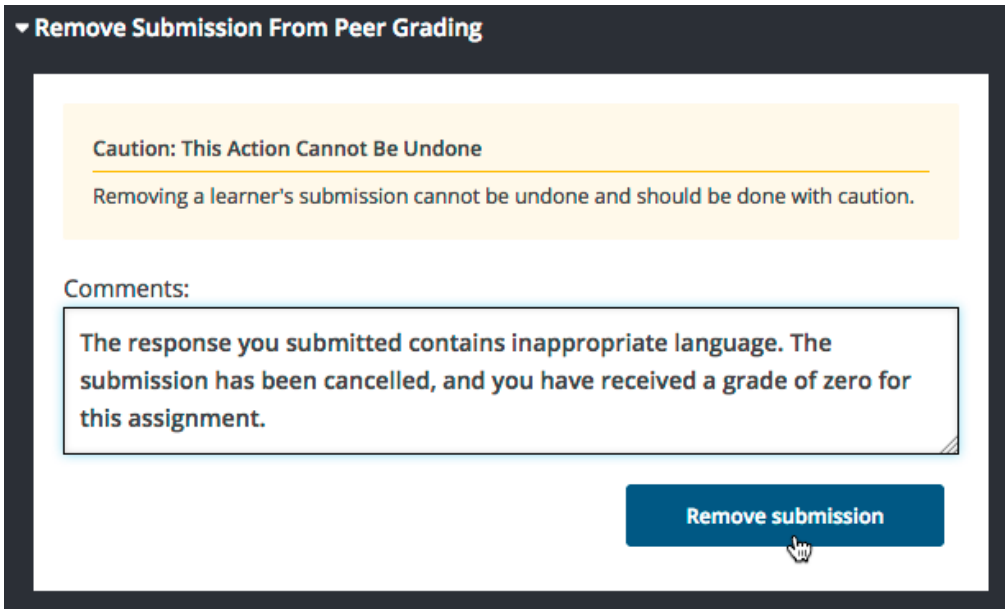
3. Scroll to the end of the problem, and then select **Manage Individual Learners**.

4. Enter the learner's username or email, and then select **Submit**.

   The **Manage Individual Learners** dialog updates with expandable sections for each of the assessment steps in the assignment and other actions you can take on the learner's response.

5. Select **Remove Submission from Peer Grading**.

6. Enter a comment to explain the removal. The learner sees this comment when she views her response in the open response assessment problem.

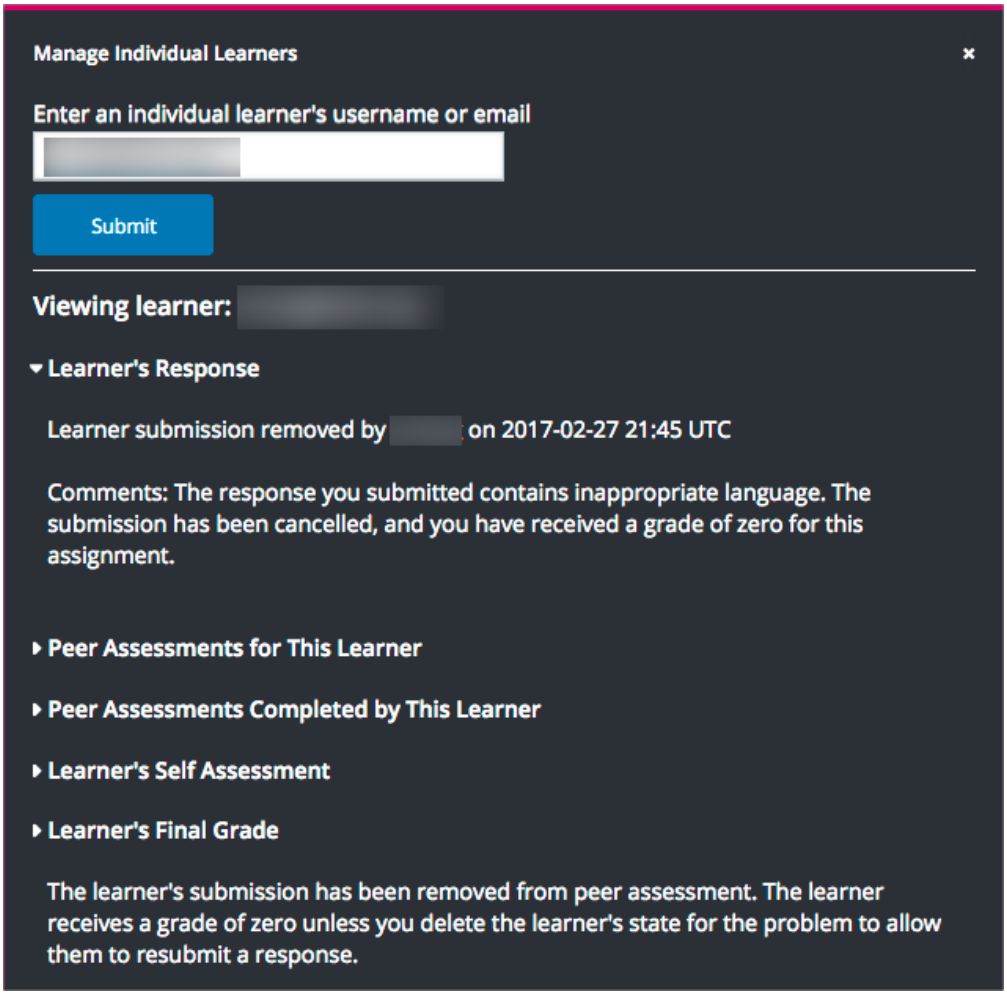


7. Select **Remove submission**.

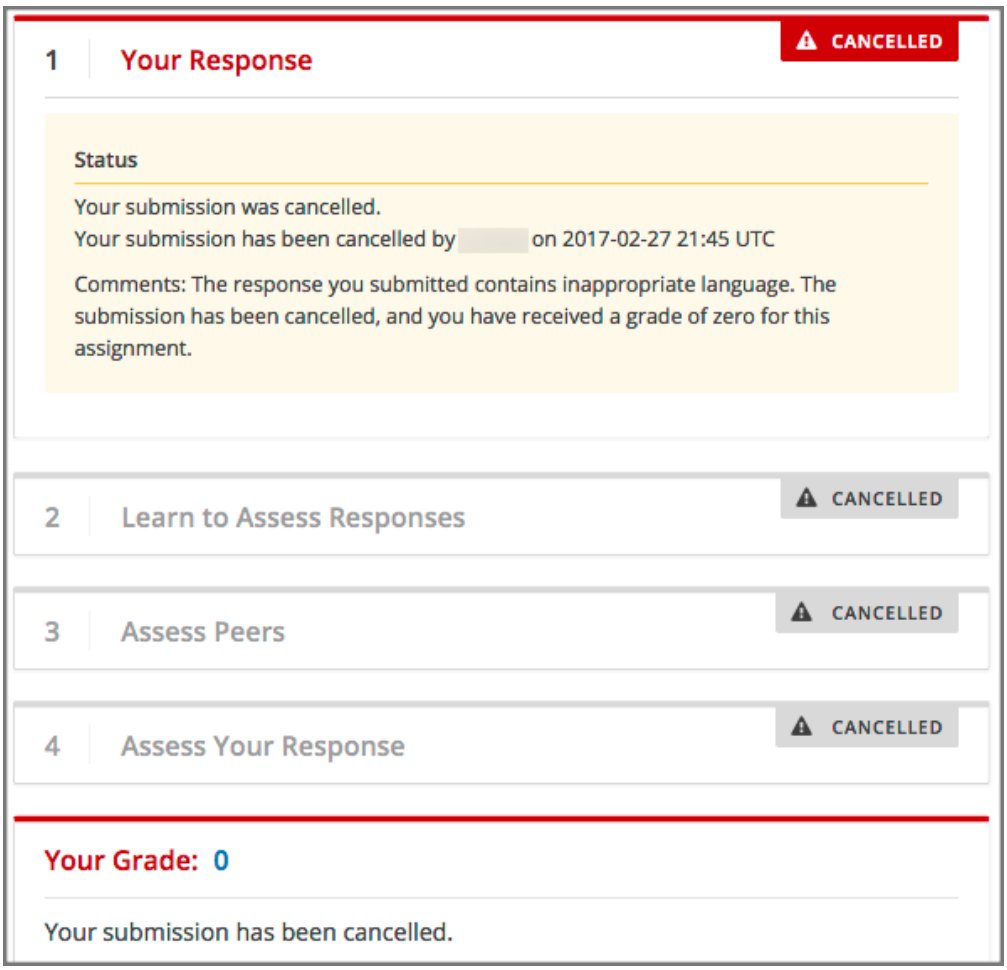
   The inappropriate submission is permanently removed from peer assessment. Removed submissions are also removed from the list of Top Responses if they were previously listed.

8. Optionally, delete the learner's state for the problem. This step allows the learner to submit another response. For more information, see Delete a Learner's State for a Problem.

When you access *this learner's information* again by selecting **Manage Individual Learners**, instead of the response, you see a note showing the date and time that the submission was removed, and the comments that you entered.



When the learner views the assignment in the course, she sees that all steps in the assignment have a status of "Cancelled". Under **Your Response**, instead of the text of their response, she sees the date and time that their response was cancelled, and the comments relating to the removal of their submission.



Locate a Specific Submission in an ORA Assignment

If you are alerted to an inappropriate ORA submission that you want to cancel and *remove from peer assessment*, locate the specific submission by following these steps.

1. Ask the person who reported the incident to send you a sample of text from the inappropriate response.
2. *Generate an ORA data report*.
3. Search the report for text that matches the sample text from the inappropriate response.
4. From any matching entries in the spreadsheet, locate the username of the learner who posted the submission.
5. Make a note of the username, and follow the steps to *remove a learner response from peer grading*.

Accessing Metrics for ORA Assignments

After you release an open response assessment assignment, you can access various metrics for the assignment or for all open response assessment assignments in the course. For example, you can view the number of learners in each step of the assignment or in possible states such as "Waiting" or "Completed" within the assignment. In addition to viewing metrics for the assignment or for assignments in a course, you can also access assignment details for an individual learner, or *generate a report* containing learner and response details for ORA assignments in the course.

For information about tasks that you can perform on learner responses in an ORA assignment, including *performing a grade override assessment* or *cancelling a learner's submission*, see *Managing Open Response Assessment Assignments*.

View Statistics for All ORA Assignments in a Course

To view metrics about all of the ORA assignments in a course, follow these steps.

1. View the live version of the course.
2. Select **Instructor** to open the instructor dashboard.
3. On the instructor dashboard, select **Open Responses**.

The **Open Responses** tab of the instructor dashboard displays the following information.

- The number of course units that include an ORA assignment.
- The number of ORA assignments in the course.
- The total number of submitted responses.
- The number of learners who are in each of the training, peer, self, and staff workflow states.
- The number of learners who have received a final grade.

The **Open Responses** tab also displays the same information for each separate ORA assignment in the course, grouped by the course units that include ORA assignments.

View Statistics for a Single ORA Assignment

To view metrics about learners in a single ORA assignment, including the number who are active in each step, follow these steps.

1. Open the ORA assignment in the course.
2. Scroll to the bottom of the assignment and select **View Assignment Statistics**.

   You see statistics for the assignment, including the total number of responses and the location ID for the assignment.

   In the **Learner Progress** section, for each assessment step in the assignment, you can see the number of learners who are currently working through (but who have not completed) that step. Only assessment types that exist in the assignment are included.

   Note: If a Staff Assessment step exists in the assignment, this step will always show 0 active learners, because no learner actions are required for that step.

   In addition to learners who are active in the assessment steps of the assignment, you can see the number of learners who are in the following states in the assignment.

   - **Waiting**: Learners who have finished the requirements for a step and are waiting for their responses to be assessed by peers or staff.
   - **Done**: Learners who have completed all of their required steps, and have received the required number of reviews.
   - **Cancelled**: Learners who have had their responses cancelled.

In the **Dates** section below **Learner Progress**, the release and due dates for each step in the assignment are shown.

Generate a Report for ORA Assignments

To generate a report containing details of the ORA assignments in the course, follow these steps.

1. View the live version of your course.
2. Select **Instructor**, and then select **Data Download**.
3. In the **Reports** section, select **Generate ORA Data Report**.

   A status message indicates that the ORA data report is being generated. This process might take some time to complete, but you can navigate away from this page and do other work while it runs.

   To check the progress of the report generation, reload the page in your browser and scroll down to the **Pending Tasks** section. The table shows the status of active tasks.

   When the report is complete, a linked .csv file name becomes available above the **Pending Tasks** section. File names are in the format `{course_id}_ORA_data_{datetime}.csv`. The most recently generated reports appear at the top of the list.
4. To open or save the generated ORA data report, locate and select the link for the grade report you requested.

   You can open .csv files in a spreadsheet application to sort, graph, and compare data.

Interpret the ORA Data Report

The ORA data report for your course is a time-stamped .csv file that contains data for all the ORA assignments in your course. For each ORA assignment in the course, the report provides information that includes each learner's anonymized ID, response, assessments details and scores, and the final score for the assignment. For more details about each column in the report, see the following descriptions.

Submission ID	Item ID	Anonymized Student ID	Date/Time Response Submitted	Response	Assessment Details	Assessment Scores	Date/Time Final Score Given	Final Score Points Earned	Final Score Points Possible	Feedback Statements Selected	Feedback on Peer Assessments
87f07e90-e231-11e5-af0b-0ab35812f93f	3	5bd5df3b2ed3a97acd5ad6f2fa88fb9e	2016-03-04 17:50:06.662467+00:00	{u'parts': [{u'text': u'Censorship is never a good idea. Anyone can be offended by something different, and if libraries started to respond to complaints by making certain books unavailable, then as the quote in the question says, we would soon "have no books left on the shelf for any of us."'}]}	Assessment #7 -- scored_at: 2016-03-04 18:07:46.046305+00:00 -- type: PE -- scorer_id: 7d554d4d3c75cc153c77ca2022a1e0d1 Assessment #5 -- scored_at: 2016-03-04 17:53:30.650492+00:00 -- type: PE -- scorer_id: ed8e6b92c55476286033378 5f764f998 -- overall_feedback: Use of the quotation was good.	Assessment #7 -- Content: Fair (1) -- Ideas: Fair (3) Assessment #5 -- Content: Good (3) -- Ideas: Good (5) -- feedback: Expresses the main idea clearly. Assessment #4 -- Content: Good (3) -- Ideas: Fair (3)	2016-03-04 17:53:30.776582+00:00	8	8	These assessments were useful.	I feel that the assessments performed on my response were fair.

The .csv file contains one row of data for each response from a learner.

- The IDs in the **Submission ID** and **Item ID** columns uniquely identify the problem within the course content and the learner's submission for that problem.
- The **Anonymized Student ID** column lists an ID for each learner without revealing confidential, personally identifiable data such as email addresses and usernames.
- The **Date/Time Response Submitted** column displays the date and time that the learner submitted her response, in YYYY-MM-DD HH-MM-SS format.
- The **Response** column displays the content of the learner's response.
- The **Assessment Details** column displays the following details for the assessments that were performed on the response.
  - The time and date that the assessment was submitted.
  - The type of assessment: self (SE), peer (PE), staff (ST).
  - The ID of the person who performed the assessment.
  - Any text comments about the response that were included in the assessment.
- The **Assessment Scores** column lists the scores that the response received in self, peer, or staff assessments.
- The **Date/Time Final Score Given**, **Final Score Points Earned**, and the **Final Score Points Possible** columns provide details of the final score that the response received. If a response has not received enough assessments for the assignment to be considered complete, these columns show a value of "None".
- The **Feedback Statements Selected** and **Feedback on Peer Assessments** columns together show the information that learners provided in the **Provide Feedback on Peer Assessments** section of their ORA assignments. This section is available to learners only when all assessments for an assignment have been completed, and provides an optional way for learners to comment on their experience of the peer assessment process.

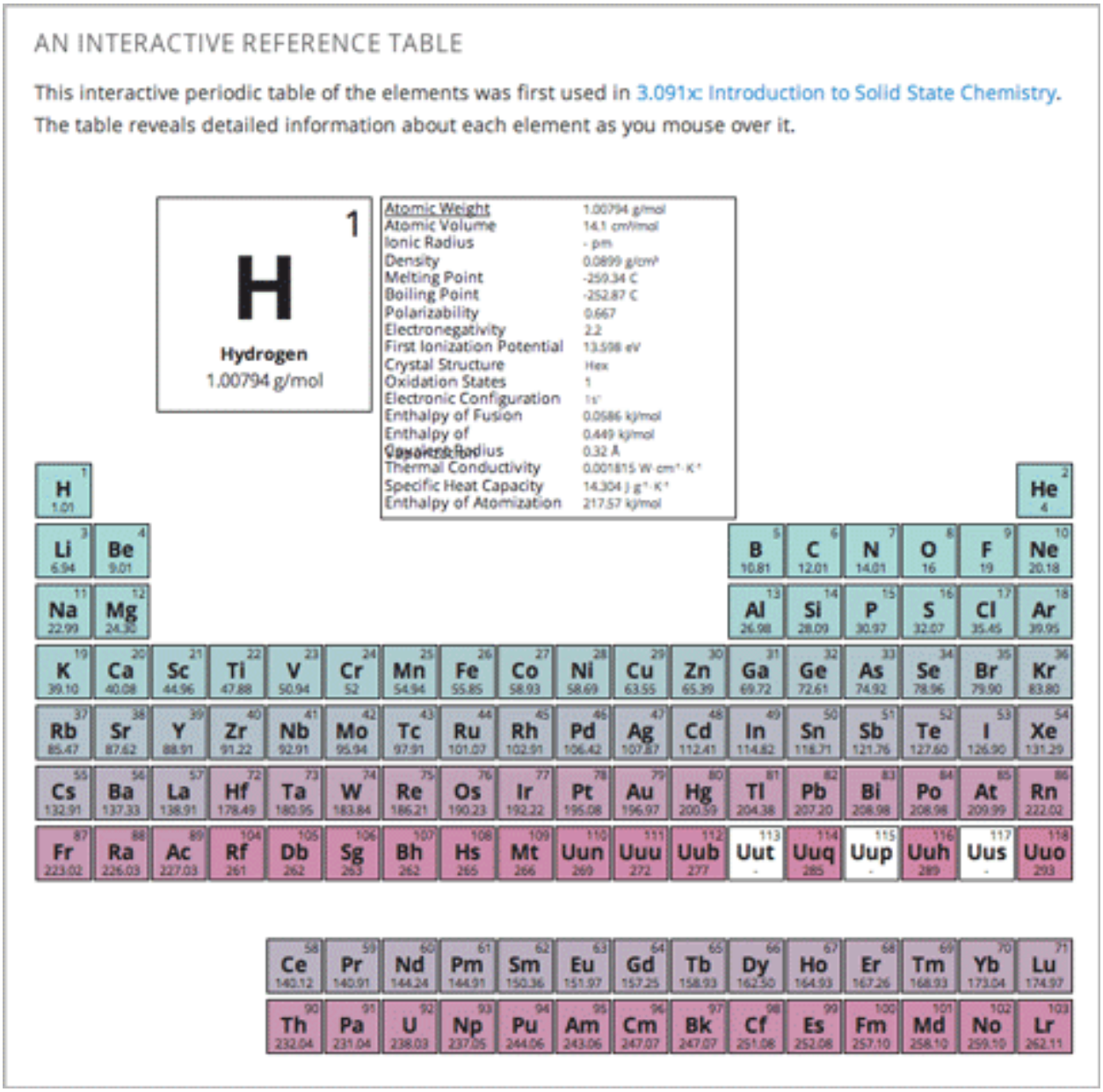
  The **Feedback Statements Selected** column displays the text of the feedback statements (if any) that the learner selected to describe their experience of the peer assessment process. Learners can select either "These assessments were useful" or "These assessments were not useful". They can also select either or both of "I disagree with one or more of the peer assessments of my response" and "Some comments I received were inappropriate".

  If a learner also provided a free-form comment in the text field below the selectable feedback statements, the text appears in the **Feedback on Peer Assessments** column.

Periodic Table Tool

Note: EdX does not support this tool.

You can create an interactive periodic table of the elements to help your students learn about various elements' properties. In the table below, detailed information about each element appears as the student moves the mouse over the element.



Create the Periodic Table Tool

To create a periodic table, you need the following files:

- Periodic-Table.js
- Periodic-Table.css
- Periodic-Table-Colors.css
- PeriodicTableHTML.txt

To download all of these files in a .zip archive, go to http://files.edx.org/PeriodicTableFiles.zip.

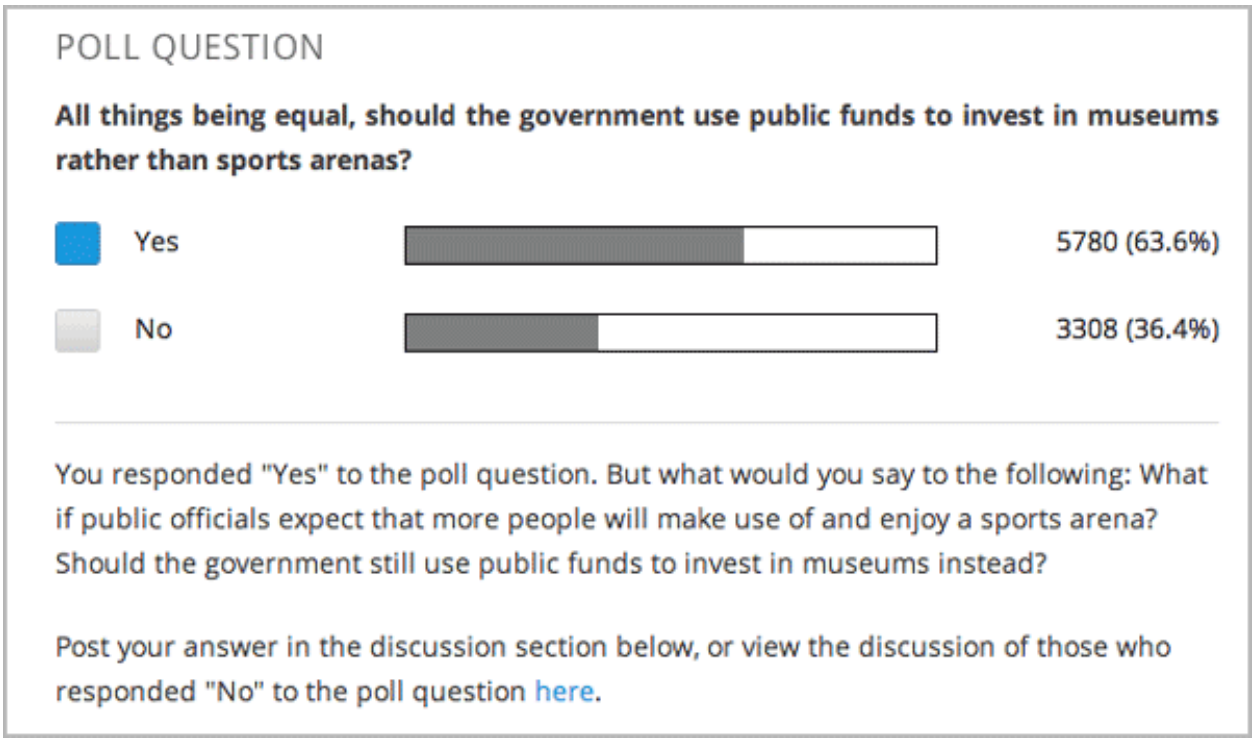
To create the periodic table, you need an HTML component.

1. Upload all of the files listed above *except PeriodicTable.txt* to the **Files & Uploads** page in your course.
2. In the unit where you want to create the problem, click **HTML** under **Add New Component**, and then click **HTML**.
3. In the component that appears, click **Edit**.
4. In the component editor, switch to the **HTML** tab.
5. Open the PeriodicTable.txt file in any text editor.
6. Copy all of the text in the PeriodicTable.txt file, and paste it into the HTML component editor. (Note that the PeriodicTableHTML.txt file contains over 6000 lines of code. Paste all of this code into the component editor.)
7. Click **Save**.

Poll Tool for OLX

Note: EdX does not support this tool.

You can run polls in your course so that your students can share opinions on different questions.



Note: Creating a poll requires you to export your course, edit some of your course's XML files in a text editor, and then re-import your course. We recommend that you create a backup copy of your course before you create the poll. We also recommend that you only edit the files that will contain polls in the text editor if you are very familiar with editing XML.

Terminology

Sections, subsections, units, and components have different names in the **Course Outline** view and in the list of files that you see after you export your course and open the .xml files for editing. The following table lists the names of these elements in the **Course Outline** view and in a list of files.

Course Outline View	File List
Section	Chapter
Subsection	Sequential
Unit	Vertical
Component	Discussion, HTML, problem, or video

For example, when you want to find a specific section in your course, you look in the **Chapter** folder when you open the list of files that your course contains. To find a unit, you look in the **Vertical** folder.

Create a Poll

1. In the unit where you want to create the poll, create components that contain all the content that you want *except* for the poll. Make a note of the 32-digit unit ID that appears in the **Unit Identifier** field under **Unit Location**.
2. Export your course. For information about how to do this, see Exporting and Importing a Course. Save the .tar.gz file that contains your course in a memorable location so that you can find it easily.
3. Locate the .tar.gz file that contains your course, and then unpack the .tar.gz file so that you can see its contents in a list of folders and files.
   - To do this on a Windows computer, you need to download a third-party program. For more information, see How to Unpack a tar File in Windows, How to Extract a Gz File, The gzip Home Page, or the Windows section of the How to Open .tar.gz Files page.
   - For information about how to do this on a Mac, see the Mac OS X section of the How to Open .tar.gz Files page.
4. In the list of folders and files, open the **Vertical** folder.

   Note: If your unit is not published, open the **Drafts** folder, and then open the **Vertical** folder in the **Drafts** folder.

5. In the **Vertical** folder, locate the .xml file that has the same name as the unit ID that you noted in step 1, and then open the file in a text editor such as Sublime 2. For example, if the unit ID is e461de7fe2b84ebeabe1a97683360d31, you open the e461de7fe2b84ebeabe1a97683360d31.xml file.

   The file contains a list of all the components in the unit, together with the URL names of the components. For example, the following file contains an HTML component followed by a discussion component.

```
<vertical display_name="Test Unit">
 <html url_name="b59c54e2f6fc4cf69ba3a43c49097d0b"/>
 <discussion url_name="8320c3d511484f3b96bdedfd4a44ac8b"/>
</vertical>
```

6. Add the following poll code in the location where you want the poll. Change the text of the prompt to the text that you want.

```
<poll_question display_name="Poll Question">
  <p>Text of the prompt</p>
  <answer id="yes">Yes</answer>
  <answer id="no">No</answer>
</poll_question>
```

   In the example above, if you wanted your poll to appear between the HTML component and the discussion component in the unit, your code would resemble the following.

```
<vertical display_name="Test Unit">
 <html url_name="b59c54e2f6fc4cf69ba3a43c49097d0b"/>
 <poll_question display_name="Poll Question">
   <p>Text of the prompt</p>
   <answer id="yes">Yes</answer>
   <answer id="no">No</answer>
 </poll_question>
 <discussion url_name="8320c3d511484f3b96bdedfd4a44ac8b"/>
</vertical>
```

7. After you add the poll code, save and close the .xml file.
8. Re-package your course as a .tar.gz file.
   - For information about how to do this on a Mac, see How to Create a Tar GZip File from the Command Line.
   - For information about how to do this on a Windows computer, see How to Make a .tar.gz on Windows.
9. In Studio, re-import your course. You can now review the poll question and answers that you added in Studio.

Note:

- Although polls render correctly in Studio, you cannot edit them in Studio. You will need to follow the export/import process outlined above to make any edits to your polls.
- A .csv file that contains student responses to the problem is not currently available for polls. However, you can obtain the aggregate data directly in the problem.

Format description

The main tag of poll module input is:

```
<poll_question> ... </poll_question>
```

`poll_question` can include any number of the following tags: any xml and `answer` tag. All inner xml, except for `answer` tags, we call "question".

poll_question tag

Xmodule for creating poll functionality - voting system. The following attributes can be specified for this tag:

```
name - Name of xmodule.
[display_name| AUTOGENERATE] - Display name of xmodule. When this attribute is not␣
↪defined - display name autogenerate with some hash.
[reset | False] - Can reset/revote many time (value = True/False)
```

answer tag

Define one of the possible answer for poll module. The following attributes can be specified for this tag:

```
id - unique identifier (using to identify the different answers)
```

Inner text - Display text for answer choice.

Example

Example of poll

```
<poll_question name="second_question" display_name="Second question">
    <h3>Age</h3>
    <p>How old are you?</p>
    <answer id="less18">&lt; 18</answer>
    <answer id="10_25">from 10 to 25</answer>
    <answer id="more25">&gt; 25</answer>
</poll_question>
```

Example of poll with unable reset functionality

```
<poll_question name="first_question_with_reset" display_name="First question with␣
↪reset"
    reset="True">
    <h3>Your gender</h3>
    <p>You are man or woman?</p>
    <answer id="man">Man</answer>
    <answer id="woman">Woman</answer>
</poll_question>
```

Poll Tool

Note: EdX offers full support for this tool.

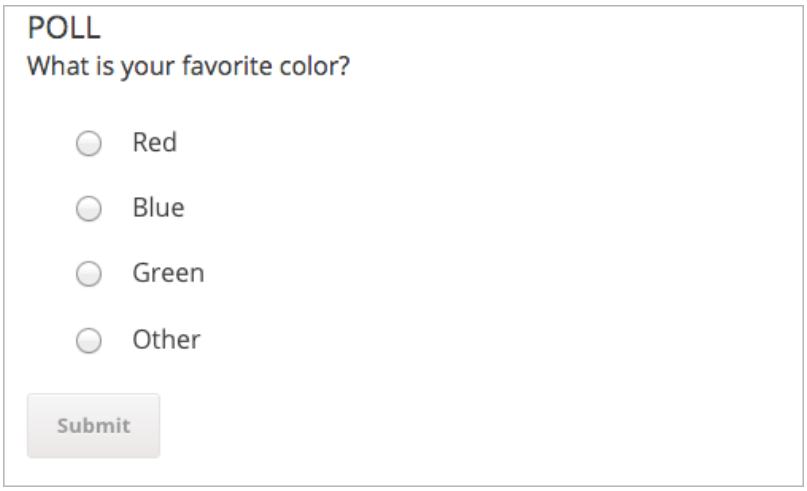
This section describes how to include polls in your course.

Overview

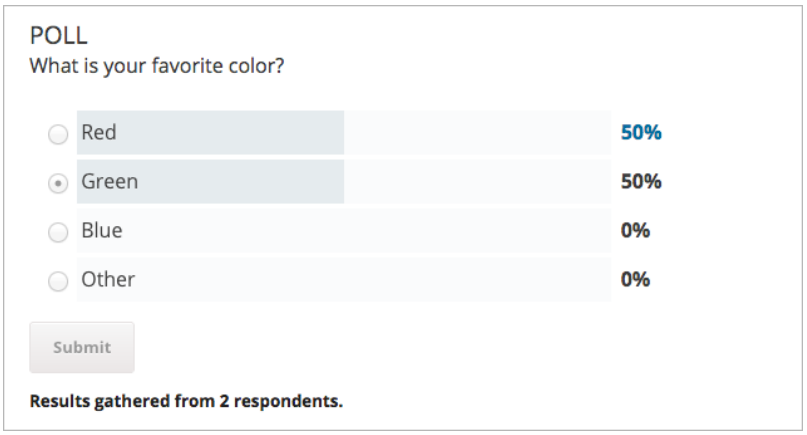
You can include polls in your course to gather learners' opinions on various questions.

For a poll, you configure one question and multiple possible answers. If you need to ask multiple questions, use the *Survey Tool*.

The following example poll has four possible answers to the question.



After learners submit their answers to the poll question, they see the poll results that have been gathered at this time, unless the poll has been configured to hide results.



Enable the Poll Tool

Before you can add a poll to your course, you must enable the poll tool in Studio or OLX (open learning XML).

To enable the poll tool in Studio, you add the `"poll"` key to the **Advanced Module List** on the **Advanced Settings** page. (Be sure to include the quotation marks around the key value.) For more information, see Enabling Additional Exercises and Tools.

Alternatively, you can use OLX to enable the poll tool.

Enable the Poll Tool in OLX

To enable polls in your course, you edit the XML file that defines the course structure.

Open the XML file for the course in the `course` directory. In the `course` element's `advanced-modules` attribute, add the string `poll`.

For example, the following XML code enables polls in a course.

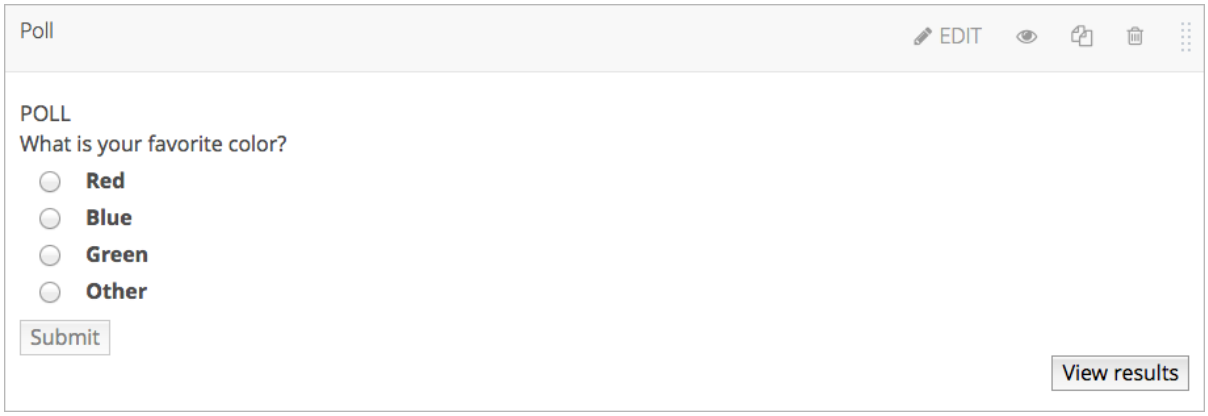
```
<course advanced_modules="[&quot;survey&quot;,
    &quot;poll&quot;]" display_name="Sample Course"
    start="2015-01-01T00:00:00Z">
    ...
</course>
```

Add a Poll in edX Studio

You must enable the poll tool before you add the component.

1. On the Course Outline page, open the unit where you want to add the poll.
2. Under **Add New Component** click **Advanced**, and then select **Poll**.

   The new component is added to the unit, with the default poll that contains three answer fields.



3. In the new component, select **Edit**.
4. In the **Display Name** field, enter the name for the component.
5. In the **Question/Prompt** field, enter text that learners see above the poll. You can use Markdown in this field.
6. In the **Feedback** field, enter text that learners see after they submit a responses. You can use Markdown in this field.
7. In the **Private Results** field, to hide poll results from learners, select **True**. If you leave the default value, **False**, learners see poll results after they submit responses.

8. In the **Maximum Submissions** field, change the value to the number of times that you want to allow learners to submit responses. Enter **0** to allow unlimited responses.

   Note: If you allow learners to submit responses more than once, you should set **Private Results** to **True**. Otherwise, learners will be able to change their responses after seeing others' responses.

   ---

9. Configure answers for the poll.
   (a) In each **Answer** field, enter the answer text that learners see.
   (b) You must enter either text or an image path, or both, for each answer. To enter an image, use the Studio URL for the image.
   (c) If you use an image, you must enter useful alternative text in the **Image alternate text** field for non-sighted users.
   (d) To add answers, select **Add answer** at the bottom of the editor. New answers are added at the bottom of the list.
   (e) To change the order of answers, select the up and down buttons next to each answer.
   (f) To remove an answer, select **Delete** next to the answer.
10. Select **Save**.

Add a Poll in OLX

To add a poll XBlock in OLX, you create the `poll` element. You can embed the `poll` element in the `vertical` element, or you can create the `poll` element as a standalone file that you reference in the vertical.

The following example shows the OLX definition for a poll with four answers.

```
<poll url_name="f4ae7de0006f426aa4eed4b0b8112da5" xblock-family="xblock.v1"
  feedback="Feedback"
  display_name="Poll"
  private_results="false"
  question="What is your favorite color?"
  max_submissions="1"
  answers="[
            [&quot;R&quot;,
              {
                &quot;img&quot;: &quot;/static/image.png&quot;,
                &quot;img_alt&quot;: &quot;Alt 1&quot;,
                &quot;label&quot;: &quot;Red&quot;
              }
            ],
            [&quot;B&quot;,
              {
                &quot;img&quot;: &quot;/static/image.png&quot;,
                &quot;img_alt&quot;: &quot;Alt 2&quot;,
                &quot;label&quot;: &quot;Blue&quot;
              }
            ],
            [&quot;G&quot;,
              {
                &quot;img&quot;: &quot;/static/image.png&quot;,
                &quot;img_alt&quot;: &quot;Alt3&quot;,
                &quot;label&quot;: &quot;Green&quot;
```

```
                }
              ],
              [&amp;quot;O&amp;quot;,
                {
                  &amp;quot;img&amp;quot;: &amp;quot;/static/image.png&amp;quot;,
                  &amp;quot;img_alt&amp;quot;: &amp;quot;Alt 4&amp;quot;,
                  &amp;quot;label&amp;quot;: &amp;quot;Other&amp;quot;
                }
              ]
            ]
"/>
```

poll Element Attributes

The following table describes the attributes of the `poll` element.

Attribute	Description
`url_name`	The unique identifier of the poll.
`xblock-family`	The XBlock version used. Must be `xblock.v1`.
`private_results`	Whether the poll results are shown to learners (`true`) or not (`false`).
`display_name`	The display name for the poll.
`question`	The prompt for the poll.
`feedback`	The text shown to learners after they submit a response.
`max_submissions`	The number of times a learner can submit poll answers. Use `0` to allow unlimited submissions. If you use a value other than `1`, set `private_results` to `true`. Otherwise, learners will be able to change their responses after seeing others' responses.
`answers`	An array of answers in the poll. Each answer has a unique identifier, and a dictionary that defines values for the following names. • `img`, the static URL of the answer image. • `img_alt`, the alternative text for the image. • `label`, the answer text. Each answer must have a value for `img` or `label`, or both.

Editing Published Polls

Do not publish a poll until you have completed and tested it. You should avoid changing a poll after learners have begun to use it.

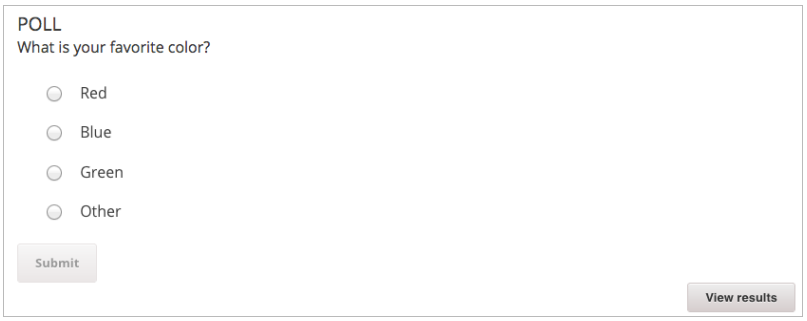
If you must edit a poll after learners have submitted answers take into account the following implications.

- If you edit the value of an answer, previous submissions are associated with the new answer value. This change can result in an inaccurate picture of the responses.
- If you change the poll so that previous submissions are invalid, by removing an answer, those submissions are deleted when learners next view the unit. Learners with invalid submissions can submit new responses.

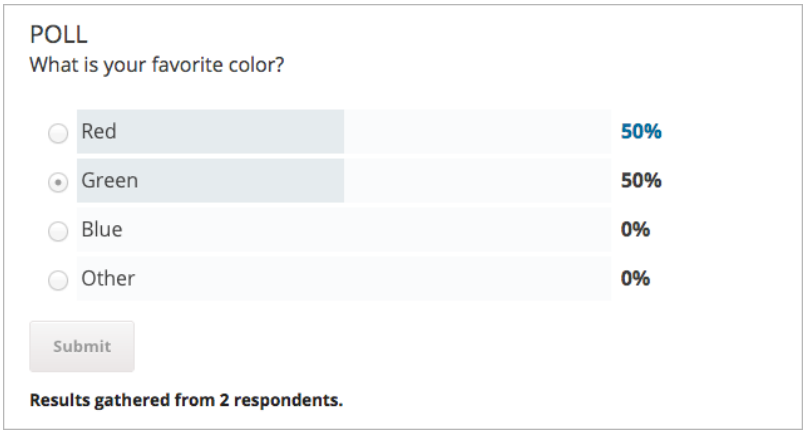
View Poll Results

When you view the poll as a course staff member, you can view results of the poll inside the course.

Select **View results** in the poll.



The results of the poll are then displayed.



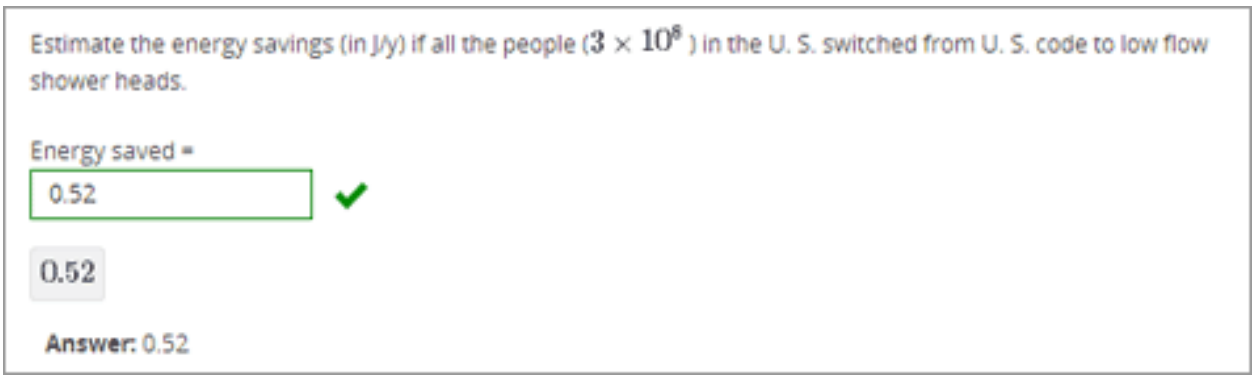
Problem Written in LaTeX

Note: EdX does not support this problem type.

Warning: This problem type is a prototype, and is not supported. By default, the ability to create these problems is not enabled in Studio. You must change the advanced settings in your course before you can create problems with LaTeX. Use this problem type with caution.

If you have an problem that is already written in LaTeX, you can use this problem type to easily convert your code into XML. After you paste your code into the LaTeX editor, you make a few adjustments.

Note: If you want to use LaTeX to typeset mathematical expressions in problems that you haven't yet written, use any of the other problem templates together with MathJax. For more information about how to create mathematical expressions in Studio using MathJax, see *Using MathJax for Mathematics*.



Create a Problem Written in LaTeX

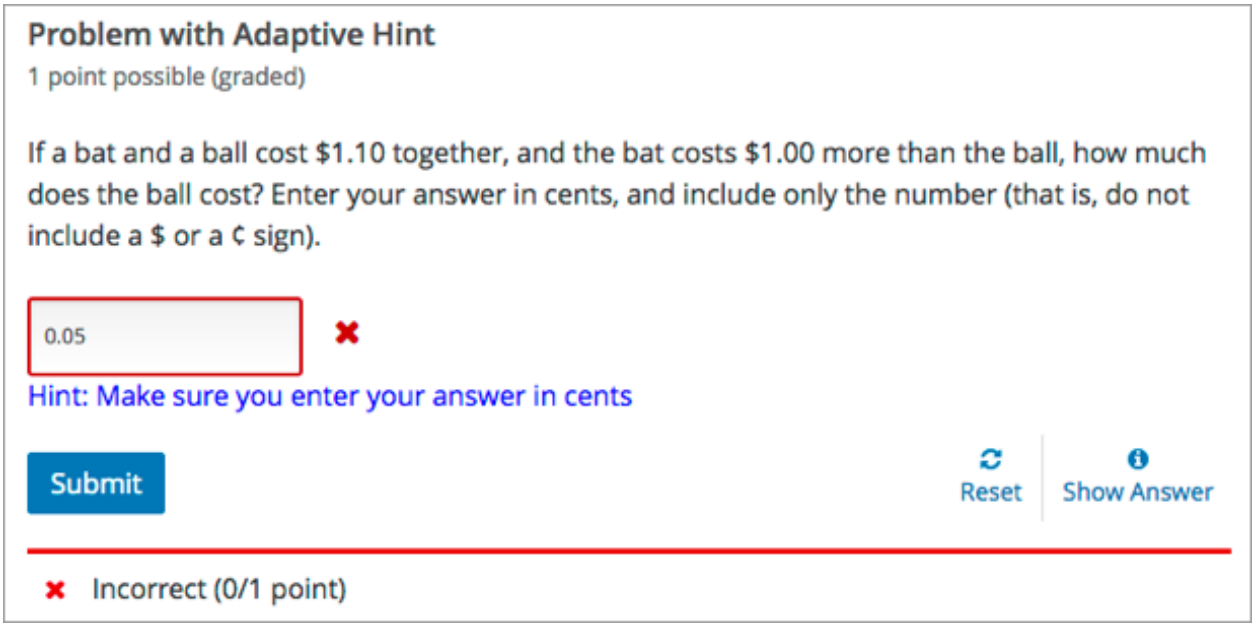
To create a problem written in LaTeX, follow these steps.

1. Enable the policy key in your course.
   (a) In Studio, click **Settings**, and then click **Advanced Settings**.
   (b) In the field for the **Enable LaTeX Compiler** policy key, change **false** to **true**.
   (c) At the bottom of the page, click **Save Changes**.
2. In the unit where you want to create the problem, click **Problem** under **Add New Component**, and then click the **Advanced** tab.
3. Click **Problem Written in LaTeX**.
4. In the component editor that appears, click **Edit**.
5. In the lower left corner of the component editor, click **Launch LaTeX Source Compiler**.
6. Replace the example code with your own code. You can also upload a Latex file into the editor from your computer by clicking **Upload** in the bottom right corner.
7. In the lower left corner of the LaTeX source compiler, click **Save & Compile to edX XML**.

Problem with Adaptive Hint

Note: EdX does not support this problem type.

A problem with an adaptive hint evaluates a student's response, then gives the student feedback or a hint based on that response so that the student is more likely to answer correctly on the next attempt. These problems can be text input problems or multiple choice problems.



Create a Problem with an Adaptive Hint

To create the problem illustrated above, follow these steps.

1. In the unit where you want to create the problem, click **Problem** under **Add New Component**, and then click the **Advanced** tab.
2. Click **Problem with Adaptive Hint**.
3. In the component that appears, click **Edit**.
4. In the component editor, replace the example code with the code below.
5. Click **Save**.

```
<problem>
        <text>
            <script type="text/python" system_path="python_lib">
    def test_str(expect, ans):
      print expect, ans
      ans = ans.strip("'")
      ans = ans.strip('"')
      return expect == ans.lower()

    def hint_fn(answer_ids, student_answers, new_cmap, old_cmap):
      aid = answer_ids[0]
      ans = str(student_answers[aid]).lower()
      print 'hint_fn called, ans=', ans
      hint = ''
      if '0.05' in ans:
         hint = 'Make sure you enter your answer in cents'

      if hint:
        hint = "&lt;font color='blue'&gt;Hint: {0}&lt;/font&gt;".format(hint)
        new_cmap.set_hint_and_mode(aid,hint,'always')
            </script>
            <p>If a bat and a ball cost $1.10 together, and the bat costs $1.00 more␣
↪than the
```

```
            ball, how much does the ball cost? Enter your answer in cents, and␣
↪include only
            the number (that is, do not include a $ or a ¢ sign).</p>
            <p>
                <customresponse cfn="test_str" expect="5">
                    <textline correct_answer="5" label="How much does the ball cost?"/
↪>
                    <hintgroup hintfn="hint_fn"/>
                </customresponse>
            </p>
        </text>
</problem>
```

Problem with Adaptive Hint XML

Template

```
<problem>
  <text>
    <script type="text/python" system_path="python_lib">
def test_str(expect, ans):
  print expect, ans
  ans = ans.strip("'")
  ans = ans.strip('"')
  return expect == ans.lower()

def hint_fn(answer_ids, student_answers, new_cmap, old_cmap):
  aid = answer_ids[0]
  ans = str(student_answers[aid]).lower()
  print 'hint_fn called, ans=', ans
  hint = ''
  if 'incorrect answer 1' in ans:
     hint = 'hint for incorrect answer 1'
  elif 'incorrect answer 2' in ans:
     hint = 'hint for incorrect answer 2'

  if hint:
    hint = "&lt;font color='blue'&gt;Hint: {0}&lt;/font&gt;".format(hint)
    new_cmap.set_hint_and_mode(aid,hint,'always')
</script>
    <p>TEXT OF PROBLEM</p>
    <p>
      <customresponse cfn="test_str" expect="ANSWER">
        <textline correct_answer="answer" label="LABEL TEXT"/>
        <hintgroup hintfn="hint_fn"/>
      </customresponse>
    </p>
  </text>
</problem>
```

Note: If the hints that you supply include characters, the letters must be lowercase.

Tags

- <text>: Surrounds the script and text in the problem.
- <customresponse>: Indicates that this problem has a custom response.
- <textline>: Creates a response field in the LMS where the student enters a response.
- <hintgroup>: Specifies that the problem contains at least one hint.

Tag: <customresponse>

Attributes

(none)

Children

- <textline>
- <hintgroup>

Tag: <textline>

Attributes

Attribute	Description
label (required)	Contains the text of the problem.
size (optional)	Specifies the size, in characters, of the response field in the LMS.
hidden (optional)	If set to "true", students cannot see the response field.
correct_answer (optional)	The answer to the problem. To supply a correct_answer value that includes letters, all letters **must be lowercase**. (Students' responses to the problem are not case sensitive. They can contain both uppercase and lowercase letters.)

Children

(none)

Tag: <hintgroup>

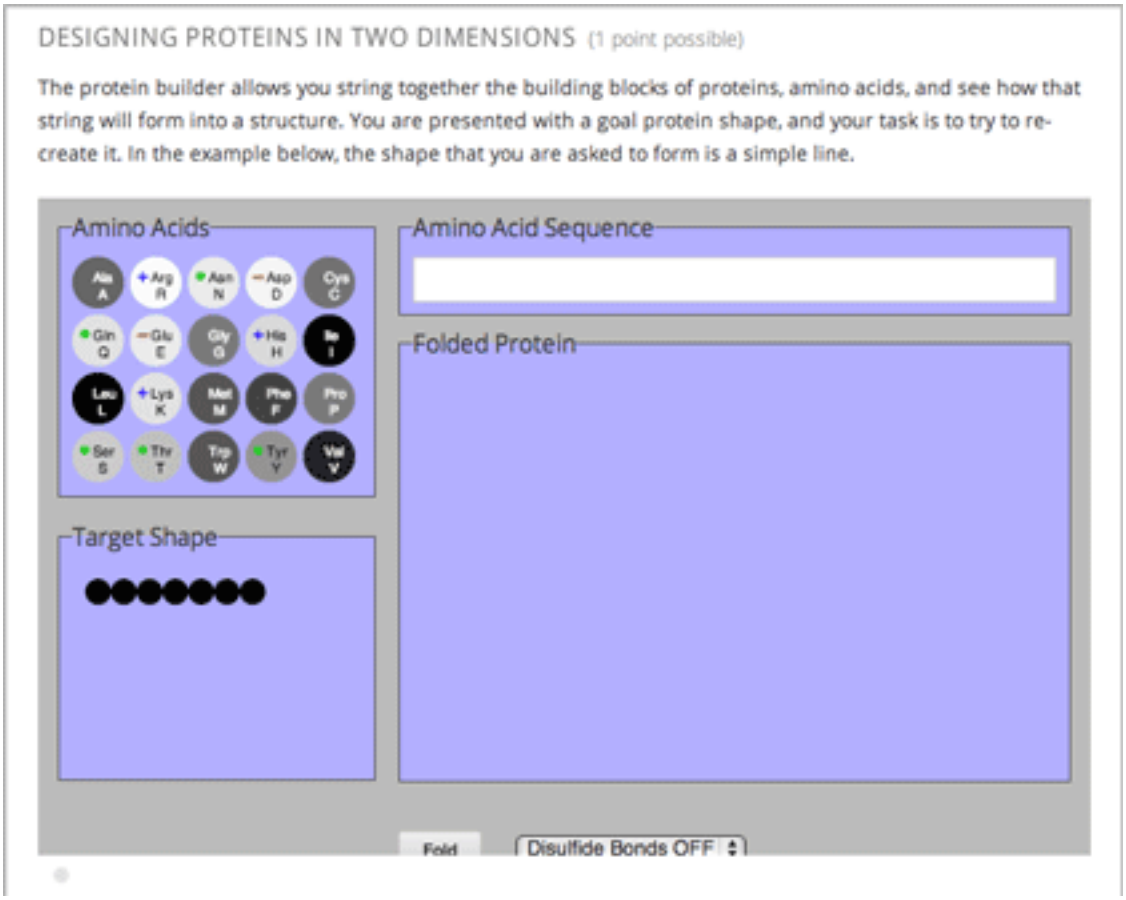
Attributes

Attribute	Description
hintfn	Must be set to **hint_fn** (that is, the tag must appear as <hintgroup hintfn="hint_fn"/>).

Protex Protein Builder Tool

Note: EdX does not support this tool.

The Protex protein builder asks students to create specified protein shapes by stringing together amino acids. In the example below, the goal protein shape is a simple line.



Create the Protein Builder Tool

To create the protein builder, follow these steps.

1. Under **Add New Component**, click **Problem**, and then click **Blank Advanced Problem**.
2. In the component that appears, click **Edit**.
3. In the component editor, paste the problem component code from the section that follows.
4. Make any changes that you want, and then click **Save**.

Protein Builder Tool Code

```
<problem>
    <p>The protein builder allows you string together the building blocks of proteins,
↪ amino acids, and see how that string will form into a structure. You are presented␣
↪with a goal protein shape, and your task is to try to re-create it. In the example␣
↪below, the shape that you are asked to form is a simple line.</p>
   <p>Be sure to click "Fold" to fold your protein before you click "Check".</p>

<script type="loncapa/python">

def protex_grader(expect,ans):
  import json
  ans=json.loads(ans)
```

```
  if "ERROR" in ans["protex_answer"]:
    raise ValueError("Protex did not understand your answer. Try folding the protein.
↪")
  return ans["protex_answer"]=="CORRECT"

</script>

  <text>
    <customresponse cfn="protex_grader">
      <designprotein2dinput width="855" height="500" target_shape="W;W;W;W;W;W;W"/>
    </customresponse>
  </text>
  <solution>
    <p>
      Many protein sequences, such as the following example, fold to a straight line.
↪You can play around with the protein builder if you're curious.
    </p>
    <ul>
      <li>
          Stick: RRRRRRR
      </li>
    </ul>
  </solution>
</problem>
```

In this code:

- **width** and **height** specify the dimensions of the application, in pixels.
- **target_shape** lists the amino acids that, combined in the order specified, create the shape you've asked students to create. The list can only include the following letters, which correspond to the one-letter code for each amino acid. (This list appears in the upper-left corner of the protein builder.)

A	R	N	D
C	Q	E	G
H	I	L	K
M	F	P	S
T	W	Y	V

Recommender Tool

Note: EdX does not support this tool.

The recommender provides learners with a list of online resources related to the course content. These resources are jointly managed by course team members and the learners.

Overview

The most common use of the recommender is for remediation of errors and misconceptions, followed by providing additional, more advanced resources.

For example, if a learner is working through a physics problem, the recommender could be used to show links to concepts used in the problem on Wikipedia, PhET, and OpenStax, as well as in the course itself. The recommender can help fill complex knowledge gaps and help move learners in the right direction.

Learners and course team members can complete the following tasks with the recommender.

- Add new resources.
- Edit existing resources and work jointly to improve the quality of resources (for example, give an informative resource summary).
- Manage quality by voting for high quality resources or flagging resources as spam or abuse.

Course team members can endorse useful resources or remove irrelevant entries.

If you use the recommender, you should inform learners through course content or course updates about the tool.

An example of a recommender in a course follows. The upper part of the figure illustrates a question in a problem set where the recommender is attached. The middle of the figure shows a list of resources and several gadgets for users to work on the resources. The bottom portion shows additional information about a given resource on mouse-over event.

Course team members should be sure to review all supplemental materials to assure that they are accessible before making them available through your course. For more information, see Accessibility Best Practices for Developing Course Content.

Enable the Recommender Tool

Before you can add a recommender component to your course, you must enable the recommender tool in Studio.

To enable the recommender tool in Studio, you add the `"recommender"` key to the **Advanced Module List** on the **Advanced Settings** page. (Be sure to include the quotation marks around the key value.) For more information, see Enabling Additional Exercises and Tools.

Add a Recommender

To add a recommender to a course, follow these steps.

1. Go to the unit in the course outline where you want to add the recommender.
2. Under **Add New Component**, select **Advanced**.

3. Select **recommender**.
4. In the component that appears, select **Edit**, and then specify settings as needed.
    - Whether to take users on an introduction tour when they see the tool the first time. If selected, the first time (and only the first time) a user sees the recommender, there will be a short guided tutorial.
    - Whether to disable the user interface functions which are under development. Because these are untested and under development, please leave these disabled unless otherwise advised by edX staff.
    - How many resources you want to show in each page of the resource list.
    - How many page icons you want to show in the pagination control (that is, the page range). The icons for pages from (current page - page range) to (current page + page range) will be shown.
5. Select **Save**.
6. Optionally, open the unit in the LMS and suggest some resources.

Survey Tool

Note: EdX offers full support for this tool.

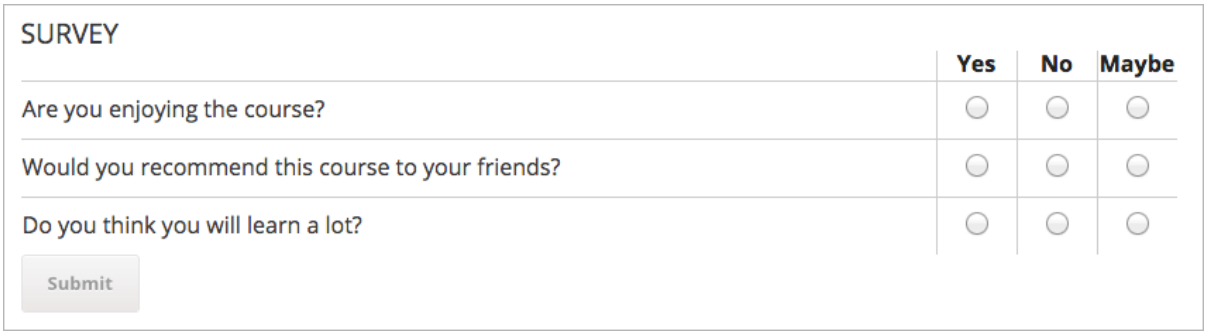
This section describes how to include surveys in your course.

Overview

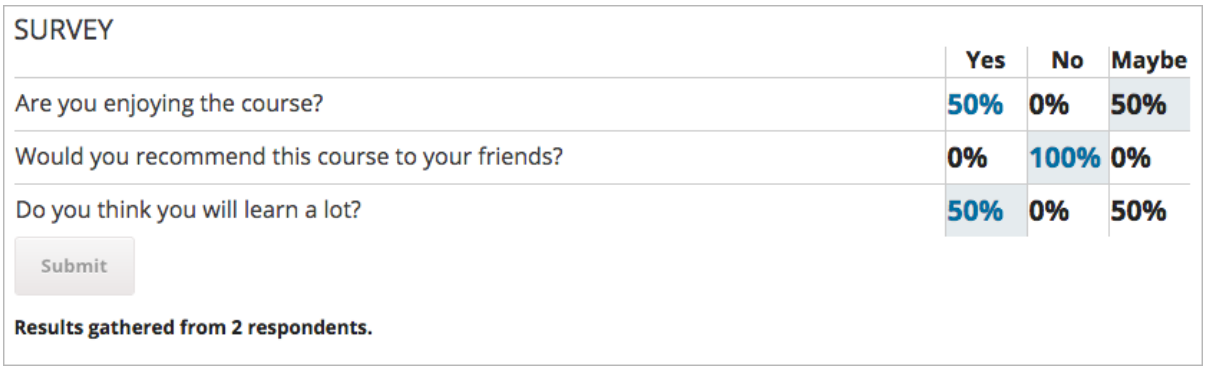
You can include surveys in your course to collect learner responses to multiple questions.

For a survey, you configure multiple question and multiple possible answers. The set of answers is used for each question in the survey. If you need to ask only one question, use the *Poll Tool*.

The following example survey has three questions, each with the same three possible answers.



After learners submit their answers to the survey, they see the survey results that have been gathered at this time, unless the survey has been configured to hide results.



Enable the Survey Tool

Before you can add a survey to your course, you must enable the survey tool in Studio or OLX (open learning XML).

To enable the survey tool in Studio, you add the `"survey"` key to the **Advanced Module List** on the **Advanced Settings** page. (Be sure to include the quotation marks around the key value.) For more information, see Enabling Additional Exercises and Tools.

Alternatively, you can use OLX to enable the survey tool.

Enable the Survey Tool in OLX

To enable the survey tool, you edit the XML file that defines the course structure.

Open the XML file for the course in the `course` directory. In the `course` element's `advanced-modules` attribute, add the string `survey`.

For example, the following XML code enables the survey tool.

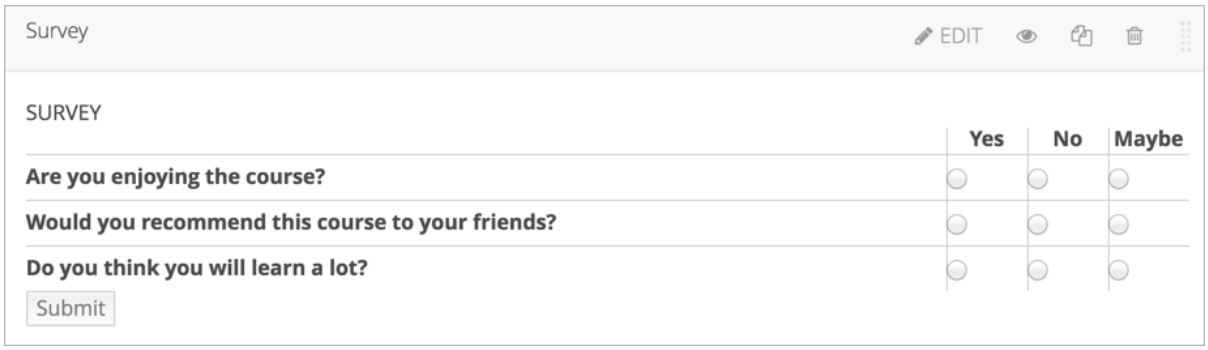
```
<course advanced_modules="[&amp;quot;survey&amp;quot;,
    &amp;quot;poll&amp;quot;]" display_name="Sample Course"
    start="2015-01-01T00:00:00Z">
    ...
</course>
```

Add a Survey in edX Studio

You must *enable the survey tool* before you add the component.

1. On the Course Outline page, open the unit where you want to add the survey.
2. Under **Add New Component** click **Advanced**, and then select **Survey**.

   The new component is added to the unit, with the default survey that contains three answer fields and three questions.



3. In the new component, select **Edit**.
4. In the **Display Name** field, enter the name for the component.
5. In the **Feedback** field, enter text that learners see after they submit responses.
6. In the **Private Results** field, to hide survey results from learners, select **True**. If you leave the default value, **False**, learners see survey results after they submit responses.
7. In the **Maximum Submissions** field, to allow learners to submit responses more than once, change the value. Enter **0** to allow unlimited responses.

Note: If you allow learners to submit responses more than once, you should set **Private Results** to **True**. Otherwise, learners will be able to change their responses after seeing others' responses.

---

8. Configure answers for the survey. Each answer is displayed to learners as a column, with a radio button they can select. Each answer is used for each survey question.
   (a) In each **Answer** field, enter the text for the column heading that learners will see.
   (b) To add answers, select **Add answer** at the bottom of the editor. New answers are added at the bottom of the list.
   (c) The top answer in the list is displayed to learners as the left-most answer column in the survey, and the bottom answer is displayed in the right-most column. To change the order of answers, select the up and down buttons next to each answer.
   (d) To remove an answer, select **Delete** next to the answer.
9. Configure questions for the survey. Each question is displayed to learners in the left-most column.
   (a) You must enter either text or an image path, or both, for each question. To enter an image, use the Studio URL for the image.
   (b) The survey template contains three questions. To add questions, select **Add question** at the bottom of the editor. New questions are added at the bottom of the list.
   (c) If you use an image, you must enter useful alternative text in the **Image alternate text** field for non-sighted users.
   (d) Questions are displayed to learners as rows in the order you configure them. To change the order of questions, select the up and down buttons next to each question.

(e) To remove a question, select **Delete** next to the question.

10. Select **Save**.

Add a Survey in OLX

To add a survey XBlock in OLX, you create the `survey` element. You can embed the `survey` element in the `vertical` element, or you can create the `survey` element as a stand-alone file that you reference in the vertical.

The following example shows the OLX definition for a survey with two questions.

```
<survey
  url_name="unique identifier for the survey"
  xblock-family="xblock.v1"
  questions="[
               [&quot;unique code for question 1&quot;,
                 {
                   &quot;img&quot;: &quot;Static URL to image&quot;,
                   &quot;img_alt&quot;: &quot;Alternative text for image&quot;,
                   &quot;label&quot;: &quot;Text of question 1&quot;
                 }
               ],
               [&quot;unique code for question 2&quot;,
                 {
                   &quot;img&quot;: &quot;Static URL to image&quot;,
                   &quot;img_alt&quot;: &quot;Alternative text for image&quot;,
                   &quot;label&quot;: &quot;Text of question 2&quot;
                  }
                ]
             ]"
  feedback="Feedback displayed to learner after submission"
  private_results="false"
  block_name="Display name for survey"
  max_submissions="1"
  answers="[
            [
              &quot;Unique identifier for answer 1&quot;,
              &quot;Answer text&quot;
            ],
            [
              &quot;Unique identifier for answer 2&quot;,
              &quot;Answer text&quot;
            ]
          ]"
/>
```

survey Element Attributes

The following table describes the attribute of the `survey` element.

Attribute	Description
`url_name`	The unique identifier of the survey.
`xblock-family`	The XBlock version used. Must be `xblock.v1`.
`questions`	An array of questions in the survey. Each question has a unique identifier, and a dictionary that defines values for the following names. • `img`, the static URL of the question image. • `img_alt`, the alternative text for the image. • `label`, the question text. Each question must have a value for `img` or `label`, or both.
`answers`	An array of answers in the survey. Each answer has a unique identifier, and a dictionary that defines values for the following names. • `img`, the static URL of the answer image. • `img_alt`, the alternative text for the image. • `label`, the answer text. Each answer must have a value for `img` or `label`, or both.
`feedback`	The text shown to learners after they submit a response.
`private_results`	Whether the survey results are shown to learners (`true`) or not (`false`).
`block_name`	The display name for the survey.
`max_submissions`	The number of times a learner can submit survey answers. Use `0` to allow unlimited submissions. If you use a value other than `1`, set `private_results` to `true`. Otherwise, learners will be able to change their responses after seeing others' responses.

Editing Published Surveys

Do not publish a survey until you have completed and tested it. You should avoid changing a survey after learners have begun using it.

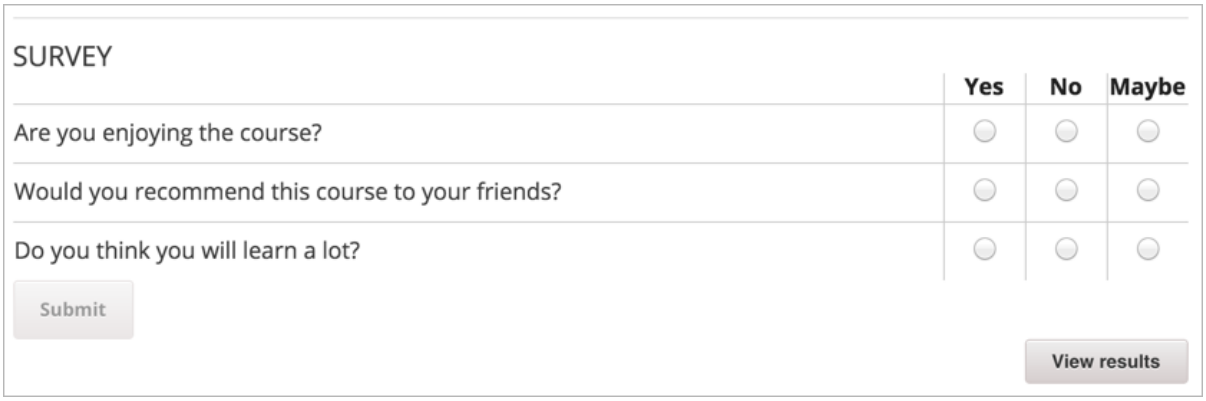
If you must edit a survey after learners have submitted answers, take into account the following implications.

- If you edit the value of a question or answer, previous submissions are associated with the new question or answer value. This change can result in an inaccurate picture of the responses.
- If you change the survey so that previous submissions are invalid, by removing a question or answer, those submissions are deleted when learners next view the unit. Learners with invalid submissions are permitted to submit new responses.

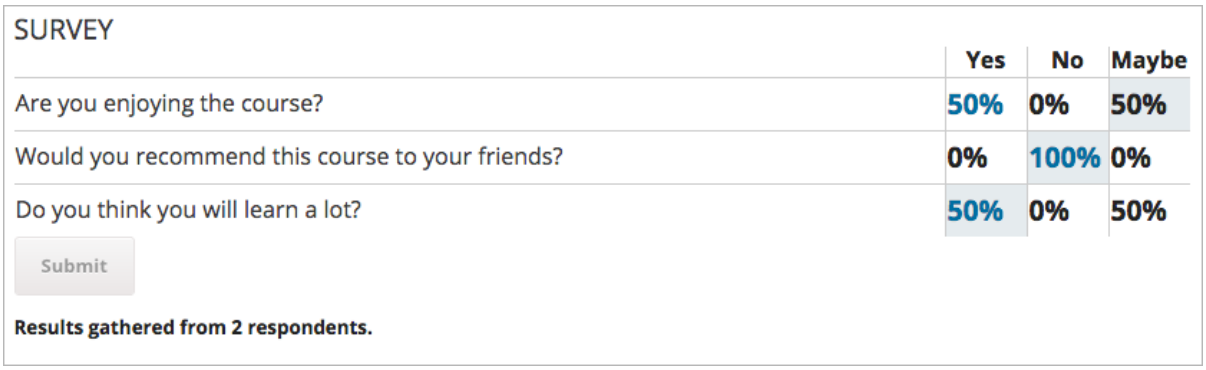
View Survey Results

When you view the survey as a course staff member, you can view results of the survey inside the course.

Select **View results** in the survey.



The results of the survey are then displayed.



Symbolic Response

This topic is planned to document features that the current symbolic response supports. In general, it allows the input and validation of math expressions, up to commutativity and some identities.

Features

This is a partial list of features, to be revised over time.

- sub and superscripts: an expression following the `^` character indicates exponentiation. To use superscripts in variables, the syntax is `b_x__d` for the variable `b` with subscript `x` and super `d`.

  An example of a problem.

```
<symbolicresponse expect="a_b^c + b_x__d" size="30">
  <textline math="1"
   preprocessorClassName="SymbolicMathjaxPreprocessor"
   preprocessorSrc="/static/js/capa/symbolic_mathjax_preprocessor.js"/>
</symbolicresponse>
```

  It's a bit of a pain to enter that.

- The script-style math variant. What would be outputted in latex if you entered `\mathcal{N}`. This is used in some variables.

  An example:

```
<symbolicresponse expect="scriptN_B + x" size="30">
  <textline math="1"/>
</symbolicresponse>
```

There is no fancy preprocessing needed, but if you had superscripts or something, you would need to include that part.

Text Input Problem

Note: EdX offers full support for this problem type.

The text input problem type is a core problem type that can be added to any course. At a minimum, text input problems include a question or prompt and a response field for free form answer text. By adding hints, feedback, or both, you can give learners guidance and help when they work on a problem.

- *Deprecated Hinting Method*

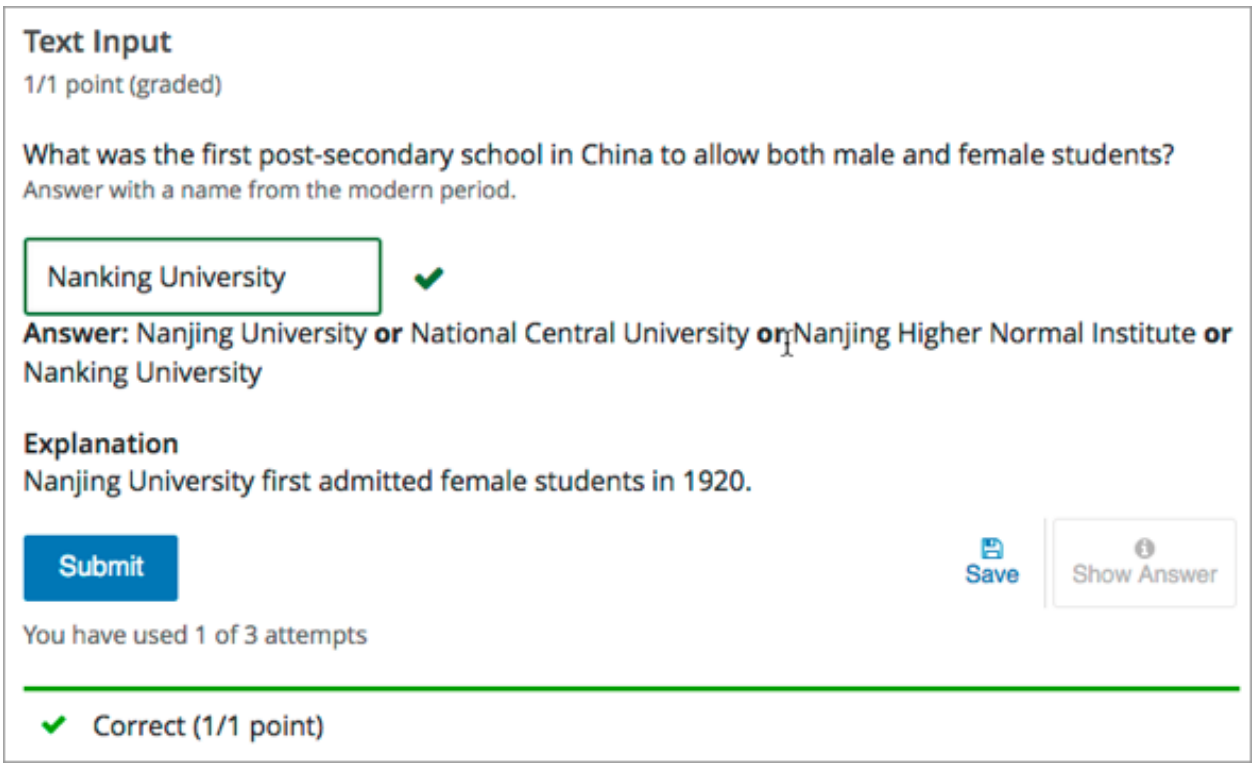
For more information about the core problem types, see Working with Problem Components.

Overview

In text input problems, learners enter text into a response field. The response can include numbers, letters, and special characters such as punctuation marks. Because the text that the learner enters must match the instructor's specified answer exactly, including spelling and punctuation, edX recommends that you specify more than one correct answer for text input problems to allow for differences in capitalization and typographical errors.

Example Text Input Problem

In the LMS, learners enter a value into a response field to complete a text input problem. An example of a completed text input problem follows.



To add the example problem illustrated above, in Studio you use the simple editor to enter the following text and Markdown formatting.

```
>>What was the first post-secondary school in China to allow both male and female␣
↪students?||Answer with a name from the modern period.<<

= Nanjing University
or= National Central University
or= Nanjing Higher Normal Institute
or= Nanking University

[explanation]
```

```
Nanjing University first admitted female students in 1920.
[explanation]
```

The OLX (open learning XML) markup for this example text input problem follows.

```
<problem>
  <stringresponse answer="Nanjing University" type="ci">
    <label>What was the first post-secondary school in China to allow both
     male and female students?</label>
    <description>Answer with a name from the modern period.</description>
    <additional_answer answer="National Central University"/>
    <additional_answer answer="Nanjing Higher Normal Institute"/>
    <additional_answer answer="Nanking University"/>
    <textline size="20"/>
    <solution>
      <div class="detailed-solution">
        <p>Explanation</p>
        <p>Nanjing University first admitted female students in 1920.</p>
      </div>
    </solution>
  </stringresponse>
</problem>
```

Analyzing Performance on Text Input Problems

For the text input problems in your course, you can use edX Insights to review aggregated learner performance data and examine submitted answers. For more information, see Using edX Insights.

Adding a Text Input Problem

You add text input problems in Studio by selecting the **Problem** component type and then using either the simple editor or the advanced editor to specify the prompt and the acceptable answer or answers.

Note: You can begin work on the problem in the simple editor, and then switch to the advanced editor. However, after you save any changes you make in the advanced editor, you cannot switch back to the simple editor.

Use the Simple Editor to Add a Text Input Problem

When you add a text input problem, you can choose one of these templates.

- **Text Input**
- **Text Input with Hints and Feedback**

These templates include the Markdown formatting that you use in the simple editor to add a problem without, or with, hints and feedback. To use the simple editor to add a problem, follow these steps.

1. In the unit where you want to create the problem, under **Add New Component** select **Problem**.
2. From the list of **Common Problem Types**, select the type of problem you want to add. Studio adds a template for the problem to the unit.
3. Select **Edit**. The simple editor opens to a template that shows the Markdown formatting that you use for this problem type.
4. Replace the guidance provided by the template to add your own text for the question or prompt, answer options, explanation, and so on.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Use the Advanced Editor to Add a Text Input Problem

You can use the advanced editor to identify the elements of a text input problem with OLX. For more information, see *Text Input Problem XML Reference*. To use the advanced editor to add a problem, follow these steps.

1. Follow steps 1-3 for creating the problem in the simple editor.
2. Select **Advanced Editor**. The advanced editor opens the template and shows the OLX markup that you can use for this problem type.
3. Replace the guidance provided by the template to add your own text. For example, replace the question or prompt, answer options, and explanation.

   To format equations, you can use MathJax. For more information, see *Using MathJax for Mathematics*.
4. Update the OLX to add optional elements and attributes required for your problem.
5. Select **Settings** to provide an identifying **Display Name** and define settings for the problem. For more information, see Defining Settings for Problem Components.
6. Select **Save**.

Adding Multiple Correct Responses

You can specify more than one correct response for text input problems. For example, instead of requiring learners to enter an answer of "Dr. Martin Luther King, Junior" exactly, you can also allow answers of "Martin Luther King, Jr." "Doctor Martin Luther King," and other variations. To do this, you can use the simple editor or the advanced editor.

Add Multiple Correct Responses in the Simple Editor

To specify additional correct responses in the simple editor, include `or=` before each additional correct response.

```
>>What African-American led the United States civil rights movement during the 1960s?<
↪<
=Dr. Martin Luther King, Jr.
or=Dr. Martin Luther King, Junior
or=Martin Luther King, Jr.
or=Martin Luther King
```

Add Multiple Correct Responses in the Advanced Editor

To specify an additional correct response in the advanced editor, within the `<stringresponse>` element add the `<additional_answer />` element with an `answer=""` attribute value.

```
<problem>
  <stringresponse answer="Dr. Martin Luther King, Jr." type="ci" >
    <label>What African-American led the United States civil rights movement during␣
↪the 1960s?</label>
    <additional_answer answer="Dr. Martin Luther King, Junior"/>
    <additional_answer answer="Martin Luther King, Jr."/>
    <additional_answer answer="Martin Luther King"/>
    <textline size="30"/>
  </stringresponse>
</problem>
```

Adding Feedback to a Text Input Problem

For an overview of feedback in problems, see *Adding Feedback and Hints to a Problem*. In text input problems, you can provide feedback for the correct answer or for a specified incorrect answer. Use feedback on incorrect answers as an opportunity to address common learner misconceptions. Feedback for text input questions should also provide guidance to the learner on how to arrive at the correct answer.

If you define multiple correct responses for the question, you can define feedback for each response.

Configure Feedback in the Simple Editor

You can configure feedback in the simple editor. When you add a text input problem, select the template **Text Input with Hints and Feedback**. This template has example formatted feedback that you can replace with your own text.

In the simple editor, you configure feedback for a text input problem with the following Markdown formatting.

```
=Correct Answer {{Feedback for learners who enter this answer.}}
not=Incorrect Answer {{Feedback for learners who enter this answer.}}
```

For example, the following problem has feedback for the correct answer and two common incorrect answers.

```
>>What is the largest state in the U.S. in terms of land area?<<

=Alaska {{Alaska is the largest state in the U.S. in terms of not only land
area, but also total area and water area. Alaska is 576,400 square miles,
more than double the land area of the second largest state, Texas.}}

not=Texas {{While many people think Texas is the largest state in terms of
land area, it is actually the second largest and contains 261,797 square
miles.}}

not=California {{California is the third largest state and contains 155,959
square miles.}}
```

Configure Feedback in the Advanced Editor

In the advanced editor, you configure answer feedback with the following syntax.

```
<problem>
  <stringresponse answer="Correct Answer" type="ci">
    <label>Question text</label>
    <correcthint>Feedback for the correct answer</correcthint>
    <stringequalhint answer="Incorrect Answer">Hint for the incorrect answer</
↪stringequalhint>
    <textline size="20"/>
  </stringresponse>
</problem>
```

For example, the following problem has feedback for the correct answer and two common incorrect answers.

```
<problem>
  <stringresponse answer="Alaska" type="ci">
    <label>What is the largest state in the U.S. in terms of land area?</label>
    <correcthint>Alaska is the largest state in the U.S. in terms of not
     only land area, but also total area and water area. Alaska is 576,400
     square miles, more than double the land area of the second largest
     state, Texas.</correcthint>
    <stringequalhint answer="Texas">While many people think Texas is the
     largest state in terms of land area, it is actually the second
     largest and contains 261,797 square miles.</stringequalhint>
    <stringequalhint answer="California">California is the third largest
     state and contains 155,959 square miles.</stringequalhint>
    <textline size="20"/>
  </stringresponse>
</problem>
```

Customizing Feedback Labels

By default, the feedback labels shown to learners are **Correct** and **Incorrect**. If you do not define feedback labels, learners see these terms when they submit an answer, as in the following example.

```
Incorrect: California is the third largest state and contains 155,959 square
miles.
```

You can configure the problem to override the default labels. For example, you can configure a custom label for a specific wrong answer.

```
Close but wrong: California is the third largest state and contains 155,959
square miles.
```

Note: The default labels **Correct** and **Incorrect** display in the learner's requested language. If you provide custom labels, they display as you define them to all learners. They are not translated into different languages.

Customize a Feedback Label in the Simple Editor

In the simple editor, you configure custom feedback labels with the following syntax.

```
not=Answer {{Label:: Feedback}}
```

That is, you provide the label text, followed by two colon (:) characters, before the feedback text.

For example, the following feedback is configured to use a custom label.

```
not=Texas {{Close but wrong:: While many people think Texas is the largest
state in terms of land area, it is actually the second largest of the 50 U.S.
states, containing 261,797 square miles.}}
```

Customize a Feedback Label in the Advanced Editor

In the advanced editor, you configure custom feedback labels with the following syntax.

```
<correcthint label="Custom Label">Feedback</correcthint>
<stringequalhint answer="Incorrect Answer" label="Custom Label">Feedback</
↪stringequalhint>
```

For example, the following feedback is configured to use custom labels.

```
<correcthint label="Right you are">Alaska is the largest state in the U.S.
 in terms of not only land area, but also total area and water area. Alaska
 is 576,400 square miles, more than double the land area of the second
 largest state, Texas.</correcthint>
<stringequalhint answer="Texas" label="Close but wrong">While many people
 think Texas is the largest state in terms of land area, it is actually the
 second largest of the 50 U.S. states containing 261,797 square miles.</
↪stringequalhint>
```

Adding Hints to a Text Input Problem

You can add hints to a text input problem using the simple editor or the advanced editor. For an overview of hints in problems, see *Adding Feedback and Hints to a Problem*.

Configure Hints in the Simple Editor

In the simple editor, you configure hints with the following syntax.

```
||Hint 1||
||Hint 2||
||Hint n||
```

Note: You can configure any number of hints. The learner views one hint at a time and views the next one by selecting **Hint** again.

For example, the following problem has two hints.

```
||A fruit is the fertilized ovary from a flower.||
||A fruit contains seeds of the plant.||
```

Configure Hints in the Advanced Editor

In the advanced editor, you add the `<demandhint>` element immediately before the closing `</problem>` tag, and then configure each hint using the `<hint>` element.

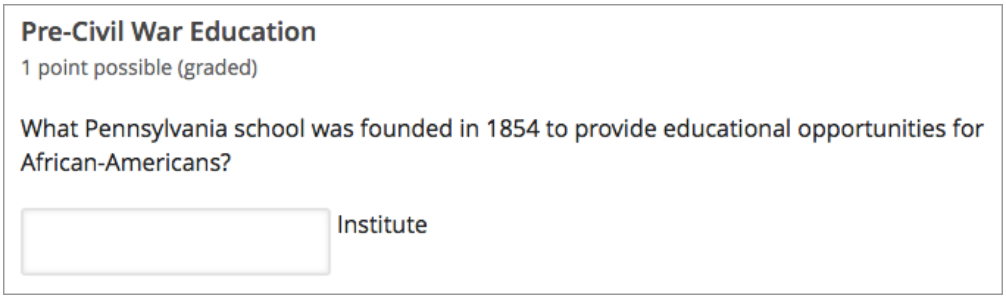
```
.
.
.
<demandhint>
  <hint>Hint 1</hint>
  <hint>Hint 2</hint>
  <hint>Hint 3</hint>
</demandhint>
</problem>
```

For example, the following OLX for a multiple choice problem shows two hints.

```
.
.
.
</multiplechoiceresponse>
<demandhint>
  <hint>A fruit is the fertilized ovary from a flower.</hint>
  <hint>A fruit contains seeds of the plant.</hint>
</demandhint>
</problem>
```

Adding Text after the Response Field

You might want to include a word, phrase, or sentence after the response field in a text input problem to help guide your learners or resolve ambiguity.



To do this, you use the advanced editor.

In the problem, locate the `textline` element. This element creates the response field for the problem and is a child of the `stringresponse` element. An example follows.

```
<problem>
  <stringresponse answer="Ashmun" type="ci">
    <label>What Pennsylvania school was founded in 1854 to provide
     educational opportunities for African-Americans?</label>
    <textline size="20"/>
  </stringresponse>
</problem>
```

To add text after the response field, add the `trailing_text` attribute together with the text that you want to use inside the `textline` element.

```
<problem>
  <stringresponse answer="Ashmun" type="ci">
    <label>What Pennsylvania school was founded in 1854 to provide
```

```
     educational opportunities for African-Americans?</label>
    <textline size="20" trailing_text="Institute"/>
  </stringresponse>
</problem>
```

Case Sensitivity and Text Input Problems

By default, text input problems do not require a case sensitive response. You can change this default to require a case sensitive answer.

To make a text input response case sensitive, you use the advanced editor.

In the advanced editor, the `stringresponse` element has a `type` attribute. By default, the value for this attribute is set to `ci`, for "case insensitive". An example follows.

```
<problem>
  <stringresponse answer="Paris" type="ci">
  .
  .
  .
  </stringresponse>
</problem>
```

Learners who submit an answer of either "Paris" or "paris" are scored as correct.

To make the response case sensitive, change the value of the `type` attribute to `cs`.

```
<problem>
  <stringresponse answer="Paris" type="cs">
  .
  .
  .
  </stringresponse>
</problem>
```

Learners who submit an answer of "Paris" are scored as correct, but learners who submit an answer of "PARIS" are scored as incorrect.

Response Field Length in Text Input Problems

By default, the response field for text input problems is 20 characters long.

You should preview the unit to ensure that the length of the response input field accommodates the correct answer, and provides extra space for possible incorrect answers.

If the default response field is not long enough, you can change it using the advanced editor.

In the advanced editor, the `textline` element has a `size` attribute. By default, the value for this attribute is set to `20`. An example follows.

```
<problem>
  <stringresponse answer="Democratic Republic of the Congo" type="ci">
    .
    .
    .
    <textline size="20"/>
```

```
  </stringresponse>
</problem>
```

To change the response field length, change the value of the `size` attribute.

```
<problem>
  <stringresponse answer="Democratic Republic of the Congo" type="ci">
    .
    .
    .
    <textline size="40" />
  </stringresponse>
</problem>
```

Allowing Regular Expressions as Answers for Text Input Problems

You can configure a text input problem to allow a regular expression as an answer. Allowing learners to answer with a regular expression can minimize the number of distinct correct responses that you need to define for the problem: if a learner responds with the correct answer formed as a plural instead of a singular noun, or a verb in the past tense instead of the present tense, the answer is marked as correct.

To do this, you use the advanced editor.

In the advanced editor, the `stringresponse` element has a `type` attribute. You can set the value for this attribute to `regexp`, with or without also including `ci` or `cs` for a case insensitive or case sensitive answer. An example follows.

```
<problem>
  <stringresponse answer="string pattern" type="regexp ci">
    .
    .
    .
  </stringresponse>
</problem>
```

The regular expression that the learner enters must contain, in whole or in part, the answer that you specify.

In this example, learners who submit an answer of “string pattern”, “String Patterns”, “string patterned”, or “STRING PATTERNING” are all scored as correct, but learners who submit an answer of “Strings Pattern” or “string patern” are scored as incorrect.

Text Input Problem XML Reference

Template

```
<problem>
  <stringresponse answer="Correct answer 1" type="ci regexp">
    <label>Question text</label>
    <description>Optional tip</description>
    <correcthint>Provides feedback when learners submit the correct
     response.</correcthint>
    <additional_answer answer="Correct answer 2"/>
    <additional_answer answer="Correct answer 3"/>
    <stringequalhint answer="Incorrect answer 1">Provides feedback when
     learners submit the specified incorrect response.</stringequalhint>
```

```
    <stringequalhint answer="Incorrect answer 2">Provides feedback when
     learners submit the specified incorrect response.</stringequalhint>
    <textline size="20" />
  </stringresponse>
  <demandhint>
    <hint>The first text string to display when learners request a hint.</hint>
    <hint>The second text string to display when learners request a hint.</hint>
  </demandhint>
</problem>
```

Elements

For text input problems, the `<problem>` element can include this hierarchy of child elements.

```
<stringresponse>
    <label>
    <description>
    <additional_answer>
    <correcthint>
    <stringequalhint>
    <textline>
    <solution>
<demandhint>
    <hint>
```

In addition, standard HTML tags can be used to format text.

`<stringresponse>`

Required. Indicates that the problem is a text input problem.

Attributes

Attribute	Description
`answer` (required)	Specifies the correct answer. Note that if you do not also add the `type` attribute and set it to `regexp`, the learner's answer must match the value for this attribute exactly.
`type` (optional)	Specifies whether the problem requires a case sensitive response and if it allows regular expressions. • If `type="ci"`, the problem is not case sensitive. • If `type="cs"`, the problem is case sensitive. • If `type="regexp"`, the problem allows regular expressions. You can also combine these values in a space separated list. For example, `<stringresponse type="regexp cs">` specifies that the problem allows regular expressions and is case sensitive.

Children

- <label>
- <description>
- <textline>
- <additional_answer>
- <correcthint>
- <stringequalhint>
- <solution>

<label>

Required. Identifies the question or prompt. You can include HTML tags within this element.

Attributes

None.

Children

None.

<description>

Optional. Provides clarifying information about how to answer the question. You can include HTML tags within this element.

Attributes

None.

Children

None.

<textline>

Required. Creates a response field in the LMS where the learner enters a text string.

Attributes

Attribute	Description
`size`	Optional. Specifies the size, in characters, of the response field in the LMS. Defaults to 20.
`hidden`	Optional. If set to "true", learners cannot see the response field.
`correct_answer`	Optional. Lists the correct answer to the problem.
`trailing_text`	Optional. Specifies text to appear immediately after the response field.

Children

None.

`<additional_answer>`

Optional. Specifies an additional correct answer for the problem. A problem can contain an unlimited number of additional answers.

Attributes

Attribute	Description
`answer`	Required. The text of the alternative correct answer.

Children

`<correcthint>`

`<correcthint>`

Optional. Specifies feedback to appear after the learner submits a correct answer.

Attributes

Attribute	Description
`label`	Optional. The text of the custom feedback label.

Children

None.

`<stringequalhint>`

Optional. Specifies feedback to appear after the learner submits an incorrect answer.

Attributes

Attribute	Description
`answer`	Required. The text of the incorrect answer.
`label`	Optional. The text of the custom feedback label.

Children

None.

`<solution>`

Optional. Identifies the explanation or solution for the problem, or for one of the questions in a problem that contains more than one question.

This element contains an HTML division `<div>`. The division contains one or more paragraphs `<p>` of explanatory text.

`<demandhint>`

Optional. Specifies hints for the learner. For problems that include multiple questions, the hints apply to the entire problem.

Attributes

None.

Children

`<hint>`

`<hint>`

Required. Specifies additional information that learners can access if needed.

Attributes

None.

Children

None.

Deprecated Hinting Method

The following example shows the XML format with the <hintgroup> element that you could use in the past to configure hints for text input problems. Problems using this XML format will continue to work in the edX Platform. However, edX recommends that you use the new way of configuring hints documented above.

```
<problem>
  <stringresponse answer="Correct answer 1" type="ci regexp">
    <label>Question text</label>
    <additional_answer>Correct answer 2</additional_answer>
    <additional_answer>Correct answer 3</additional_answer>
    <textline size="20" />
  <hintgroup>
    <stringhint answer="Incorrect answer A" type="ci" name="hintA" />
      <hintpart on="hintA">
        <startouttext />Text of hint for incorrect answer A<endouttext />
      </hintpart >
    <stringhint answer="Incorrect answer B" type="ci" name="hintB" />
      <hintpart on="hintB">
        <startouttext />Text of hint for incorrect answer B<endouttext />
      </hintpart >
    <stringhint answer="Incorrect answer C" type="ci" name="hintC" />
      <hintpart on="hintC">
        <startouttext />Text of hint for incorrect answer C<endouttext />
      </hintpart >
  </hintgroup>
  </stringresponse>
  <solution>
    <div class="detailed-solution">
    <p>Explanation or Solution Header</p>
    <p>Explanation or solution text</p>
    </div>
  </solution>
</problem>
```

Word Cloud Tool

Note: EdX offers provisional support for this tool.

In a word cloud, learners enter words into a field in response to a question or prompt. The words that all of the learners enter then appear instantly as a colorful graphic, with the most popular responses appearing largest. The graphic becomes larger as more learners answer. Learners can see how their peers answered as well as contribute their thoughts to the group.

For example, the following word cloud was created from learners' responses to a question about why they think addressing climate change is difficult.

Enable the Word Cloud Tool

Before you can add a word cloud to your course, you must enable the word cloud tool.

To enable the word cloud tool in Studio, you add the `"word_cloud"` key to the **Advanced Module List** on the **Advanced Settings** page. (Be sure to include the quotation marks around the key value.) For more information, see Enabling Additional Exercises and Tools.

Create a Word Cloud

To create a word cloud, follow these steps.

1. In the unit where you want to create the problem, select **Advanced** under **Add New Component**.
2. In the list of problem types, select **Word Cloud**.
3. In the component that appears, select **Edit**.
4. In the component editor, specify these settings.
    - **Display Name**: This name appears as a heading above the problem. Replace the default display name with a specific title or name for your word cloud. For example, "Why is climate change so difficult to address?"

      Unique, descriptive display names help you and your learners to identify assignments quickly and accurately as you navigate through the course.
    - **Inputs**: The number of text boxes provided for learners to enter words, phrases, or sentences.

- **Instructions**: Instructions that give learners specific information about how to contribute to this word cloud. For example, "In each the provided text boxes, enter a word or phrase to describe why you think it has been so difficult to make progress in addressing climate change."

  These instructions appear below the display name for the word cloud. It is important to provide clear instructions, especially for learners who have accessibility requirements.

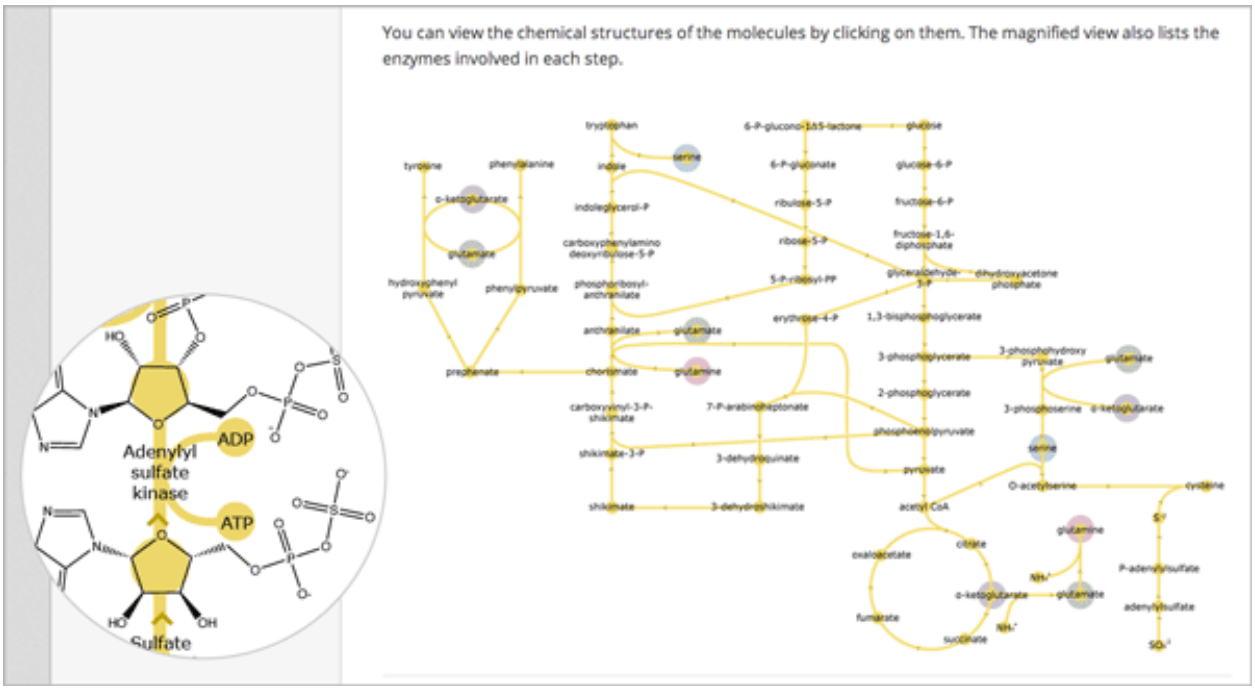
- **Maximum Words**: The maximum number of words that the generated word cloud displays. If learners enter 300 different words but the maximum is set to 250, only the 250 most commonly entered words appear in the word cloud.
- **Show Percents**: Select `True` or `False`. If this option is set to `True`, after learners have contributed to a word cloud, the frequency that each of their own contributed words has been entered is shown as a percentage of all words entered.

5. Select **Save**.

Zooming Image Tool

Note: EdX does not support this tool.

If you present information to your learners using very large or very detailed images, learners might not be able to clearly see all the information in the image. Use the zooming image tool to provide learners with the ability to zoom in to and enlarge selected areas of your image.



Components of a Zooming Image Tool

To create a zooming image tool, you need the following files.

- The image that you want learners to see when they access the unit.

- The image that appears in the magnified area when a learner selects the regular image. This image can be larger than the regular image.
- The `jquery.loupeAndLightbox.js` JavaScript file from `http://files.edx.org/jqueryloupeAndLightbox.js`.

Create a Zooming Image Tool

1. Download the `jquery.loupeAndLightbox.js` file by right-clicking the following link, and then selecting the option to save or download the linked file.

   http://files.edx.org/jqueryloupeAndLightbox.js

2. In Studio, select **Content** and then select **Files & Uploads** to upload your regular-sized image file, your small image file, and the `jquery.loupeAndLightbox.js` file. For more information about uploading files for your course, see Adding Files to a Course.
3. Add a zooming image tool to your course. In the course outline in Studio, select **Add New Component**, select **HTML**, and then select **Zooming Image Tool**.
4. In the new component that appears, select **Edit**.
5. In the component editor, replace the default problem text with your own text.
6. Select **HTML** from the toolbar to edit the HTML source code.
7. Scroll down in the file, and then replace the following placeholders with your own content.
   - Replace the following file name and path with the name and path of the image that you want to appear magnified when the user hovers over the regular image.

     `https://studio.edx.org/c4x/edX/DemoX/asset/pathways_detail_01.png`

     For example, your file name and path might be `/static/Image1.jpg`.
   - Replace the following file name and path with the name and path of the image that you want to appear when the page opens.

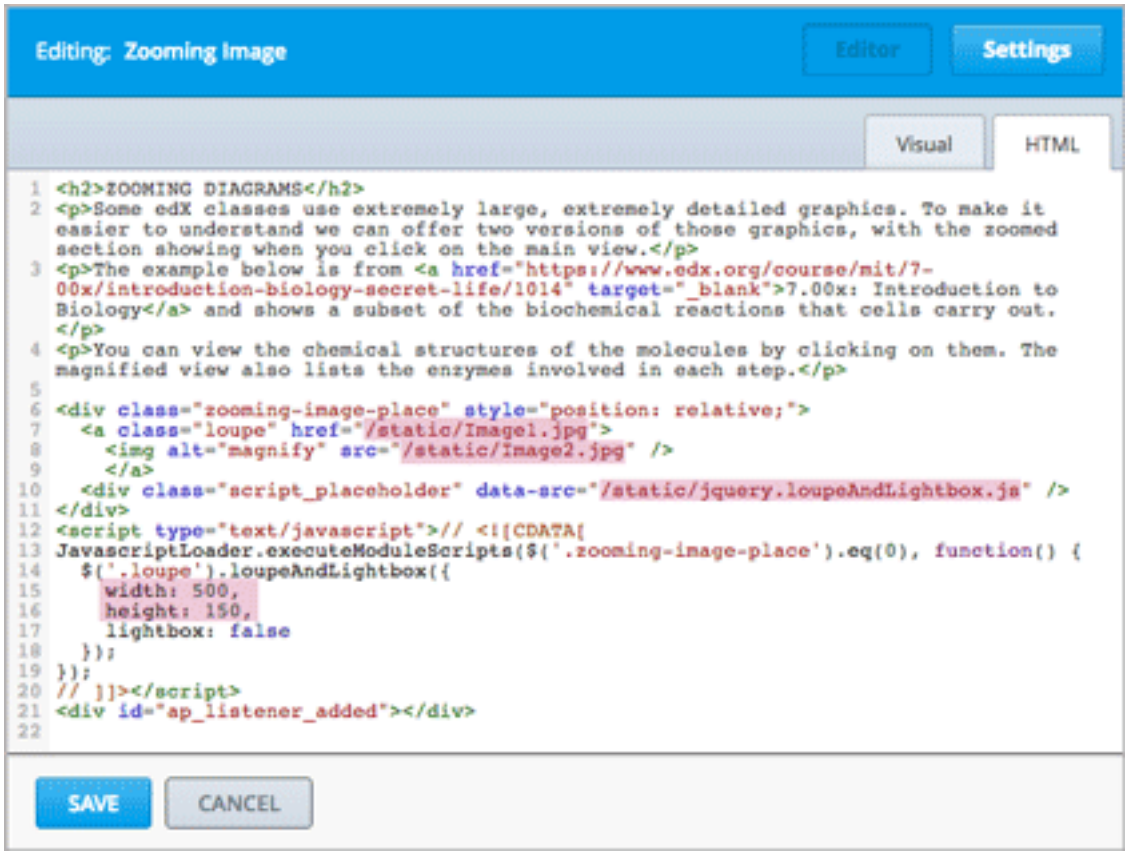
     `https://studio.edx.org/c4x/edX/DemoX/asset/pathways_overview_01.png`

     For example, your file name and path might be `/static/Image2.jpg`.
   - Replace the following name and file path with the name and path of the JavaScript file that you downloaded from `files.edx.org`.

     `https://studio.edx.org/c4x/edX/DemoX/asset/jquery.loupeAndLightbox.js`

     Because you uploaded the `jquery.loupeAndLightbox.js` file to the **Files & Uploads** page, your file name and path is `/static/jquery.loupeAndLightbox.js`.
   - (Optional) If you want the magnified area to be larger or smaller, change the `width` and `height` values from 350 to larger or smaller numbers. For example, you can change both numbers to 450. Or, if you want a horizontal oval instead of a circle, you can change the `width` value to a number such as 500 and the `height` value to a number such as 150.

   The HTML in your zooming image tool might resemble the following.



8. Select **Save**.

CHAPTER 12

Content Experiments

The topics in this section describe how to use OLX (open learning XML) to create and configure content experiments.

Overview of Content Experiments

This section provides an introduction to using content experiments.

For more information, see Configure Your Course for Content Experiments and Add Content Experiments to Your Course.

Overview

You use content experiments to show different course content to different groups of learners. Also known as "A/B tests" or "split tests", content experiments enable you to research and compare the performance of learners in different groups to gain insight into the relative effectiveness of your course content.

If your course uses content experiments, the grade report that you generate from the instructor dashboard includes a column identifying the experiment group that each learner has been assigned to. For more information, see Interpreting the Grade Report.

For information about analyzing events from content experiments, see Testing Events for Content Experiments in the *EdX Research Guide*.

Courses with Multiple Content Experiments

You can run multiple content experiments in your course. You can set up each experiment to use the same groups of learners, or you can set up each experiment to be independent and use a different grouping.

Important: If your course has multiple experiments, it is critical that you decide in advance if the experiments share the same groups of learners or if each experiment has its own unique grouping. If two experiments share the same grouping, then any learner that is in Group A for the first experiment will also be in Group A for the second one. If you want the experiments to be independent, then the experiments must use different groupings so that learners are randomly assigned for each experiment.

To determine the available groupings of learners, you set up group configurations using Studio or *using XML*.

You then select which group configuration to use when you add a content experiment using Studio or *using XML*.

Guidelines for Modifying Group Configurations

Review these guidelines if you must modify a group configuration after a course starts. These guidelines apply for courses built in Studio or using OLX (open learning XML).

Modifying a Group Configuration

After the course starts, **do not**:

- Delete group configurations.
- Change the `id` value of a group configuration.

Modifying Experiment Groups

After the course starts, **do not** change the `id` value of an experiment group.

You can change experiment group names at any time.

Removing Experiment Groups from Group Configurations

After a course in which you are running a content experiment has started, learners in a specific experiment group might have difficulties with the content or with the course experience. In this situation, you can remove the experiment group from the group configuration. Content that was specified for that experiment group is then no longer visible to learners.

Learners in the removed experiment group are reassigned evenly to one of the other experiment groups in the group configuration. Any problems that these learners completed in the removed experiment group content do not count toward their grades. The learners must begin the problem set again and complete all the problems in the experiment group content to which they are reassigned.

Removing an experiment group affects event data for the course. Ensure that researchers who are evaluating your course results are aware of the experiment group that you removed and the date on which you removed it.

Set Up Group Configuration for OLX Courses

You define group configurations in the `policy.json` file in the `policies` directory of an OLX (open learning XML) course.

To specify group configurations, you modify the value for the `user_partitions` policy key.

Note: `user_partitions` is the internal edX Platform name for group configurations.

The value for `user_partitions` is a JSON collection of group configurations, each of which defines the experiment groups of learners.

Note: Use names for group configurations that are meaningful. You select from the list of group configuration names when you add a content experiment.

See the following examples for more information.

Example: One Group Configuration

The following code shows an example JSON object that defines a group configuration with two learner segments.

```
"user_partitions": [{"id": 0,
                     "name": "Name of the group configuration",
                     "description": "Description of the group configuration.",
                     "version": 1,
                     "groups": [{"id": 0,
                                 "name": "Group 1",
                                 "version": 1},
                                {"id": 1,
                                 "name": "Group 2",
                                 "version": 1}]
                               }
                     ]
```

In this example:

- The `"id": 0` identifies the group configuration. For XML courses, the value is referenced in the `user_partition` attribute of the `<split_test>` element in the content experiment file.
- The `groups` array identifies the experiment groups to which learners are randomly assigned. For XML courses, each group `id` value is referenced in the `group_id_to_child` attribute of the `<split_test>` element.

Example: Multiple Group Configurations

The following code shows an example JSON object that defines two group configurations. The first group configuration divides learners into two experiment groups, and the second divides learners into three experiment groups.

```
"user_partitions": [{"id": 0,
                     "name": "Name of Group Configuration 1",
                     "description": "Description of Group Configuration 1.",
                     "version": 1,
                     "groups": [{"id": 0,
                                 "name": "Group 1",
```

```
                                "version": 1},
                               {"id": 1,
                                "name": "Group 2",
                                "version": 1}]}
                    {"id": 1,
                     "name": "Name of Group Configuration 2",
                     "description": "Description of Group Configuration 2.",
                     "version": 1,
                     "groups": [{"id": 0,
                                 "name": "Group 1",
                                 "version": 1},
                                {"id": 1,
                                 "name": "Group 2",
                                 "version": 1}
                                 {"id": 2,
                                 "name": "Group 3",
                                 "version": 1}
                                 ]}
                ]
```

Note: As this example shows, each group configuration is independent. Group IDs and names must be unique within a group configuration, but not across all group configurations in your course.

Add a Content Experiment in OLX

You work with multiple XML files to configure a content experiment. This section steps through the files involved in a content experiment that shows different content to two different groups of learners.

Define the Content Experiment in the Sequential File

You reference a content experiment in the file for the subsection, or sequential, in the `sequential` directory. For example:

```
...
<vertical url_name="name for the unit that contains the A/B test" display_name="A/B␣
↪Test Unit">
    <split_test url_name="name of A/B test file in the split_test folder"/>
</vertical>
.....
```

The `<split_test>` element's `url_name` value references the name of the content experiment file in the `split_test` directory.

Caution: You can only define a content experiment in a unit, or vertical, in which different collections of components are associated with different experiment groups. You cannot define a content experiment at a subsection (sequential) or section (chapter) level and have different units or subsections associated with different groups.

Define the Experiment Content in the Split Test File

After you define the content experiment in the sequential file, you define the course content you want to test in the file in the `split_test` directory. This is the file referenced in the `<split_test>` element in the sequential file, as shown above.

In the content experiment file, you add elements for the experiment content. For this example, you add two `<vertical>` elements to compare the two different sets of content.

```
<split_test url_name="AB_Test.xml" display_name="A/B Test" user_partition_id="0"
            group_id_to_child='{"0": "i4x://path-to-course/vertical/group_a",
                               "1": "i4x://path-to-course/vertical/group_b"}'>
    <vertical url_name="group_a" display_name="Group A">
       <html>Welcome to group A.</html>
       <video url_name="group_a_video"/>
    </vertical>
    <vertical url_name="group_b" display_name="Group B">
        <html>Welcome to group B.</html>
        <problem display_name="Checkboxes">
            <p>A checkboxes problem presents checkbox buttons for learner input.
               Learners can select more than one option presented.</p>
            <choiceresponse>
                <checkboxgroup label="Select the answer that matches">
                    <choice correct="true">correct</choice>
                    <choice correct="false">incorrect</choice>
                    <choice correct="true">correct</choice>
                </checkboxgroup>
            </choiceresponse>
        </problem>
    </vertical>
</split_test>
```

In this example:

- The `user_partition_id` value references the ID of the experiment defined in the `policy.json` file.
- The `group_id_to_child` value references the IDs of the groups defined in the `policy.json` file and maps the group IDs to specific content.

  For example, the value for group `0`, `i4x://path-to- course/vertical/group_a`, maps to the `<vertical>` element with the `url_name` equal to `group_a`. Therefore, learners in group 0 see the content in that vertical.

For information about the `policy.json` file, see *Set Up Group Configuration for OLX Courses*.

Test Content Experiments

You should test content experiments in your course before the course starts, to ensure that content is delivered to experiment groups as you intended.

For more information, see Test Content Experiments.

CHAPTER 13

Example of an OLX Course

OLX (open learning XML) is a flexible system that you can use to create edX courses in many ways. While there is no best way to structure courses in all situations, there are best practices that help make an OLX course easier to create and maintain.

This section uses the edX-Insider course as an example of how to create an OLX course. The files for edX-Insider are stored in GitHub, so you can explore how the course is made for yourself.

These topics examine the overall structure of edX-Insider and how the courseware is defined.

The Structure of edX-Insider

This topic describes the structure of the edX-Insider course.

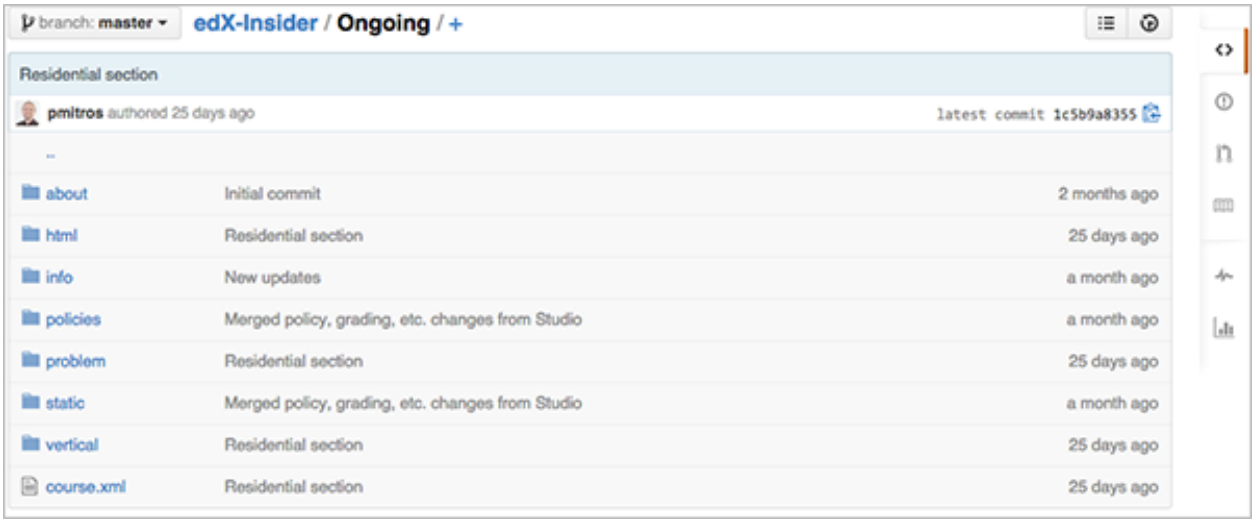
For information about how a generic OLX (open learning XML) course is structured, see *OLX Course Structure*.

For information about how a course exported from edX Studio is structured, see *Example of OLX for a Studio Course*.

Note: The structure and content of edX-Insider can change without corresponding updates being made to this reference guide.

edX-Insider and Directory File Structures

All files and subdirectories that comprise edX-Insider are stored in the Ongoing directory in the edX-Insider Git repository.



Top-level Directory

The Ongoing directory in the edX-Insider Git repository contains the `course.xml` file as well as XBlock and Platform directories.

- The course.xml file contains the XML for the courseware. All chapters and sequentials are defined in `course.xml`.
- Most verticals are defined in `course.xml`; two verticals are referenced in other files.
- The content of some HTML XBlocks is embedded within `course.xml`; other HTML XBlocks are referenced in other files.
- Problems are referenced in other files.

For more information, see *The edX-Insider course.xml File*.

The HTML XBlock Directory

While some HTML content is embedded in `course.xml`, many HTML XBlocks are stored as separate files in the `HTML` directory.

Example of a Referenced XBlock

You can reference an XBlock from the `course.xml` file.

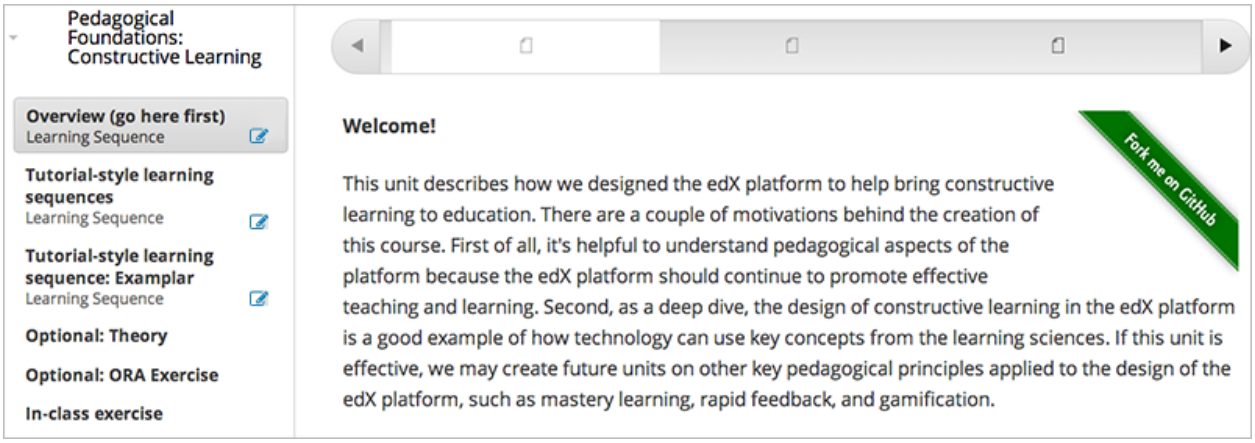
For example, in `course.xml`, the first vertical in the courseware contains a single HTML XBlock with the display name `Week overview`, which references `Week_overview` in the `url_name` attribute:

```
<chapter display_name="Pedagogical Foundations: Constructive Learning"
    url_name="Week_2_Technology_enabled_constructive_learning">
    <sequential format="Learning Sequence" graded="true"
        display_name="Overview (go here first)"
```

```
        url_name="Overview_go_here_first">
        <vertical display_name="Week's overview" url_name="Week_s_overview">
            <html display_name="Week overview" filename="Week_overview"
                url_name="Week_overview"/>
```

There is a file called `Week_overview.html` in the `html` directory that contains the content for that HTML component. For detailed information, see the Week_overview.html file in GitHub.

For a learner, that HTML component appears as the first unit of the course.



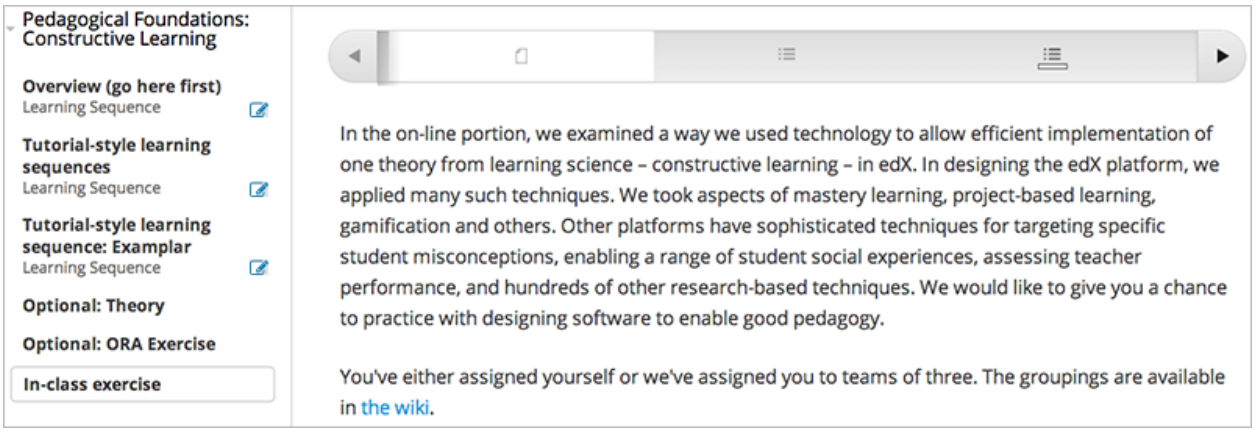
Example of an Inline XBlock

You can include XBlock content within the `course.xml` file. You can do this for ease of reading and maintenance when you do not need to reuse the content.

For example, in `course.xml`, the sequential with the display name `In-class exercise` contains embedded HTML content.

```
<sequential display_name="In-class exercise" url_name="in_class">
    <html display_name="Overview" url_name="overview">
        <p>In the on-line portion,
           we examined a way we used technology to allow efficient
           implementation of one theory from learning science - constructive
           learning - in edX. In designing the edX platform, we applied many
           such techniques. We took aspects of mastery learning, project-
           based learning, gamification and others. Other platforms have
           sophisticated techniques for targeting specific student
           misconceptions, enabling a range of student social experiences,
           assessing teacher performance, and hundreds of other research-
           based techniques. We would like to give you a chance to practice
           with designing software to enable good pedagogy.
        </p>
            . . .
    </html>
```

For a student, that HTML component appears as a unit of the course in the same way as a referenced HTML component does.



Platform Directories

The edX-Insider course contains information in the course subdirectories as described below.

`about` Directory

The `about` directory contains the following files.

- `overview.html`, which contains the content for the course overview page that students see in the the Learning Management System (LMS).
- `short_description.html`, which contains the content for the course in the course list.

For more information, see *The Course About Pages*.

`info` Directory

The `info` directory contains the following files.

- `handouts.html`, which contains the content for the **Course Handouts** page in the course.
- `updates.html`, which contains the course updates students see when opening a course.

`policies` Directory

The `policies` directory contains the following files.

- `assets.json`, which defines all files used in the course, such as images.
- A course directory named `Ongoing`, which contains:
  - `grading_policy.json`, which defines how student work is graded in the course.
  - `policy.json`, which defines various settings in the course.

For more information, see *Course Policies*.

`static` Directory

The `static` directory contains the files used in the course, such as images or PDFs.

For more information, see *Course Assets*.

`vertical` Directory

The `vertical` directory contains the XML for two verticals used in the course.

- `constructive_ora_exercise.xml`
- `in_class_ora.xml`

You can embed verticals in the `course.xml` file, and this is usually the most straightforward option. However, with OLX, you can also store XML for verticals in separate files in the `vertical` directory.

In this case, verticals for open response assessments are stored in their own files.

The vertical files are referenced in `course.xml` as follows:

```
<vertical url_name="constructive_ora_exercise"></vertical>
```

And:

```
<vertical url_name="in_class_ora"></vertical>
```

The edX-Insider `course.xml` File

The courseware for edX-Insider is defined in the course.xml file and follows the organization described in *The Courseware Structure*.

- *edX-Insider Course Hierarchy*
- *Sequentials that Contain XBlocks*

edX-Insider Course Hierarchy

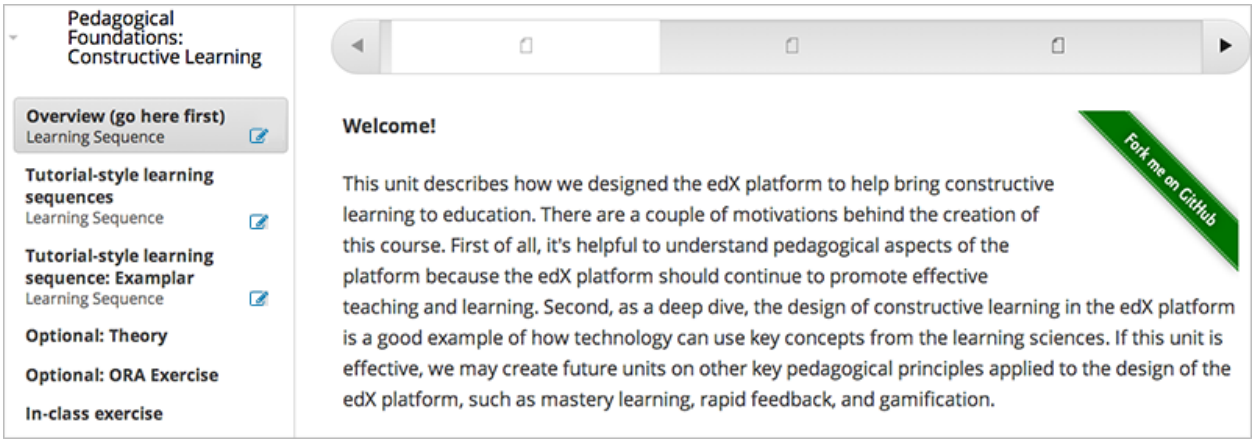
The edX-Insider courseware is organized into chapters, sequentials, and verticals.

For example, the following XML defines the first chapter, sequential, and vertical in the course.

```
<chapter display_name="Pedagogical Foundations: Constructive Learning"
    url_name="Week_2_Technology_enabled_constructive_learning">
    <sequential format="Learning Sequence" graded="true"
        display_name="Overview (go here first)"
        url_name="Overview_go_here_first">
        <vertical display_name="Week's overview" url_name="Week_s_overview">
            <html display_name="Week overview" filename="Week_overview"
                url_name="Week_overview"/>
                <done display_name="Read week overview"
                    url_name="Read_week_overview" align="right"/>
```

The vertical `Week's Overview` contains an HTML component that references the file `Week_overview` in the `HTML` directory.

Learners see this content in the Learning Management System as follows.



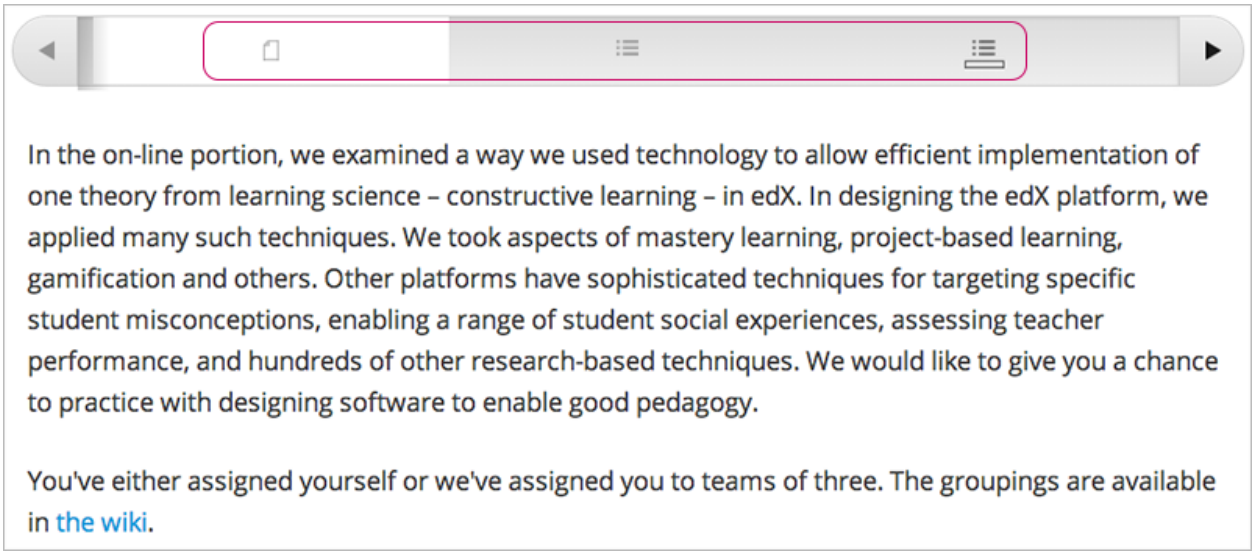
Sequentials that Contain XBlocks

One advantage of OLX (open learning XML) is the flexibility it allows in how you organize your course. For example, edX-Insider demonstrates that you can nest XBlocks and problems directly in a sequential, without the need for a vertical. This streamlines the course creation process while maintaining consistency in how students interact with courseware.

The following example XML defines a sequential that has, as children, an HTML XBlock, a reference to a vertical that is defined in another file, and a reference to a problem defined in another file.

```
<sequential display_name="In-class exercise" url_name="in_class">
    <html display_name="Overview" url_name="overview">
            <p>In the on-line portion, . . . </p>
            <table border="0">
                <tr>
                    <td align="right">3pm-3:10pm</td><td>Welcome!</td>
                </tr>
                . . .
            </table>
    </html>
    <vertical url_name="in_class_ora"></vertical>
    <problem url_name="res_survey" filename="res_survey"
        display_name="Survey: What next?">
    </problem>
</sequential>
```

The learner sees this sequential as follows.



In the on-line portion, we examined a way we used technology to allow efficient implementation of one theory from learning science – constructive learning – in edX. In designing the edX platform, we applied many such techniques. We took aspects of mastery learning, project-based learning, gamification and others. Other platforms have sophisticated techniques for targeting specific student misconceptions, enabling a range of student social experiences, assessing teacher performance, and hundreds of other research-based techniques. We would like to give you a chance to practice with designing software to enable good pedagogy.

You've either assigned yourself or we've assigned you to teams of three. The groupings are available in the wiki.

CHAPTER 14

Example of OLX for a Studio Course

You can export a course from edX Studio. When you export the course, you download a .tar.gz file with the OLX (open learning XML) course content. You can then extract the course OLX files for use with local tools or a source control system such as GitHub.

As explained in this document, OLX provides for flexibility in how you structure your course content. However, edX Studio exports OLX content in a specific structure.

This section documents the how edX Studio currently exports courses, so that you can understand and manually navigate through the structure of exported courses.

Note: The format of Studio course exports is subject to change. As a result, any tools that you create specifically for the current Studio export format might not work for future versions. To avoid problems with manually authored OLX courses, we strongly recommend that you base any scripts that you create on the OLX format specification rather than on the current Studio export format.

In this section, we use a course that is part of the edX Platform, the Manual Testing course, as an example of the OLX course exported from edX Studio. We examine the overall structure of the Manual Testing course, as well as how the courseware is defined.

The files for the Manual Testing course are stored in GitHub, so you can explore how the course is structured for yourself.

For more information, see the following topics.

The Structure of the Manual Testing Course

This section describes the structure of the Manual Testing course.

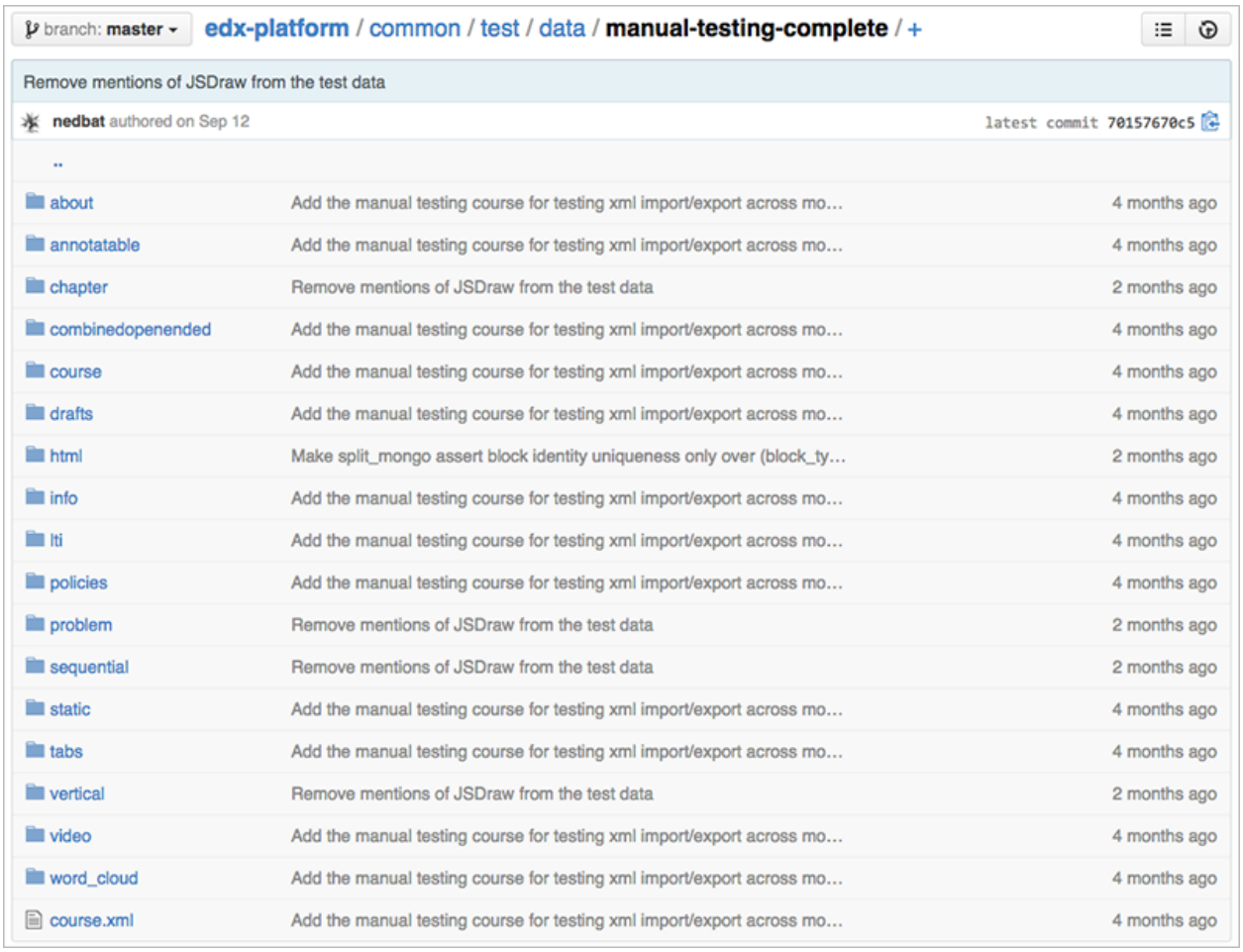
For information about how a generic OLX (open learning XML) course is structured, see *OLX Course Structure*.

For information about how a non-Studio OLX course can be structured, see *The Structure of edX-Insider*.

Note: The structure and content of the manual testing course can change without corresponding updates being made to this reference guide.

Manual Testing Course and Directory File Structures

All files and subdirectories that comprise the Manual Testing course are stored in the manual-testing-complete directory in the edx-platform Git repository.



branch: master ▾ edx-platform / common / test / data / manual-testing-complete / +

Remove mentions of JSDraw from the test data

nedbat authored on Sep 12 latest commit 70157670c5

..		
about	Add the manual testing course for testing xml import/export across mo...	4 months ago
annotatable	Add the manual testing course for testing xml import/export across mo...	4 months ago
chapter	Remove mentions of JSDraw from the test data	2 months ago
combinedopenended	Add the manual testing course for testing xml import/export across mo...	4 months ago
course	Add the manual testing course for testing xml import/export across mo...	4 months ago
drafts	Add the manual testing course for testing xml import/export across mo...	4 months ago
html	Make split_mongo assert block identity uniqueness only over (block_ty...	2 months ago
info	Add the manual testing course for testing xml import/export across mo...	4 months ago
lti	Add the manual testing course for testing xml import/export across mo...	4 months ago
policies	Add the manual testing course for testing xml import/export across mo...	4 months ago
problem	Remove mentions of JSDraw from the test data	2 months ago
sequential	Remove mentions of JSDraw from the test data	2 months ago
static	Add the manual testing course for testing xml import/export across mo...	4 months ago
tabs	Add the manual testing course for testing xml import/export across mo...	4 months ago
vertical	Remove mentions of JSDraw from the test data	2 months ago
video	Add the manual testing course for testing xml import/export across mo...	4 months ago
word_cloud	Add the manual testing course for testing xml import/export across mo...	4 months ago
course.xml	Add the manual testing course for testing xml import/export across mo...	4 months ago

Top-level Directory

The manual-testing-complete directory in the edx-platform Git repository contains the `course.xml` file as well as XBlock and Platform directories.

As a course exported for edX Studio, the `course.xml` file does not contain the courseware content directly. All chapters, sequentials, verticals, and XBlocks are defined in separate files and referenced from their parent files.

The `course.xml` file contains a single line that references the courseware structure file in the `course` directory.

```
<course url_name="2014" org="ManTestX" course="ManTest1"/>
```

The value of `url_name`, `2014`, matches the filename of the courseware structure file, `2014.xml`.

The Courseware Structure File

The courseware structure is defined in the 2014.xml file in the `course` directory.

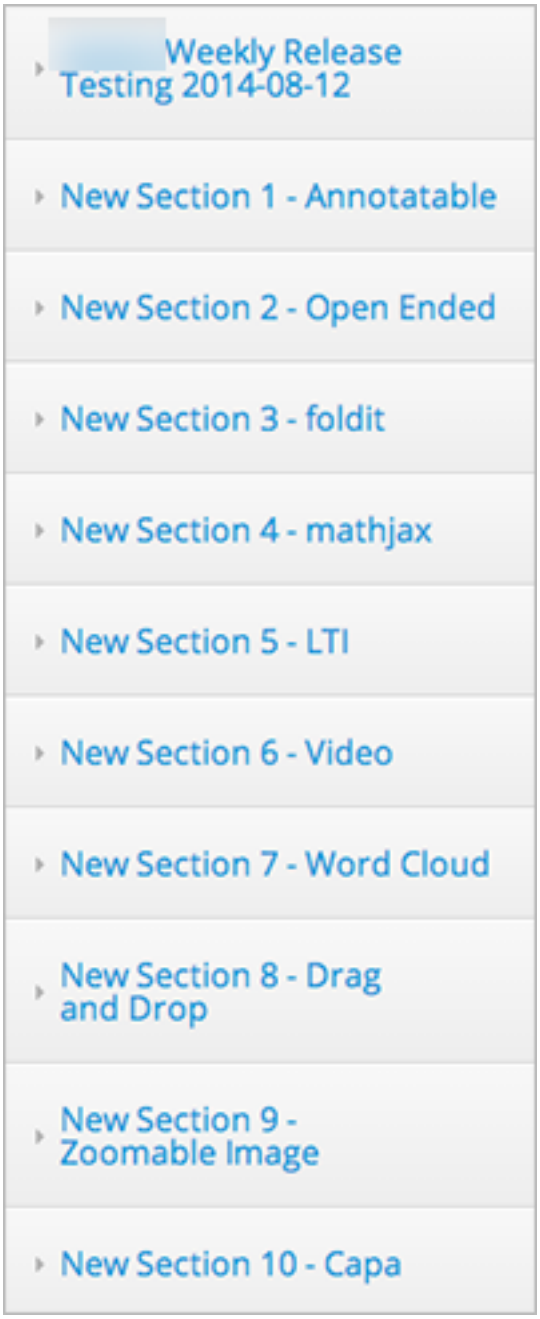
The `2014.xml` file specifies advanced settings as attributes of the `course` element, and lists the chapters (or sections) that make up the courseware.

```
<course advanced_modules="[&amp;quot;annotatable&amp;quot;,
  &amp;quot;combinedopenended&amp;quot;, &amp;quot;peergrading&amp;quot;, &amp;quot;lti&amp;quot;,
```

```
  &quot;word_cloud&quot;]" display_name="Manual Smoke Test Course 1"
  lti_passports="[&quot;ims:12345:secret&quot;]"
  pdf_textbooks="[{&quot;tab_title&quot;: &quot;An Example Paper&quot;,
  &quot;id&quot;: &quot;0An_Example_Paper&quot;, &quot;chapters&quot;:
  [{&quot;url&quot;: &quot;/static/1.pdf&quot;, &quot;title&quot;:
  &quot;Introduction &quot;}]}]" show_calculator="true" show_chat="true"
  start="2014-06-26T00:00:00Z">
    <chapter url_name="a64a6f63f75d430aa71e6ce113c5b4d2"/>
    <chapter url_name="d68c2861c10a4c9d92a679b4cfc0f924"/>
    <chapter url_name="ab97a6dbfafd48868c36bed4c8c5391d"/>
    <chapter url_name="5bb7a5ab824f460580a756a4f347377c"/>
    <chapter url_name="ce2fd991d84b4a5ca75350eb8e350627"/>
    <chapter url_name="be8a64868f2f460ea490e001a25e588d"/>
    <chapter url_name="3d216a50442f4cd5a1d4171c68f13f58"/>
    <chapter url_name="a0178ff300514e24829e239604dce12c"/>
    <chapter url_name="2df1fe87253549199f30cabb19e14b7c"/>
    <chapter url_name="60989ac1589241ed9dbca0f2070276fd"/>
    <wiki slug="ManTestX.ManTest1.2014"/>
</course>
```

For each `chapter` element, the value of the `url_name` attribute matches the name of the XML file in the `chapter` directory. For example, the first `chapter` element's `url_name` attribute, `a64a6f63f75d430aa71e6ce113c5b4d2`, matches `a64a6f63f75d430aa71e6ce113c5b4d2.xml` in the `chapter` directory.

Learners see the chapters that the `course.xml` file defines in the LMS on the **Course** page in the course navigation pane.



Chapter Files

The structure of each chapter, or section, in the courseware is defined in the XML file in the `chapter` directory.

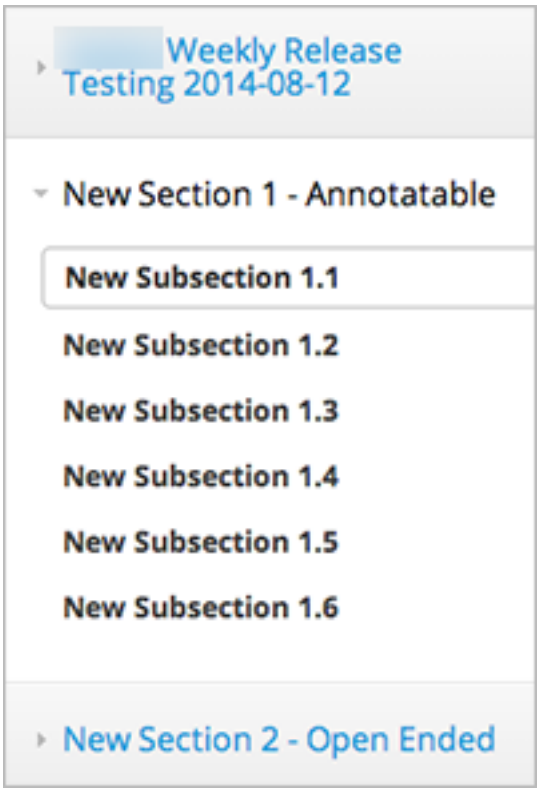
Each chapter file specifies the sequentials, or subsections, in the chapter. An example follows.

```
<chapter display_name="New Section 1 - Annotatable">
  <sequential url_name="d7d631967807476485aa26ba0c39a992"/>
  <sequential url_name="f09502cf408742c2aa3c92705ab1dce7"/>
  <sequential url_name="0e86943b2cb54a56a1a14c13da3f388d"/>
  <sequential url_name="948737f132254c2aa65f6024edee7e68"/>
  <sequential url_name="f9372e3b199a4986a46c8d18e094b931"/>
```

```
  <sequential url_name="d912a92ed03d4f818661a1636b8a6f9b"/>
</chapter>
```

For each `sequential` element, the value of the `url_name` attribute matches the name of the XML file in the `sequential` directory. For example, the first `sequential` element's `url_name` attribute, `7d631967807476485aa26ba0c39a992`, matches `7d631967807476485aa26ba0c39a992.xml` in the `sequential` directory.

Learners see the sequentials that the chapter file defines in the LMS on the **Course** page in the course navigation pane.



Sequential Files

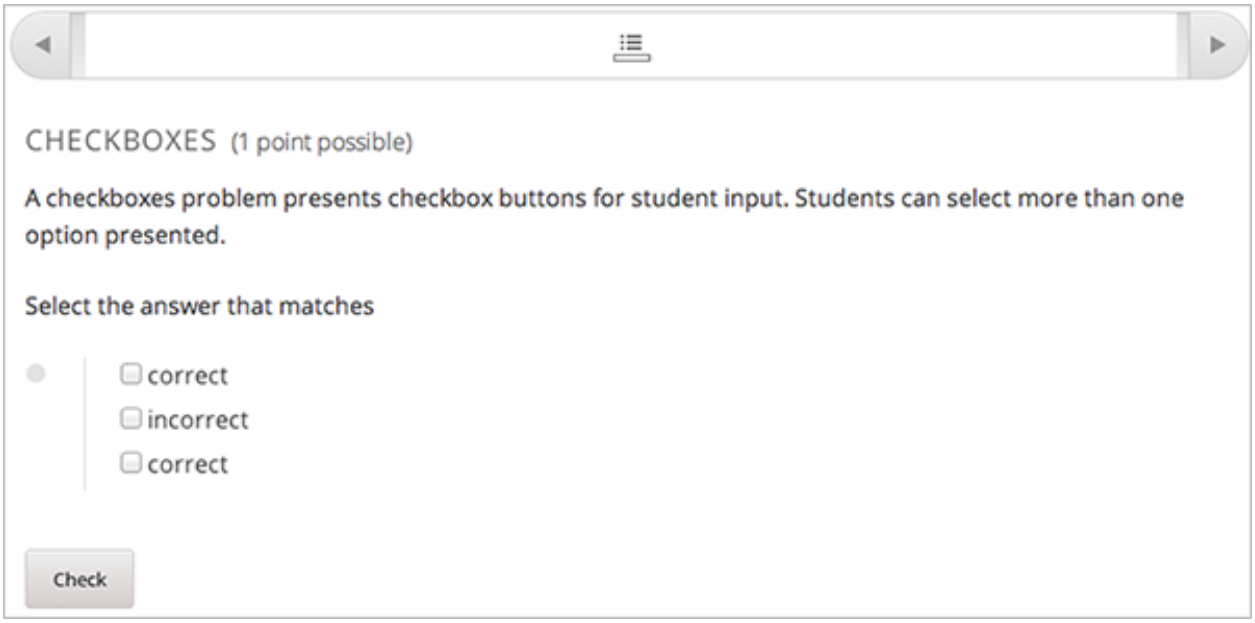
The structure of each sequential, or subsection, in the courseware is defined in the XML file in the `sequential` directory.

Each sequential file specifies the verticals, or units, in the subsection. An example follows.

```
<sequential display_name="New Subsection 10.4">
  <vertical url_name="e81c7ddcf5434387a2a6163ca973520c"/>
</sequential>
```

For each `vertical` element, the value of the `url_name` attribute matches the name of the XML file in the `vertical` directory. For example, the `vertical` element's `url_name` attribute, `e81c7ddcf5434387a2a6163ca973520c`, matches `e81c7ddcf5434387a2a6163ca973520c.xml` in the `vertical` directory.

Learners see the verticals that the sequential file defines in the LMS on the **Course** page in the unit navigation bar. The following example shows a sequential with one vertical, which has one XBlock.



Vertical Files

The structure of each vertical, or unit, in the courseware is defined in the XML file in the `vertical` directory.

Each vertical file specifies the XBlocks, or components, in the unit. For example, the following vertical contains one problem:

```
<vertical display_name="checkbox ">
  <problem url_name="a473cecce312487a8339995bde24be53"/>
</vertical>
```

Each `vertical` element contains a child element for each XBlock in the vertical.

Learners see the XBlocks that the vertical file defines in the LMS.

For each XBlock, the value of the `url_name` attribute matches the name of the XML file in the XBlock directory. The XBlock directory name is specific to the type of XBlock and matches the XML element name. The following XBlock types are included in the Manual Testing course.

- `annotable`
- `combineopenended`
- `html`
- `lti`
- `problem`
- `video`
- `word_cloud`

Other XBlock Files

XBlock files contain the actual content learners engage with in the learning management system.

The root element of an XBlock file is the type of XBlock, as well as the parent directory name. For example, the root element of files in the `html` directory is `html`.

An example of a problem XBlock follows.

```
<problem display_name="Blank Common Problem" markdown="Capital of France is
  Paris:&#10;&#10;[[false, (true)]]&#10;">
  <p>Capital of France is Paris:</p>
  <optionresponse>
    <optioninput options="('false','true')" correct="true"/>
  </optionresponse>
</problem>
```

Learners see the problem in the vertical (or unit) page as follows.

Platform Directories

The Manual Testing course includes platform directories to support non-courseware parts of the OLX course. For more information, see *edX Platform Directories*.

CHAPTER 15

Open Learning XML Uses

Use OLX with edX Studio

TBD

Build a Course in OLX and Deploy to the edX LMS

TBD

Convert Content in Other Formats to OLX

TBD

CHAPTER 16

Draft Course Content

TBD

CHAPTER 17

Glossary

A - C - D - E - F - G - H - I - K - L - M - N - O - P - R - S - T - V - W - XYZ

Note: Most of the links to documentation provided in this glossary are to the Building and Running an edX Course guide, for edX partners. Many of the same topics are available in the Open edX version of this guide, Building and Running an Open edX Course.

A

A/B Test

See Content Experiment.

About Page

The course page that provides potential learners with a course summary, prerequisites, a course video and image, and important dates.

For more information, see The Course About Page.

Accessible Label

In a problem component, you use special formatting to identify the specific question that learners will answer by selecting options or entering text or numeric responses.

This text is referred to as the accessible label because screen readers read all of the text that you supply for the problem and then repeat the text that is identified with this formatting immediately before reading the answer choices for the problem. This text is also used by reports and Insights to identify each problem.

All problems require accessible labels.

For more information, see The Simple Editor.

Advanced Editor

An OLX (open learning XML) editor in a problem component that allows you to create and edit any type of problem. For more information, see The Advanced Editor.

Assignment Type

The category of graded student work, such as homework, exams, and exercises. For more information, see Establishing a Grading Policy For Your Course.

C

CAPA Problem

A CAPA (computer assisted personalized approach) problem refers to any of the problem types that are implemented in the edX platform by the `capa_module` XBlock. Examples range from text input, drag and drop, and math expression input problem types to circuit schematic builder, custom JavaScript, and chemical equation problem types.

Other assessment methods are also available, and implemented using other XBlocks. An open response assessment is an example of a non-CAPA problem type.

Certificate

A document issued to an enrolled learner who successfully completes a course with the required passing grade. Not all edX courses offer certificates, and not all learners enroll as certificate candidates. For information about setting up certificates for your course, see Setting Up Course Certificates.

Chapter

See *Section*.

Checkbox Problem

A problem that prompts learners to select one or more options from a list of possible answers. For more information, see Checkbox Problem.

Chemical Equation Response Problem

A problem that allows learners to enter chemical equations as answers. For more information, see Chemical Equation Problem.

Circuit Schematic Builder Problem

A problem that allows learners to construct a schematic answer (such as an electronics circuit) on an interactive grid. For more information, see Circuit Schematic Builder Problem.

Closed Captions

The spoken part of the transcript for a video file, which is overlaid on the video as it plays. To show or hide closed captions, you select the **CC** icon. You can move closed captions to different areas on the video screen by dragging and dropping them.

For more information, see Watching Videos on the edX Video Player.

Cohort

A group of learners who participate in a class together. Learners who are in the same cohort can communicate and share experiences in private discussions.

Cohorts are an optional feature of courses on the edX platform. For information about how you enable the cohort feature, set up cohorts, and assign learners to them, see Using Cohorts in Your Courses.

Component

The part of a unit that contains your actual course content. A unit can contain one or more components. For more information, see Developing Course Components.

Content Experiment

You can define alternative course content to be delivered to different, randomly assigned groups of learners. Also known as A/B or split testing, you use content experiments to compare the performance of learners who have been exposed to different versions of the content. For more information, see Overview of Content Experiments.

Content Library

See *Library*.

Content-Specific Discussion Topic

A category within the course discussion that appears at a defined point in the course to encourage questions and conversations. To add a content- specific discussion topic to your course, you add a discussion component to a unit. Learners cannot contribute to a content-specific discussion topic until the release date of the section that contains it. Content-specific discussion topics can be divided by cohort, so that learners only see and respond to posts and responses by other members of the cohort that they are in.

For more information, see Working with Discussion Components. For information about making content-specific discussion topics divided by cohort, see Setting up Discussions in Courses with Cohorts.

Course Catalog

The page that lists all courses offered in the edX learning management system.

Course Handouts

Course handouts are files you make available to learners on the **Home** page. For more information, see Adding Course Updates and Handouts.

course mode

See *enrollment track*.

Course Navigation Pane

The navigation frame that appears at one side of the **Course** page in the LMS. The course navigation pane shows the sections in the course. When you select a section, the section expands to show subsections. When you select a subsection, the first unit in that subsection appears on the course page.

See also *Unit Navigation Bar*.

Course Run

The term or time frame in which a specific offering of your course takes place. You set the course run when you create your course. For more information, see Creating a Course.

Course Page

The page where learners access the primary instructional materials for your course. Sections, subsections, units, and components are all accessed from the **Course** page. This page was formerly called the **Courseware** page.

Course Track

See *enrollment track*.

Courseware

In OLX (open learning XML) and in data packages, "courseware" refers to the main content of your course, consisting mainly of lessons and assessments. Courseware is organized into sections, subsections, units, and components. Courseware does not include handouts, the syllabus, or other course materials.

Note that the **Course** page was formerly called the **Courseware** page.

Course-Wide Discussion Topic

Optional discussion categories that you create to guide how learners find and share information in the course discussion. Course-wide discussion topics are accessed from the **Discussion** page in your course. Examples of course-wide discussion topics include Announcements and Frequently Asked Questions. Learners can contribute to these topics as soon as your course starts. For more information, see Managing Course Discussions and Create Course-Wide Discussion Topics.

If you use cohorts in your course, you can divide course-wide discussion topics by cohort, so that although all learners see the same topics, they only see and respond to posts and responses by other members of the cohort that they are in. For information about configuring discussion topics in courses that use cohorts, see Setting up Discussions in Courses with Cohorts.

Custom Response Problem

A custom response problem evaluates text responses from learners using an embedded Python script. These problems are also called "write-your-own-grader" problems. For more information, see Write-Your-Own-Grader Problem.

D

Data Czar

A data czar is the single representative at a partner institution who is responsible for receiving course data from edX, and transferring it securely to researchers and other interested parties after it is received.

For more information, see the EdX Research Guide.

Discussion

The set of topics defined to promote course-wide or unit-specific dialog. Learners use the discussion topics to communicate with each other and the course team in threaded exchanges. For more information, see Managing Course Discussions.

Discussion Component

Discussion topics that course teams add directly to units. For example, a video component can be followed by a discussion component so that learners can discuss the video content without having to leave the page. When you add a discussion component to a unit, you create a content-specific discussion topic. See also *Content Specific Discussion Topic*.

For more information, see Working with Discussion Components.

Discussion Thread List

The navigation frame that appears at one side of the **Discussion** page in the LMS. The discussion thread list shows the discussion categories and subcategories in the course. When you select a category, the list shows all of the posts in that category. When you select a subcategory, the list shows all of the posts in that subcategory. Select a post to read it and its responses and comments, if any.

Dropdown Problem

A problem that asks learners to choose from a collection of answer options, presented as a drop-down list. For more information, see Dropdown Problem.

E

edX101

An online course about how to create online courses. The intended audience for edX101 is faculty and university administrators.

edX Edge

edX Edge is a less restricted site than edX.org. While only edX employees and consortium members can create and post content on edX.org, any users with course creator permissions for Edge can create courses with Studio on studio.edge.edx.org, then view the courses on the learning management system at edge.edx.org.

edX Studio

The edX tool that you use to build your courses. For more information, see Getting Started with Studio.

Embargo

An embargo is an official ban on trade or commercial activity with a particular country. For example, due to U.S. federal regulations, edX cannot offer certain courses (for example, particular advanced STEM courses) on the edx.org website to learners in embargoed countries. Learners cannot access restricted courses from an embargoed country. In some cases, depending on the terms of the embargo, learners cannot access any edX courses at all.

enrollment mode

See *enrollment track*.

enrollment track

Also called **course mode**, **course track**, **enrollment mode**, or **seat type**.

The enrollment track specifies the following items about a course.

- The type of certificate, if any, that learners receive if they pass the course.
- Whether learners must verify their identity to earn a certificate, using a webcam and a government-issued photo ID.
- Whether the course requires a fee.

The edX platform offers the following enrollment tracks.

- **audit**: The default enrollment track. This track does not offer certificates, does not require identity verification, and does not require a course fee.
- **verified**: This enrollment track offers verified certificates to learners who pass the course, verify their identities, and pay a required course fee. A course that offers the verified enrollment track also automatically offers a free enrollment track, either the audit track or honor track.
- **honor**: This enrollment track offers an honor code certificate to learners who pass the course. This track does not require identity verification and does not require a fee. Note, however, that as of December 2015, edx.org no longer offers honor code certificates. For more information, see News About edX Certificates.

Exercises

Practice or practical problems that are interspersed in edX course content to keep learners engaged. Exercises are also an important measure of teaching effectiveness and learner comprehension. For more information, see Adding Exercises and Tools.

Export

A tool in edX Studio that you use to export your course or library for backup purposes, or so that you can edit the course or library directly in OLX format. See also *Import*.

For more information, see Export a Course or Export a Library.

F

Forum

See *Discussion*.

G

Grade Range

Thresholds that specify how numerical scores are associated with grades, and the score that learners must obtain to pass a course.

For more information, see Set the Grade Range.

Grading Rubric

See *Rubric*.

H

Home Page

The page that opens first every time learners access your course. You can post announcements on the **Home** page. The handout navigation sidebar appears at the side of this page. This page was formerly called the **Course Info** page.

HTML Component

A type of component that you can use to add and format text for your course. An HTML component can contain text, lists, links, and images. For more information, see Working with HTML Components.

I

Image Mapped Input Problem

A problem that presents an image and accepts clicks on the image as an answer. For more information, see Image Mapped Input Problem.

Import

A tool in Studio that you use to load a course or library in OLX format into your existing course or library. When you use the Import tool, Studio replaces all of your existing course or library content with the content from the imported course or library. See also *Export*.

For more information, see Import a Course or Import a Library.

Instructor Dashboard

A user who has the Admin or Staff role for a course can access the instructor dashboard in the LMS by selecting **Instructor**. Course team members use the tools, reports, and other features that are available on the pages of the instructor dashboard to manage a running course.

For more information, see Managing a Running Course.

K

Keyword

A variable in a bulk email message. When you send the message, a value that is specific to the each recipient is substituted for the keyword.

L

Label

See *Accessible Label*.

LaTeX

A document markup language and document preparation system for the TeX typesetting program. In edX Studio, you can Import LaTeX Code into an HTML Component.

Learning Management System (LMS)

The platform that learners use to view courses, and that course team members use to manage learner enrollment, assign team member privileges, moderate discussions, and access data while the course is running.

Learning Sequence

See *Unit Navigation Bar*.

Left Pane

See *Course Navigation Pane*.

Library

A pool of components for use in randomized assignments that can be shared across multiple courses from your organization. Course teams configure randomized content blocks in course outlines to reference a specific library of components, and randomly provide a specified number of problems from that content library to each learner.

For more information, see Working with Content Libraries and Randomized Content Blocks.

Live Mode

A view that allows the course team to review all published units as learners see them, regardless of the release dates of the section and subsection that contain the units. For more information, see Viewing Published and Released Content.

LON-CAPA

The Learning Online Network with Computer-Assisted Personalized Approach e-learning platform. The structure of CAPA problem types in the edX platform is based on the LON-CAPA assessment system, although they are not compatible.

See also *CAPA Problems*.

M

Math Expression Input Problem

A problem that requires learners to enter a mathematical expression as text, such as e=m*c^2.

For more information, see Entering Mathematical and Scientific Expressions in the *EdX Learner's Guide*.

MathJax

A LaTeX-like language that you use to write equations. Studio uses MathJax to render text input such as x^2 and sqrt(x^2-4) as "beautiful math."

For more information, see Using MathJax for Mathematics.

Module

An item of course content, created in an XBlock, that appears on the **Course** page in the edX learning management system. Examples of modules include videos, HTML-formatted text, and problems.

Module is also used to refer to the structural components that organize course content. Sections, subsections, and units are modules; in fact, the course itself is a top-level module that contains all of the other course content as children.

Multiple Choice Problem

A problem that asks learners to select one answer from a list of options. For more information, see Multiple Choice Problem.

N

Numerical Input Problem

A problem that asks learners to enter numbers or specific and relatively simple mathematical expressions. For more information, see Numerical Input Problem.

O

OLX

OLX (open learning XML) is the XML-based markup language that is used to build courses on the Open edX platform.

For more information, see What is Open Learning XML?.

Open Response Assessment

A type of assignment that allows learners to answer with text, such as a short essay and, optionally, an image or other file. Learners then evaluate each others' work by comparing each response to a *rubric* created by the course team.

These assignments can also include a self assessment, in which learners compare their own responses to the rubric, or a staff assessment, in which members of course staff evaluate learner responses using the same rubric.

For more information, see Introduction to Open Response Assessments.

P

Pages

Pages organize course materials into categories that learners select in the learning management system. Pages provide access to the course content and to tools and uploaded files that supplement the course. Links to each page appear in the course material navigation bar.

For more information, see Managing the Pages in Your Course.

Partner Manager

Each EdX partner institution has an edX partner manager. The partner manager is the primary contact for the institution's course teams.

Pre-Roll Video

A short video file that plays before the video component selected by the learner. Pre-roll videos play automatically, on an infrequent schedule.

For more information, see Adding a Pre-Roll Video to Your edX Course.

Preview Mode

A view that allows you to see all the units of your course as learners see them, regardless of the unit status and regardless of whether the release dates have passed.

For more information, see Previewing Draft Content.

Problem Component

A component that allows you to add interactive, automatically graded exercises to your course content. You can create many different types of problems.

For more information, see Working with Problem Components and Adding Exercises and Tools.

Progress Page

The page in the learning management system that shows learners their scores on graded assignments in the course. For more information, see Checking Your Progress in a Course in the *EdX Learner's Guide*.

Q

Question

A question is a type of post that you or a learner can add to a course discussion topic to bring attention to an issue that the discussion moderation team or learners can resolve.

For more information, see Managing Course Discussions.

R

Research Data Exchange (RDX)

An edX program that allows participating partner institutions to request data for completed edx.org courses to further approved educational research projects. Only partner institutions that choose to participate in RDX contribute data to the program, and only researchers at those institutions can request data from the program.

For more information, see Research Data Exchange.

Rubric

A list of the items that a learner's response should cover in an open response assessment. For more information, see the Rubric topic in Introduction to Open Response Assessments.

See also *Open Response Assessment*.

S

seat type

See *enrollment track*.

Section

The topmost category in your course outline. A section can represent a time period or another organizing principle for course content. A section contains one or more subsections.

For more information, see Developing Course Sections.

Sequential

See *Subsection*.

Short Course Description

The description of your course that appears on the edX Course List page.

For more information, see Describe Your Course.

Simple Editor

The graphical user interface in a problem component that contains a toolbar for adding Markdown formatting to the text you supply. The simple editor is available for some problem types. For more information, see Editing a Problem in Studio.

Single Sign-On (SSO)

SSO is an authentication service that allows a user to access multiple related applications, such as Studio and the LMS, with the same username and password. The term SSO is sometimes used to refer to third party authentication, which is a different type of authentication system. For information about third party authentication, see *Third Party Authentication*.

Split Test

See *Content Experiment*.

Subsection

A division in the course outline that represents a topic in your course, such as a lesson or another organizing principle. Subsections are defined inside sections and contain units.

For more information, see Developing Course Subsections.

T

Text Input Problem

A problem that asks learners to enter a line of text, which is then checked against a specified expected answer.

For more information, see Text Input Problem.

Third Party Authentication

A system-wide configuration option that allows users who have a username and password for one system, such as a campus or institutional system, to log in to that system and automatically be given access to the LMS. These users do not enter their system credentials in the LMS.

For more information about how system administrators can integrate an instance of Open edX with a campus or institutional authentication system, see Enabling Third Party Authentication.

Transcript

A text version of the content of a video. You can make video transcripts available to learners.

For more information, see Step 2. Create or Obtain a Video Transcript in Working with Video Components.

U

Unit

A unit is a division in the course outline that represents a lesson. Learners view all of the content in a unit on a single page.

For more information, see Developing Course Units.

Unit Navigation Bar

The horizontal control that appears at the top of the **Course** page in the LMS. The unit navigation bar contains an icon for each unit in the selected subsection. When you move your pointer over one of these icons, the name of the unit appears. If you have bookmarked a unit, the unit navigation bar includes an identifying flag above that unit's icon.

See also *Course Navigation Pane*.

V

Vertical

See *Unit*.

Video Component

A component that you can use to add recorded videos to your course.

For more information, see Working with Video Components.

W

Whitelist

In edX courses, a whitelist is a list of learners who are being provided with a particular privilege. For example, whitelisted learners can be specified as being eligible to receive a certificate in a course, regardless of whether they would otherwise have qualified based on their grade.

In the grade report for a course, whitelisted learners have a value of "Yes" in the **Certificate Eligible** column, regardless of the grades they attained. For information about the grade report, see Interpreting the Grade Report.

Wiki

The page in each edX course that allows both learners and members of the course team to add, modify, or delete content. Learners can use the wiki to share links, notes, and other helpful information with each other. For more information, see Using the Course Wiki.

XYZ

XBlock

EdX's component architecture for writing course components: XBlocks are the components that deliver course content to learners.

Third parties can create components as web applications that can run within the edX learning management system. For more information, see Open edX XBlock Tutorial.

XSeries

A set of related courses in a specific subject. Learners qualify for an XSeries certificate when they pass all of the courses in the XSeries. For more information, see XSeries Programs.