Class 1: Course Introduction

Table of Contents / Agenda

1 Introduction to Org-Coursepack

The Org-Coursepack provides a template for developing and managing teaching materials using Org mode, a major mode in GNU Emacs.

1.1 Advantages for Instructors

- First, Org mode and modular design allow for more effective and efficient content creation.
  - Content updates get propagated across courses, semesters, and sections, minimizing the potential for inconsistencies
  - Minimizes redundancy when sharing content across courses, semesters, and sections
- Second, instructors can enjoy the benefits of having a flexible export system and an output-specific export option.
  - Consistent content across multiple output formats
    - Slides (e.g., via reveal.js or Beamer backends)
    - Handouts (e.g., via the LaTeX or reStructuredText backends)
  - Selective formatting and presentation of components depending on output format
- Third, the template contains a) utility functions written in Emacs Lisp, b) shortcuts to Org mode functions, and c) pre-built tree structures, which allow automation of many tasks including:
  - Automatic class numbering
  - Automatic creation of key content including (but not limited to)

- course schedule for syllabi;
- agenda of lecture materials; and
- exam keys.

1.2 Advantages for Students

- Consistent, properly-formatted, and strategically presented course materials add to student engagement
- Availability of materials that are easier to digest and review outside the classroom

1.3 Requirements

- Org-Coursepack is a tool based on Org mode, a major mode in GNU Emacs. It does not, however, require extensive previous experience with either, unless the user wants to make changes to pre-built scripts. Only the following basic knowledge is needed:

  Emacs Text editing using Emacs

  Org mode Org mode markup syntax. Org mode uses a straightforward markup language similar to Markdown and reStructuredText, and thus, it will be easy to learn for any user who is familiar with other markup languages.

2 Quickstart Guide

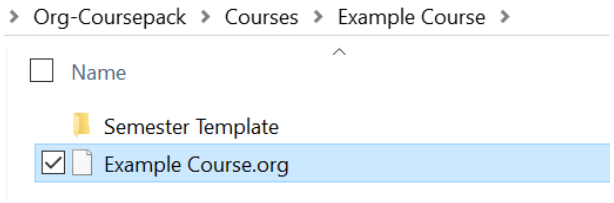
This quickstart guide will show you, step-by-step, how to create your course with `Org-Coursepack`. We focus on creating syllabi and lectures, as they typically make up the majority of course content. Other course materials, such as assignments and exams, can be created in similar ways.

2.1 Installation

Follow instructions on README to install `Org-Coursepack` along with its dependencies.

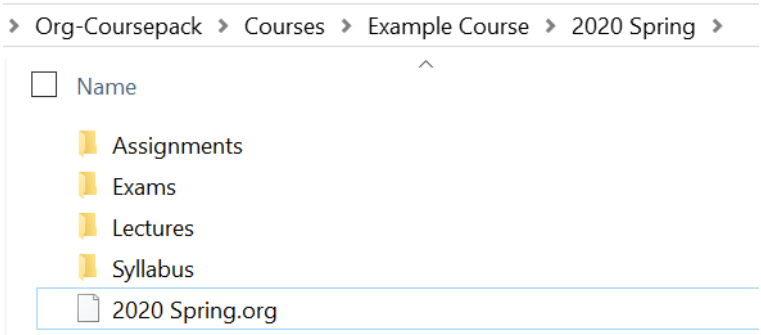
2.2 Copy and rename the `Template` directory and Org file

The quickstart guide uses `/` to refer the root path of `Org-Coursepack`. Make a copy of the `Template` directory (inside the `/Courses` directory). Change the names of the `Template` directory and the `Course Template.org` file inside the directory, so they reflect the name of the course you are creating the materials for. See example below (we changed the names to `Example Course`):



It is common for the same course to be taught across multiple semesters. Hence, the course will include a `Semester` folder for each semester it is taught. This folder includes a semester Org file, which is used to pull and organize modular course materials that can be used across courses and/or semesters. The semester Org file can also contain semester-specific content, such as class dates, classroom information, or office hours. The semester Org file is used to export course content into appropriate output formats (e.g., slides or handouts).

The example below shows that the semester is 2020 Spring. You can rename the `Semester Template` folder and the `Semester Template.org` inside that folder to reflect the semester in which you are teaching that course.



2.3 Set local variables permissions in the semester Org file

When you open `2020 Spring.org` for the first time, it will show you the following warning about local variables:

This warning asks what you would like to do about the pre-set options contained in the file. For example, the option `org-confirm-elisp-link-function` is currently set to `nil`, which means that, when links containing Emacs scripts are clicked (e.g., for exporting), Emacs will not ask for your confirmation before the action is taken. You can type `!` to permanently set these options.

When the file is open, you will see the template for constructing a course for this semester:

```
#+TITLE:     Template: Semester
#+AUTHOR:    Your Name
#+EMAIL:     Your Email
#+DESCRIPTION: Description
#+CATEGORY: Teaching

:LOCALSETUP:...

:SETUP_EXPORT:...

:COURSE_INFO:...

* Tasks [0/1]...
* Sections...
* Syllabus...
* Lectures...
* Exams                                                    :Exams:...
* Assignments                                          :Assignments:...
* Local Variables...
```

You can freely move the cursor around using the arrow keys. Many items such as such as drawers (e.g., `:LOCALSETUP:`) and subtrees (e.g., `* Sections`) are collapsible, and they are annotated with `...` when collapsed. A collapsible item under the cursor can be expanded and collapsed by pressing the `Tab` key:

```
* Tasks [0/1]...
* Sections
  :PROPERTIES:...
  * 01...
  * 02...
  * 03...
* Syllabus...
```

2.4 Rename paths to semester and course Org files specified in `#+INCLUDE` statements

The semester Org file has `#+INCLUDE` statements, which are used to pull content from other Org files (e.g., course Org file or other sections of the same semester Org file). The `#+INCLUDE` statements specify the paths to the Org files from which content are pulled. These paths should be renamed so the `#+INCLUDE` statements work properly.

Specifically, you should replace two file names in these paths.

First, replace all occurrences of `Semester Template.org` with the name of your semester Org file (in the case of our example, `2020 Spring.org`).

```
* Sections
  :PROPERTIES:...
  * 01
    :PROPERTIES:...
    :MACROS_Section_Info:...
    * Syllabus
      :PROPERTIES:...
      #+INCLUDE: "./Semester Template.org::#Syllabus" :only-contents t
  * 02...
  * 03...
```

You can use the search-and-replace feature in Emacs by pressing `M-%` (`Alt+Shift+5`) or via `Edit` -> `Replace` -> `Replace String` menu), inputting `Semester Template.org<Enter>` followed by `2020`

`Spring.org<Enter>`, and pressing `!` (replace all). Emacs will let you know how many replaces has been made.

The replace query will look like the following:

```
Query replace Semester Template.org with: 2020 Spring.org
```

As shown below, all occurrences of `Semester Template.org` are replaced by `2020 Spring.org`:

```
* Sections
  :PROPERTIES:...
** 01
   :PROPERTIES:...
   :MACROS_Section_Info:...
*** Syllabus
    :PROPERTIES:...
    #+INCLUDE: "./2020 Spring.org::#Syllabus" :only-contents t
** 02...
** 03...
```

Repeat the same process for the template Org file. That is, replace `Course Template.org` with the name of your course Org file.

```
* Syllabus
  :PROPERTIES:...
** Tasks [0/1]                                                    :noexport:...
** Intro                                                           :ignore:...
** Course Description
   #+INCLUDE: "../Course Template.org::#Syllabus/Course Description" :only-contents t
** Course Prerequisites...
```

```
Query replace Course Template.org with: Example Course.org
```

As shown below, `Course Template.org` is replaced by `Example Course.org`:

```
* Syllabus
  :PROPERTIES:...
** Tasks [0/1]                                                    :noexport:...
** Intro                                                           :ignore:...
** Course Description
   #+INCLUDE: "../Example Course.org::#Syllabus/Course Description" :only-contents t
** Course Prerequisites...
```

2.5 Inputting course information

The first few lines of the semester Org file (see `2020 Spring.org`) contain the file metadata, such as the `#+TITLE:` and `#+DESCRIPTION:` of the file. Expanding the `:COURSE_INFO:` drawer will reveal several macros which have the course metadata; the `COURSE` macro is for specifying the name of the course, `COURSE_NUM` is for specifying the course ID, and so on. They currently have filler values as shown in the image below.

```
:SETUP_EXPORT:...

:COURSE_INFO:
#+MACRO: COURSE Template
#+MACRO: COURSE_NUM COURSE 0000
#+MACRO: SEMESTER Semester
#+MACRO: OFFICE_HOURS Tue 3:30-4:30pm
:END:
```

You can fill the macro values with your own course information as shown in the image below.

```
:SETUP_EXPORT:...

:COURSE_INFO:
#+MACRO: COURSE Example Course
#+MACRO: COURSE_NUM COURSE 0000
#+MACRO: SEMESTER 2020 Spring
#+MACRO: OFFICE_HOURS Tue 3:30-4:30pm
:END:
```

2.6 Preparing your syllabus

To reduce reduncancy, the top level `* Syllabus` tree simply pulls information from the course Org file (for course-related information common across sections and semesters; e.g., course description), the semester Org file (for semester-specific information; e.g., academic year or office hours), and section-level macros (which contain section-specific information; e.g., class time and location).

Here, we show how to use a simple command to automatically generate/update a class schedule from the list of classes and their metadata in the `* Lectures` top-level tree in the semester Org file, and how to export your syllabi.

2 Syllabus Top-level Tree of the documentation contains detailed information about how to change syllabus content to fit your own course.

2.6.1 Class schedule

You can include in your syllabus a class schedule, which is a table that shows a list of class dates, class numbers, and class titles, as well as exam dates and assignment due dates. Navigate to the `* Syllabus/Class Schedule` subtree and place your cursor on the line that starts with `#+BEGIN: columnview`. You can view the class schedule by expanding the `columnview`:

```
* Class Schedule                                                    :newpage:
  :PROPERTIES:...
  :LOGBOOK:...

  #+BEGIN: columnview :hlines 1 :id Lectures :maxlevel 2 :skip-empty-rows t
  | Date                                      | Class | Topic                  |
  |-------------------------------------------+-------+------------------------|
  #+TBLFM: ::@1$>=(string("Topic"))::$2='(replace-regexp-in-string "-" "" $2)::
  #+END:
```

Then, with the cursor in place, press `C-c C-c` (hit `c` twice while holding down the `CTRL` key) to generate/update the class schedule with the most recent information. Any changes made to the order or names of classes will automatically be reflected when the instructor updates the class schedule with the `C-c C-c` command. See an example of updated class schedule below.

```
* Class Schedule                                                    :newpage:
  :PROPERTIES:...
  :LOGBOOK:...

  #+BEGIN: columnview :hlines 1 :id Lectures :maxlevel 2 :skip-empty-rows t
  | Date                                      | Class | Topic                  |
  |-------------------------------------------+-------+------------------------|
  | [2018-08-28 Tue]                          |     1 | Course Introduction    |
  |                                           |       | _Assignment 1 Due_     |
  | [2018-08-30 Thu]                          |     2 | _Exam 1_               |
  | [2018-11-21 Wed]--[2018-11-25 Sun]        |       | *Thanksgiving Holiday* |
  | {{{property(SECTION_DATE_FINAL_EXAM)}}}   |       | _Final Exam_           |
  #+TBLFM: ::@1$>=(string("Topic"))::$2='(replace-regexp-in-string "-" "" $2):
  #+END:
```

> Note that the `org-coursepack-update-lecture-metadata` script described below will automatically update this table upon execution. Hence, in general manual updating of the table is not needed.

2.6.2 Exporting a syllabus

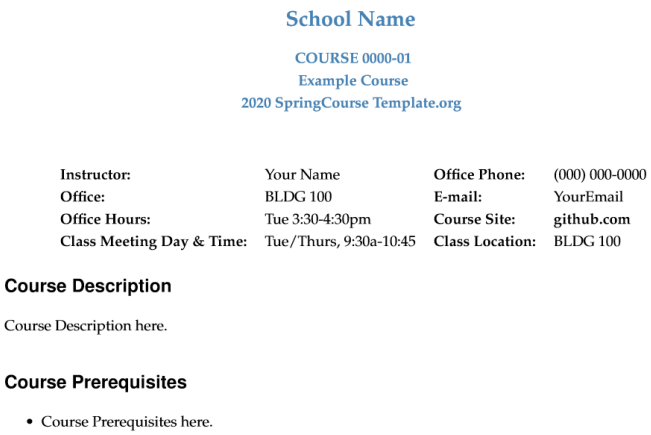
Syllabus exporting occurs at the `Syllabus` subtree under each section's headline (e.g, `* Sections/01/Syllabus`) to enable passing on section-specific information through macros, which are defined in `:PROPERTIES:` and `:MACROS_Section_Info:` drawers of the section subtree (see the example below).

```
* Sections
  :PROPERTIES:...
** 01
   :PROPERTIES:
   :SECTION: 01
   :SECTION_DATE: Tue/Thurs, 9:30a-10:45
   :SECTION_DATE_FINAL_EXAM: {{{DATE_FINAL_EXAM_01}}}
   :SECTION_LOC: BLDG 100
   :CUSTOM_ID: Sections/01
   :END:
   :MACROS_Section_Info:
   #+MACRO: DATE_FINAL_EXAM_01 [2018-12-16 Sun 13:00] - 4:00PM
   :END:
**** * Syllabus...
```

To export the syllabus in our example, navigate to the `* Sections/01/Syllabus` headline, then expand its `:PROPERTIES:` drawer. When the drawer is expanded, you will see a clickable link named `LaTeX` (see image below). First, make sure to save your file so all your changes are written to the disk. Then, clicking this link will export the syllabus for Section 1 to a PDF file using LaTeX. You can find the exported file in the `Syllabus` sub-directory. You can also click the `PDF` link in the same `:PROPERTIES:` drawer (see image below) to open the exported file.

```
* Sections
  :PROPERTIES:...
** 01
   :PROPERTIES:...
   :MACROS_Section_Info:...
**** * Syllabus
     :PROPERTIES:
     :EXPORT_TITLE: {{{COURSE_NUM}}}-{{{property(SECTION)}}} Syllabus
     :EXPORT_FILE_NAME: ./Syllabus/Syllabus (Section 1)
     :EXPORT_TO: LaTeX
     :OUTPUT_VIEW: PDF
     :END:
     #+INCLUDE: "./2020 Spring.org::#Syllabus" :only-contents t
** * 02...
```

See the screenshots of the exported syllabus (the first and the class schedule pages) below.



School Name

COURSE 0000-01
Example Course
2020 SpringCourse Template.org

Instructor:	Your Name	**Office Phone:**	(000) 000-0000
Office:	BLDG 100	**E-mail:**	YourEmail
Office Hours:	Tue 3:30-4:30pm	**Course Site:**	**github.com**
Class Meeting Day & Time:	Tue/Thurs, 9:30a-10:45	**Class Location:**	BLDG 100

Course Description

Course Description here.

Course Prerequisites

- Course Prerequisites here.

Class Schedule

Date	Class	Topic
2018-08-28 Tue	1	Course Introduction
		Assignment 1 Due
2018-08-30 Thu	2	Exam 1
2018-11-21 Wed–2018-11-25 Sun		**Thanksgiving Holiday**
2018-12-16 Sun 13:00 - 4:00PM		Final Exam

Disclaimer

- The class schedule is subject to change (except for the exam dates)

2.7 Preparing your lectures

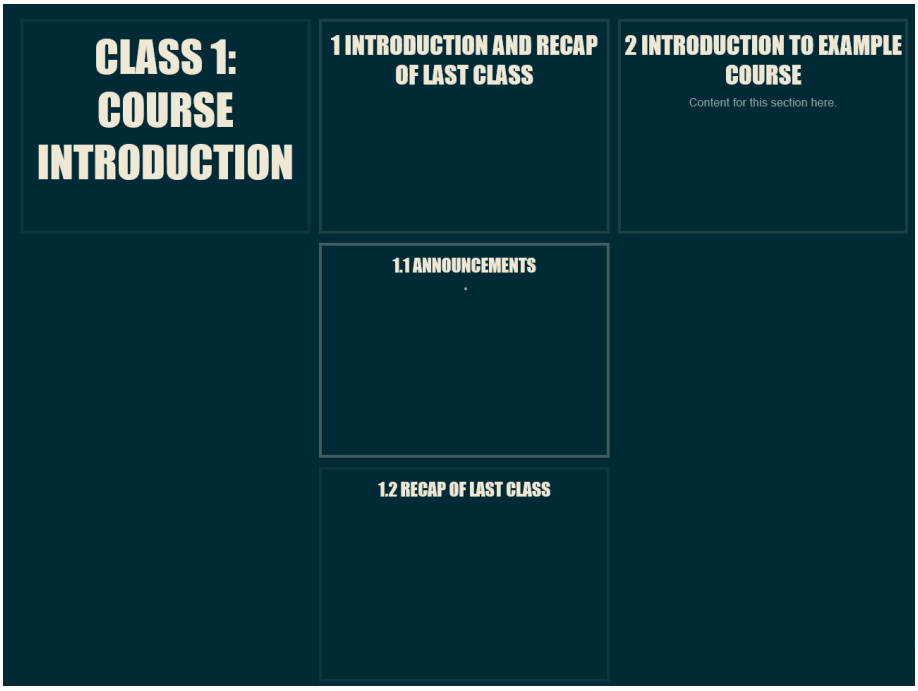
The Quickstart guide will begin by introducing the exporting functions to help beginners visualize the lecture slides and handouts generated with the `Org-Coursepack`.

2.7.1 Exporting slides and handouts

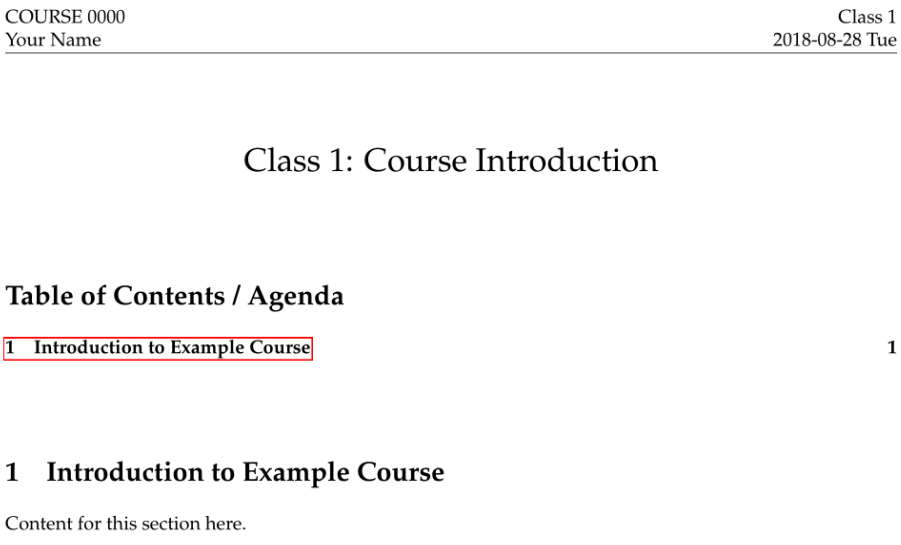
Each lecture subtree, in its `:PROPERTIES:` drawer, has clickable links for export functions (see image below). The `reveal.js` link is for exporting the lecture to html `reveal.js` slides. The `LaTeX` link is for exporting the lecture to a document-like handout (as opposed to scaled-down slides) in a PDF file format. Clicking the `HTML` and `PDF` links will open the corresponding exported file. The exported files can be located in the `Lectures` sub-directory of the semester folder. You can also click on the `HTML` and `PDF` clickable links below the export buttons (on the `:OUTPUT_VIEW:` property) to view the exported files.

```
* Course Introduction
  :PROPERTIES:
  :CLASS:    1
  :EXPORT_TITLE: Class {{{property(CLASS)}}}: {{{property(ITEM)}}}
  :EXPORT_FILE_NAME: ./Lectures/01 Course Introduction
  :DATE:     [2018-08-28 Tue]
  :EXPORT_TO:  reveal.js | Beamer | LaTeX
  :OUTPUT_VIEW: HTML | PDF
  :END:
** * Tasks [0/1]                                                :noexport:...
```

The following image shows an example of an exported html `reveal.js` slides (the slide overview mode).



The image below shows an example of an exported PDF handout.



COURSE 0000 Class 1
Your Name 2018-08-28 Tue

Class 1: Course Introduction

Table of Contents / Agenda

1 Introduction to Example Course 1

1 Introduction to Example Course

Content for this section here.

As you can see, class materials under subtrees with the `slideonly` (`handoutonly`) tag are not exported in the LaTeX (reveal.js) output. The tags allow you to specify materials you want to show only in lecture slides and not in handouts (e.g., in-class announcements) or only in handouts and not in lecture slides (e.g., supplementary information about a topic).

See Exporting Slides and Handouts for more information about exporting lectures.

2.7.2 Adding new classes/lectures and updating their metadata

The example `* Lectures` tree, as shown below, has only one lecture (i.e., `Course Introduction`). Lets try adding two additional lectures. (Note that subtrees with the `skipcount` tag are not actual lectures– they are either subtrees with auxiliary information, such as class dates and instructor's tasks, or subtrees for non-lecture events such as assignment due dates and holidays.)

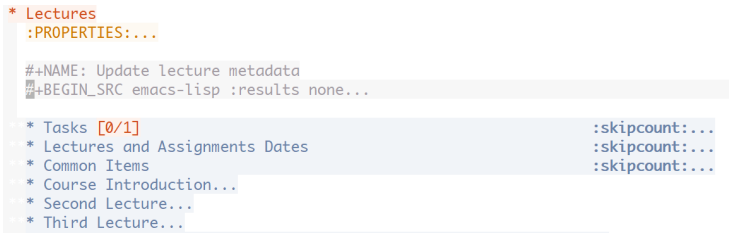
```
* Lectures
  :PROPERTIES:...

  #+NAME: Update lecture metadata
  #+BEGIN_SRC emacs-lisp :results none...

  * Tasks [0/1]                                                    :skipcount:...
  * Lectures and Assignments Dates                                 :skipcount:...
  * Common Items                                                   :skipcount:...
  * Course Introduction...
  * _Assignment 1 Due_                          :Assignment:skipcount:...
  * _Exam 1_                                                          :Exam:...
  * *Thanksgiving Holiday*                                :Holiday:skipcount:...
  * _Final Exam_                                                   :skipcount:...
```

To add two additional lectures, copy and paste the `Course Introduction` subtree twice. Then, change the names of the two additional lecture subtrees. In the example, we will simply call them `Second Lecture` and `Third Lecture`:

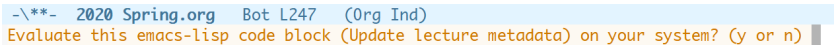
```
* Lectures
  :PROPERTIES:...

  #+NAME: Update lecture metadata
  #+BEGIN_SRC emacs-lisp :results none...

  * Tasks [0/1]                                                    :skipcount:...
  * Lectures and Assignments Dates                                 :skipcount:...
  * Common Items                                                   :skipcount:...
  * Course Introduction...
  * Second Lecture
    :PROPERTIES:...
    * Tasks [0/1]                                                      :noexport:...
    * Handout heading                                  :handoutonly:ignore:...
    * Introduction and Recap of Last Class                           :slideonly:...
    * Introduction to {{{COURSE}}}...
    * Class Recap                                                    :slideonly:...
  * Third Lecture
    :PROPERTIES:...
    * Tasks [0/1]                                                      :noexport:...
    * Handout heading                                  :handoutonly:ignore:...
    * Introduction and Recap of Last Class                           :slideonly:...
    * Introduction to {{{COURSE}}}...
    * Class Recap                                                    :slideonly:...
  * _Assignment 1 Due_                          :Assignment:skipcount:...
```

Updating metadata Expanding the `:PROPERTIES:` drawer of each lecture (e.g., `Second Lecture`) will reveal class-related metadata, such as `CLASS`, `EXPORT_FILE_NAME`, and `DATE`. You can automatically update these values by running the `org-coursepack-update-lecture-metadata` script, which is defined in `/Assets/Scripts.org`. The script is set up to be called remotely, and the call statement

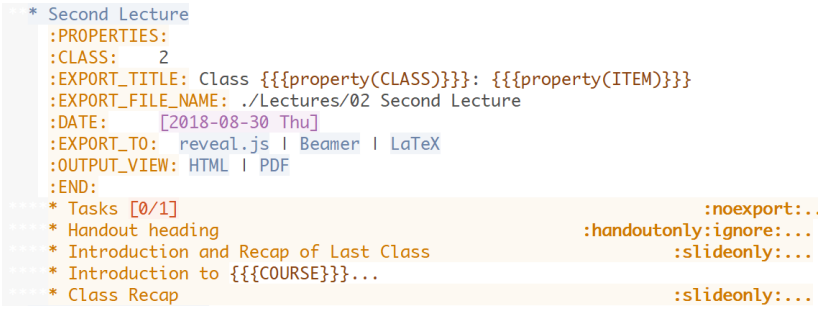
(`#+CALL: org-coursepack-update-lecture-metadata()`) is located right under the `Lectures` subtree headline (see image below)–to run the script, simply move the cursor to the call statement and press `C-c C-c`.



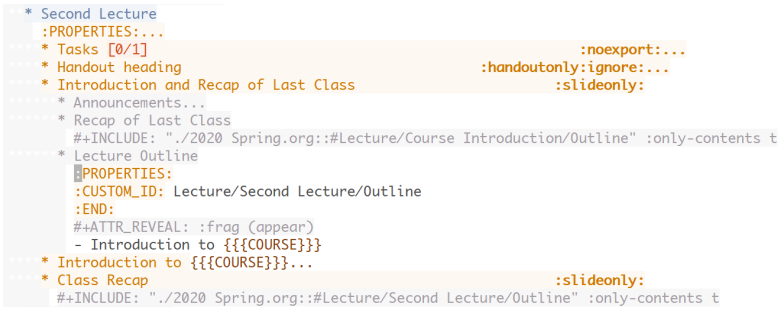
Press `y` to confirm when asked (see image below).



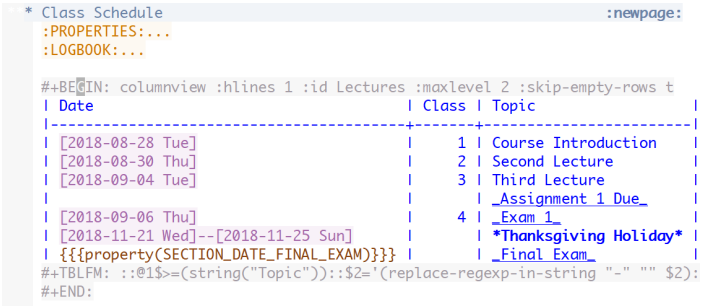
After the script finishes running, you will need to press `Shift+Tab` to reset the rendering.
As shown in the image below, the lecture metadata have been updated with appropriate values.



Creating materials for beginning and end of classes. Classes often begin with a recap of the previous lecture topics and a preview of the current lecture topics. Classes often end with a recap of the current lecture topics. Recaps and previews of lecture topics are automatically generated with the `Update lecture metadata` script described earlier by pulling the list of titles of the topic subtrees covered in a given lecture. Appropriate content are written automatically, even when the orders of topics or lectures are changed. See the screenshot below for an example.



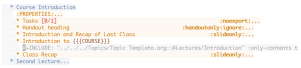
Updating class schedule in syllabus The `org-coursepack-update-lecture-metadata` is designed to automatically update the class schedule table in the syllabus with the new lecture metadata, so the table will always reflect the latest lecture metadata. See the screenshot below.



```
* Class Schedule                                                    :newpage:
  :PROPERTIES:...
  :LOGBOOK:...

  #+BEGIN: columnview :hlines 1 :id Lectures :maxlevel 2 :skip-empty-rows t
  | Date                                    | Class | Topic                  |
  |-----------------------------------------+-------+------------------------|
  | [2018-08-28 Tue]                        |     1 | Course Introduction    |
  | [2018-08-30 Thu]                        |     2 | Second Lecture         |
  | [2018-09-04 Tue]                        |     3 | Third Lecture          |
  |                                         |       | _Assignment 1 Due_     |
  | [2018-09-06 Thu]                        |     4 | _Exam 1_               |
  | [2018-11-21 Wed]--[2018-11-25 Sun]      |       | *Thanksgiving Holiday* |
  | {{{property(SECTION_DATE_FINAL_EXAM)}}} |       | _Final Exam_           |
  #+TBLFM: ::@1$>=(string("Topic"))::$2='(replace-regexp-in-string "-" "" $2):
  #+END:
```

2.7.3 Creating modular content

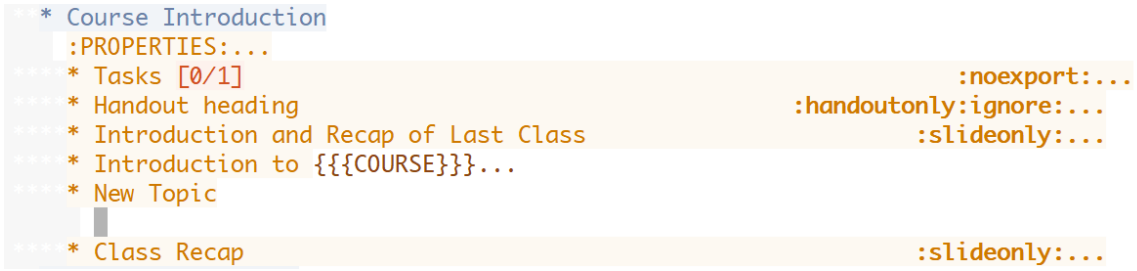
One of the biggest advantages of using the `Org-Coursepack` is that instructors can leverage Org mode's flexible inclusion functionality when developing course materials. Course materials that are stored in topic Org files can be pulled into semester Org files using the `#+INCLUDE:` statement to construct lectures (see example below).



This modular approach reduces redundancy and makes it easier to manage and update course materials. For example, any improvements made to a content can be automatically applied to all courses pulling that content. This approach is optional–instructors can also choose to put all course materials directly into a semester Org file. See the Lectures part of the documentation for more details.

2.7.4 Editing lecture content

Let's add content to a lecture. Any lecture content (e.g., Focus Groups) will belong to a topic subtree (e.g., Types of Qualitative Research Methods). In the example shown below, we simply added a `New topic` subtree at the same level as other topic subtrees (i.e., `*** New topic`).



```
* Course Introduction
  :PROPERTIES:...
*** * Tasks [0/1]                                                  :noexport:...
*** * Handout heading                                 :handoutonly:ignore:...
*** * Introduction and Recap of Last Class                          :slideonly:...
*** * Introduction to {{{COURSE}}}...
*** * New Topic

*** * Class Recap                                                   :slideonly:...
```

When creating content under the topic subtree, you can freely use Org markup language, which is similar to other popular markup languages such as Markdown and reStructuredText. The Quickstart Guide

shows several basic examples. For more detailed instructions, see the Creating Content for Slides and Handouts section of the documentation. You can also see Org manual.

Lists. Org mode uses a typical syntax (– or + for lists, `1.` for numbered lists) for lists. For example,

```
** * New Topic
****** * Lists
       Obviously you cannot use =*= to specify a list, but otherwise Org mode
       uses a typical syntax for lists. For example,

       - List
         - Nested list item 1
         - Nested list item 2


       1. Numbered list
```

Math. you can directly input LaTeX math in Org mode. For example,

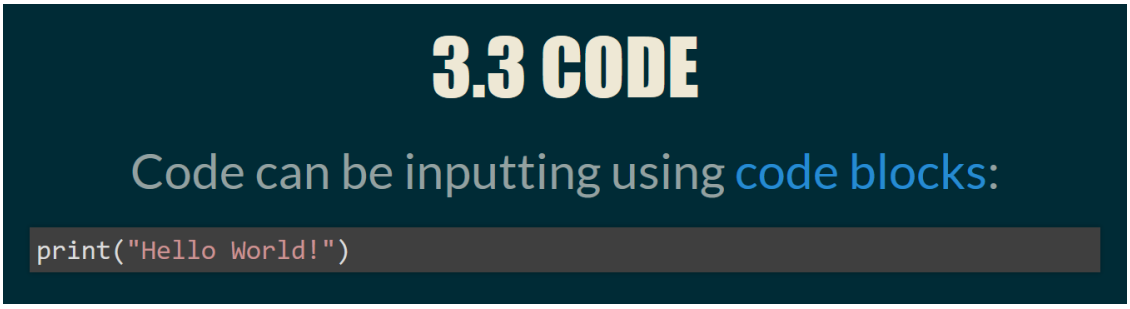
```
****** * Math
       You can directly input LaTeX math in Org mode. For example,

       \[ \cos (2\theta) = \cos^2 \theta - \sin^2 \theta \]
```

In `reveal.js` slides, math will be rendered with MathJax.

Code. Code can be inputting using code blocks, which can be easily inserted by pressing `<s` and then `<TAB>` key in Org mode. For example,

```
* Code
  Code can be inputting using code blocks:

  #+BEGIN_SRC python
  print("Hello World!")
  #+END_SRC
```

Code will be rendered with proper syntax highlighting. The following screenshots show how the above example is rendered in `reveal.js` and LaTeX outputs.



2.3 Code

Code can be inputting using code blocks:

```
print("Hello World!")
```

Slide split. Lecture content are automatically divided into different slides following the structure of the lecture subtree (e.g., content for a new topic will be presented in a new set of slides). An instructor can also force a new slide by inserting `#+REVEAL: split` into a desired location. For example,

```
* Slide split
  Users can put =#+REVEAL: split= to split a slide. For example,

  #+REVEAL: split

  This line will be shown in a new slide.
```

Fragmented contents. Fragmented contents (e.g., items in a listing being presented one after another) can be specified by putting `#+ATTR_REVEAL: :frag (appear)` before the contents that are to be fragmented. For example:

```
* Fragmented Contents
  #+ATTR_REVEAL: :frag (appear)
  - This list
  - is fragmented
  - in reveal.js slides
```

Images. Prepending `file:` to an image file path will include a local image to both lecture slides and handout. We recommend using a relative path (`../../../Assets/Images/`) for portability.

To adjust the size of the image, specify HTML attributes (e.g., `#+ATTR_HTML: :width 80%`) or LaTeX attributes (e.g., `#+ATTR_LATEX: :width 6cm`) before the image file path.

For example,

```
* Local Images
  Local images can be included in this way:

  #+ATTR_HTML: :width 20%
  #+ATTR_LATEX: :width 4cm
  file:../../../Assets/Images/Misc/affiche.png
```

Hiding specific content. Since Org mode allows embedding raw HTML and LaTeX codes, it is easy to hide specific content based on an output format. Content surrounded by `#+LATEX: \iffalse` and `#+LATEX: \fi` will not be shown in LaTeX outputs. Content surrounded by `#+REVEAL_HTML: <span hidden>` and `#+REVEAL_HTML: </span>` will not be shown in reveal.js outputs. For example,

```
Web Images / Hiding Contents
Remote images can be included by directly using its URL.

#+LATEX: \iffalse
#+ATTR_HTML: :width 20%
https://openclipart.org/image/400px/svg_to_png/125899/affiche.png

(This image is not shown in LaTeX output.)
#+LATEX: \fi

#+REVEAL_HTML: <span hidden>
However, it only works for HTML output - for LaTeX an local image is needed.

(This sentence is not shown in HTML output.)
#+REVEAL_HTML: </span>
```

The following screenshots show the exported outputs of the above content. The image is shown in the lecture slide but not in the handout.

2.6 Web Images / Hiding Contents

Remote images can be included by directly using its URL.
However, it only works for HTML output - for LaTeX an local image is needed.
(This sentence is not shown in HTML output.)

2.8 Conclusion

This concludes the quickstart guide. In addition to syllabus and lectures, the `Org-Coursepack` provides templates for common course elements, including assignments and exams. The rest of the documentation will guide you through the `Org-Coursepack` in detail.

Please let us know through GitHub issues if you have any questions or issues.

3 Overview of the Directory Structure

We present the directory structure of Org-Coursepack.

/Assets This folder contains:

- Org setup files, which include frequently used macros (e.g., for LaTex formatting).
- Supplementary course materials (if any), such as images, videos, or articles, for storage and access.

/Assets/Institutions This folder contains an institution Org file that includes institution-specific information (e.g., university policies); may have multiple Org files if teaching across multiple institutions.

/Courses Each unique course will have a subdirectory under `Courses`. A course is defined as a series of lectures occupying a given adademic calendar unit referred to as a semester. Same courses may be offered across multiple semesters. Note that a course may also have multiple sections in the same semester; for example, a Statistics 101 course may be offered to three different sets of students per semester.

/Courses/Course This folder contains:

- A course Org file that includes permanent information about the course that remains consistent across semesters (e.g., syllabus items such as learning objectives, grading schemes).
- A subfolder for each semester this course is taught.

/Courses/Course/Semester Each semester folder contains:

- A semester Org file that includes information about the course that varies by semester (e.g., classroom location, course schedule, assignment due dates). The semester Org file also pulls information from other Org files, such as course, topic, and institution Org files, to complete the course development for that semester. In other words, this is the master file that compiles all course materials for exporting.
- Subfolders are for exported course materials (if any) and are divided by type; i.e., Assignments, Lectures, Exams, and Syllabus.

/Topics This folder contains a topic Org file for each topic; these files are where course content (e.g., lecture slides and notes, exam questions, assignment guidelines) about specific topics are stored and accessed.

3.1 Example

The following example is the directory structure of this course, Org-Coursepack, as well as the template.

```
\
|
+---Assets
|   |   setup_Macros.org
|   |
|   +---Institutions
|           JOSE.org
|           Template.org
|
+---Courses
|   +---Org-Coursepack
|   |   |   Org-Coursepack.org
```

```
|   |   |
|   |   +---2020 Spring
|   |       |   2020 Spring.org
|   |       |
|   |       +---Assignments
|   |       |   |   Assignment 1.pdf
|   |       |   |   Assignment 1.tex
|   |       |
|   |       +---Lectures
|   |       |   |   01 Introduction.pdf
|   |       |   |   01 Introduction.tex
|   |       |
|   |       +---Exams
|   |       |   |   Exam 1.pdf
|   |       |   |   Exam 1.tex
|   |       |
|   |       +---Syllabus
|   |           |   Syllabus (Section 1).pdf
|   |           |   Syllabus (Section 1).tex
|   |
|   +---Template
|       |   Course Template.org
|       |
|       +---Semester Template
|           |   Semester Template.org
|           |
|           +---Assignments
|           |   |   Assignment_1.pdf
|           |   |   Assignment_1.tex
|           |
|           +---Exams
|           +---Lectures
|           |   |   01 Introduction.pdf
|           |   |   01 Introduction.tex
|           |   |
|           |
|           +---Syllabus
|               |   Syllabus (Section 1).pdf
|               |   Syllabus (Section 1).tex
|
+---Topics
    |   Org-Teaching.org
    |   Topic Template.org
```