Getting Started with PrairieLearn Agenda – 24 Jan 2020

- PrairieLearn overview
  - “Why use it?”
- Setting up your course content creation environment
  - “What do I need to use it?”
  - Software, Course directory, Questions and Assessments
- Next workshop: Writing Good Questions
  - “How do I use it (well)?”
- Getting help

Full slides are online at https://go.illinois.edu/plWorkshopSlides

Today in 4 slides (1/4)
Important Addresses

- PrairieLearn source code (and exampleCourse folder)
  - https://github.com/PrairieLearn/PrairieLearn
  - https://github.com/PrairieLearn/PrairieLearn/tree/master/exampleCourse

- PrairieLearn User Guide content developer documentation
  - https://prairielearn.readthedocs.io/
  - Start with the “Installing and Running PrairieLearn” section
  - Work linearly down the list on the left

- Production PrairieLearn server
  - https://prairielearn.engr.illinois.edu

Today in 4 slides (2/4)
What software is needed?

- Docker Desktop
  - https://www.docker.com/products/docker-desktop
  - Virtualization platform to run PrairieLearn locally

- Plaintext editor
  - https://atom.io/ or https://code.visualstudio.com/
  - Sublime Text, Notepad++
  - Avoid rich txt format: ~~Word, TextEdit~~

- GitHub client
  - https://desktop.github.com/

- Browser

Today in 4 slides (3/4)
How to get started with content

- Load your course folder into docker PrairieLearn so you can browse and edit it.
  - Get a feel for things.
- Look at a question you want to mimic in the exampleCourse
  - See if you can copy it into your course.

Today in 4 slides (4/4)
Getting Help

- PL Office Hours
  - Spring 2020: Tuesdays 11am-1pm in 2124 Siebel Center

- Next Workshop to-be-scheduled week of Jan 27

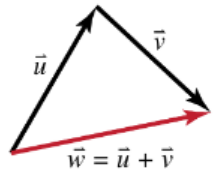
- PrairieLearn Slack chat
  - https://prairielearn.slack.com
  - If you need an invite, let Dave know. mussulma@illinois.edu
  - The #pl-help channel is great for questions.

PrairieLearn

PrairieLearn XC 101, Sp15 Assessments HW2 David E Mussulman (Student) Log out

HW2.1. Addition of vectors in Cartesian coordinates

Define the vector $\vec{w} = \vec{u} + \vec{v}$, where $\vec{u} = [10, 6]$ m/s and $\vec{v} = [8, 6]$ m/s.



What is $\vec{w}$?

$\vec{w} = [$ [ ] , [ ] $]$ m/s

Save & Grade Save only

Homework 2

Total points:	8.8/111
Score:	7%

Question

Value:	1
History:	
Awarded points:	0/5

Report an error in this question

Previous question

Next question

HW2.11. Differentiate a polynomial function of one variable

Find the derivative of

$$-7x^4 - 9x^3 + 7x^2 - 7x + 3$$

with respect to x:

$$\frac{df(x)}{dx} =$$ -7*4*x^3-9*3*x^2+14*x-7 ✓ 100%

Try question ag

Correct answer

$$\frac{df(x)}{dx} = -28x^3 - 27x^2 + 14x - 7$$

HW2.2. Fibonacci function with in-browser editor, external grading

The Fibonacci numbers are 1, 1, 2, 3, 5, 8, ..., where each number is the sum of the two before it.

$$F_n = F_{n-1} + F_{n-2}$$

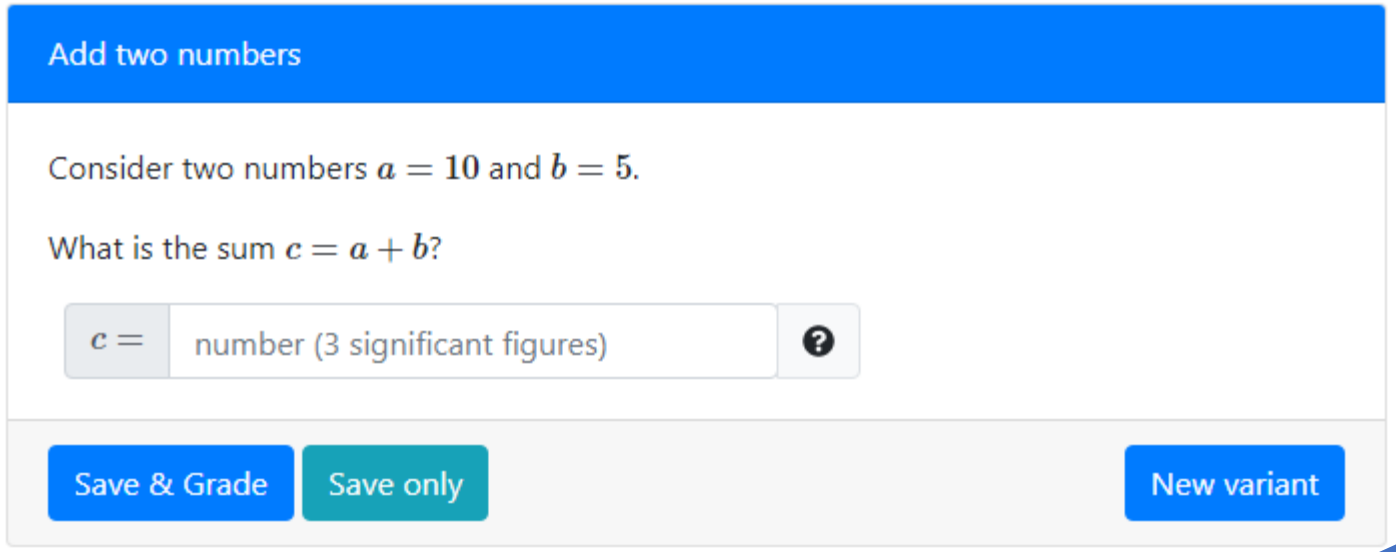
Write a Python function `fib` that takes a number `n` and returns the n^{th} Fibonacci number.

fib.py

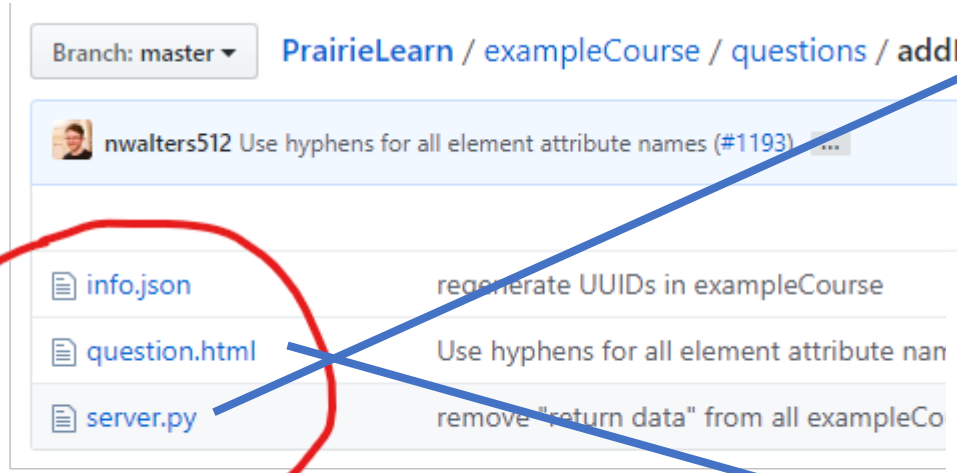
```
1 def fib(n):
2     return n+n
```

PrairieLearn questions

- Craig demos a few things as part of a talk about CBTF/PL at Berkeley.
- https://youtu.be/H6EHWmGZfpw?t=2216
  - Question demos ~ 36min – 45min in the talk




```python
import random, copy

def generate(data):

    # Sample two random integers between 5 and 10 (inclusive)
    a = random.randint(5, 10)
    b = random.randint(5, 10)

    # Put these two integers into data['params']
    data['params']['a'] = a
    data['params']['b'] = b

    # Compute the sum of these two integers
    c = a + b

    # Put the sum into data['correct_answers']
    data['correct_answers']['c'] = c
```



7 lines (5 sloc) | 252 Bytes Raw

```
<pl-question-panel>
  <p> Consider two numbers $a = {{params.a}}$ and $b = {{params.b}}$.</p>
  <p> What is the sum $c = a + b$?</p>
</pl-question-panel>

<pl-number-input answers-name="c" comparison="sigfig" digits="3" label="$c=$"></pl-number-input>
```

https://github.com/PrairieLearn/PrairieLearn/tree/master/exampleCourse/questions/addNumbers

PrairieLearn content

- Text, folders and files
- Languages used
  - HTML
  - JSON (structured text file, configuration, metadata)
  - Python (not always needed)

Questions are “question generators”, built-in randomization

- Parameterized values (1 question with variation)
- Traditional question pools (5 variations of a question, choose 2)

PrairieLearn questions

- Good for objectively answered questions
  - STEM
  - Created by a Mechanical Engineering prof

- Short answer, multiple choice
- Drawing on things (free body diagrams, finite state machines, circuit logic)
- Inline editor or file upload (coding problems)

PrairieLearn questions

- “internally graded question”
  - Answer determined and stored when question is generated
  - Python or no-code-required
  - Grading is simple comparison

- “externally graded question”
  - Question data, student submission shipped off to a Docker container
  - Do something with it (compile code, run tests, etc.)
  - Report grade result back to PL

Course content creation environment

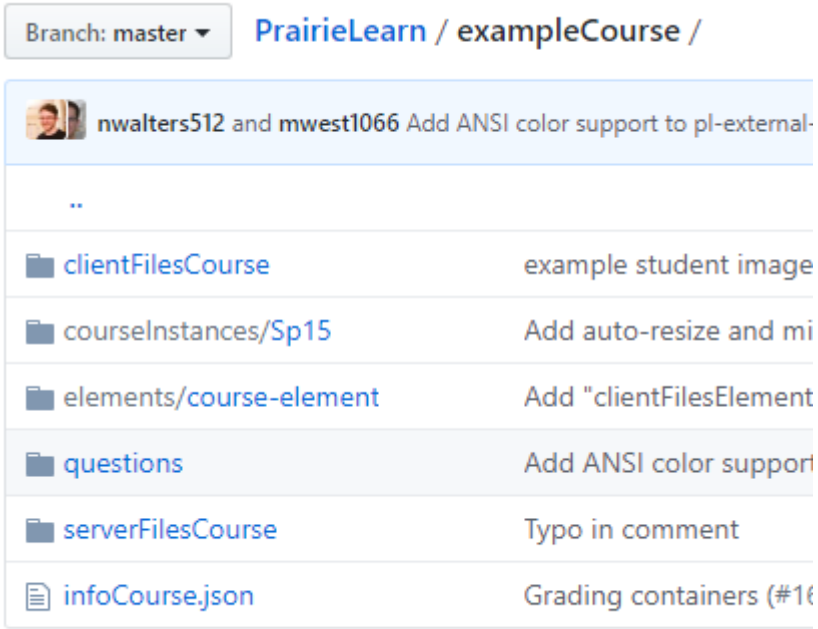
- Edit course directory, files locally with OS tools
- Use GitHub to collaborate and submit those files
- Run a development version of PrairieLearn on your laptop
  - “Sandbox” for content development and testing
  - Rapid prototyping, iterative (edit, test, edit, test)
  - Run as PrairieLearn admin, see error messages, etc.

What software is needed?

- Docker Desktop
  - https://www.docker.com/products/docker-desktop
  - Virtualization platform to run PrairieLearn locally
- Plaintext editor
  - https://atom.io/ or https://code.visualstudio.com/
  - Sublime Text, Notepad++
  - Avoid rich txt format: ~~Word, TextEdit~~
- GitHub client
  - https://desktop.github.com/
- Browser

Course folder

- Local folder on your system (can make a new one yourself)
- Eventually a private GitHub repository
  - Under the PrairieLearn github organization
- Will eventually look like:



https://github.com/PrairieLearn/PrairieLearn/tree/master/exampleCourse

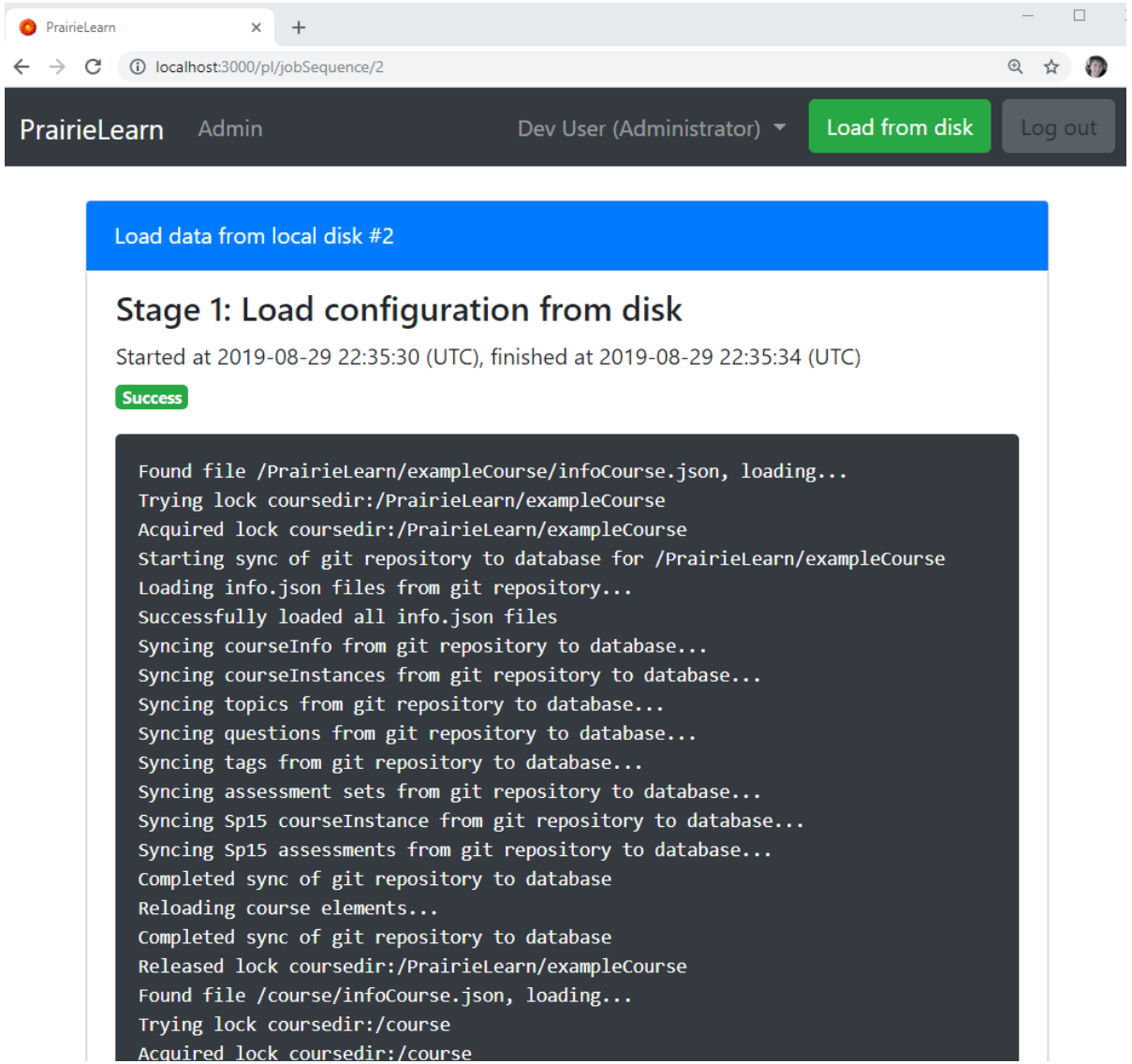
Workflow

- Make sure docker is running (whale icon in the system tray)
- Open a Terminal (MacOS) or Command Prompt (Win)
- Launch the PrairieLearn docker from the command line
  - You can see it running in that window, but you don't interact with it there.

```
PS C:\Users\mussulma>
PS C:\Users\mussulma>
PS C:\Users\mussulma> docker run -it --rm -p 3000:3000 -v C:\Users\mussulma\Documents\GitHub\pl-tam212:/course prairiele
arn/prairielearn
waiting for server to start...... done
server started

> PrairieLearn@3.2.0 start /PrairieLearn
> node server.js

info: PrairieLearn server start
info: Not using Redis for message passing
info: External grader running locally
info: PrairieLearn server ready, press Control-C to quit
info: Go to http://localhost:3000/pl
```

Interacting with PrairieLearn

- Open your browser and point it at http://localhost:3000/pl
  - That's the PrairieLearn running on your computer
- Press the green "Load from Disk" button to import course content
  - Errors? Fix them and re-load.
- Click the upper-left PrairieLearn to get back to main home screen.

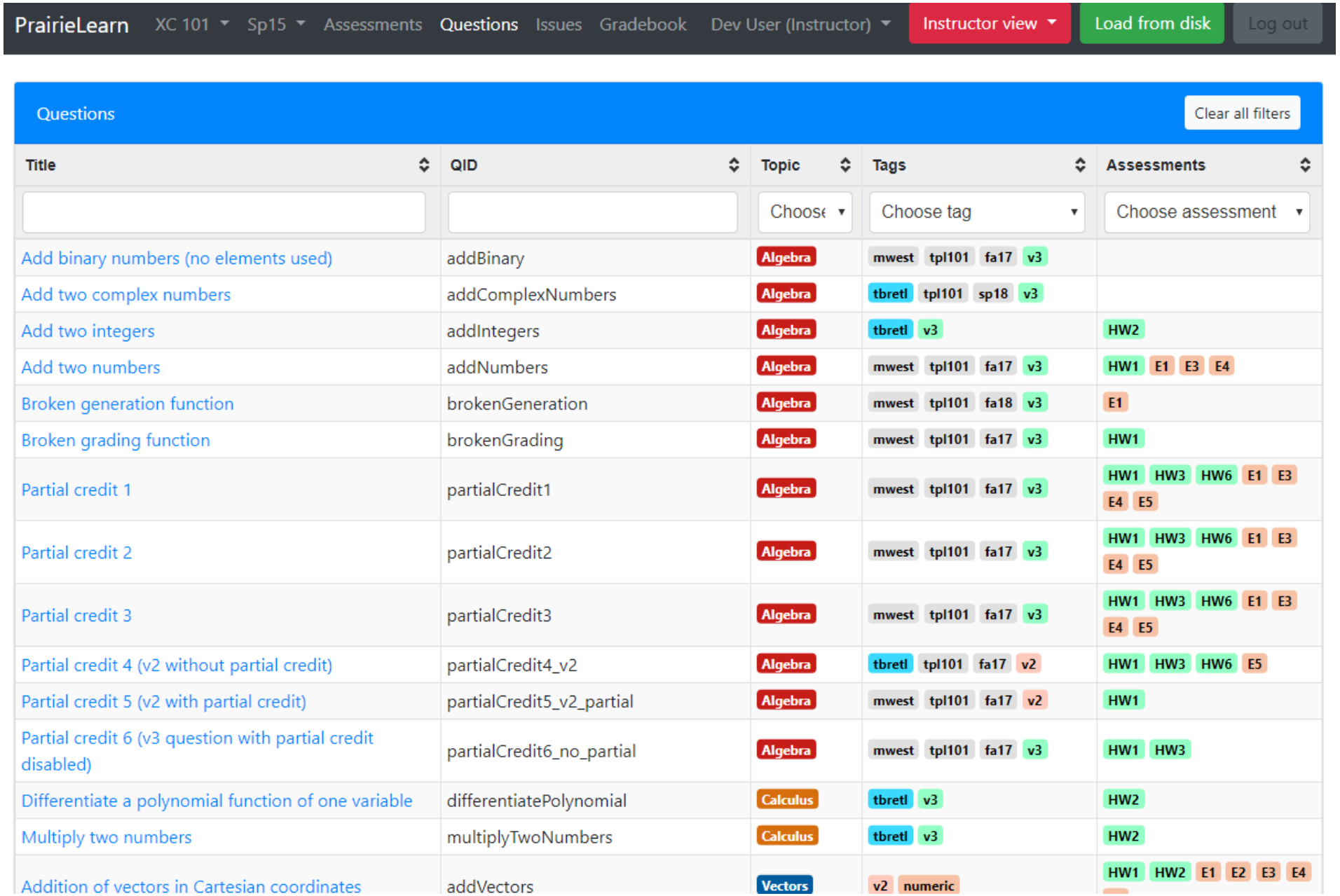


PrairieLearn Questions

This is the Questions tab in the Example Course.

You can browse questions here and try them out.

You can view the code that makes the questions in the exampleCourse folder.



Important Addresses

- PrairieLearn source code (and exampleCourse folder)
  - https://github.com/PrairieLearn/PrairieLearn
  - https://github.com/PrairieLearn/PrairieLearn/tree/master/exampleCourse
- PrairieLearn User Guide content developer documentation
  - https://prairielearn.readthedocs.io/
  - Start with the “Installing and Running PrairieLearn” section
  - Work linearly down the list on the left
- Production PrairieLearn server
  - https://prairielearn.engr.illinois.edu

How to get started with content

- Load your course folder into docker PrairieLearn so you can browse and edit it.
  - Get a feel for things.
- Look at a question you want to mimic in the exampleCourse
  - See if you can copy it into your course.

UUIDs

- The .json files for courses, courseInstances, assessments, and questions have a required “uuid” field.
  - Universally Unique Identifier

- How do you get them?
  - https://www.uuidgenerator.net/ (copy/paste)
  - Atom has a uuid generator: https://atom.io/packages/uuidgen

Page reloading and Load from Disk

- If you make a new question, courseInstance, assessment, course
  - Load from Disk
  - Tip: Any .json file edits probably need a Load from Disk to take effect

- If you are only editing an existing question
  - question.html or server.py
  - Reloading the browser page is enough to see the changes!
  - Sometimes might have to do a “new variant” – reinitialize the question

Closing thoughts

GitHub course repository

Information we need to know:

- Which course?
- GitHub usernames who should have access
  - You can sign up for a free one at https://github.com

Workflow:

- Make changes locally, commit to git, push to GitHub
- Can collaborate with others from GitHub
- Pushing to GitHub does not put the content in production

PrairieLearn in production

- Assumes admins have created the course on prairielearn.engr.Illinois.edu
- Admins also set the list of people who can sync content to the production server

- Production “Sync” functionality is like “Load from Disk”
  - Gets the latest version from your course GitHub repo
  - Is live to students at this point

- New January 2020: Edit all the content from inside PL

Getting Help

- PL Office Hours
  - Spring 2020: Tuesdays 11am-1pm in 2124 Siebel Center
- Writing good questions workshop(s) next week
  - Dave will follow up for signup times
- PrairieLearn Slack chat
  - https://prairielearn.slack.com
  - If you need an invite, let mussulma@illinois.edu know.
  - The #pl-help channel is great for questions.