1 Requirements

- Anaconda – Python package manager to setup the controlling python code and interface with Quercus through the API
- Docker - Virtual machine manager to control the running of student code within a protected and stable environment (stops the students destroying your computer or affecting other submissions)
- Quercus API key, obtained from with the user settings page on Quercus.

2 Install

1. Install anaconda and docker.
2. Get a Quercus API key and save it to `python/q.key` along with the API_URL and COURSE_ID (in the main url for the course on Quercus)

```
API_KEY=LONG_KEY
API_URL=https://q.utoronto.ca
COURSE_ID=123456
```

3. Create the conda environment to run the code. From `python/docker_scripts/` run `conda env create -f environment.yml --name phy408` . Conda will take a few minutes to complete the install and may tell you to run `conda init SHELL` to initialize your shell. Do that too.
4. Enable the environment to test it. Either `conda activate phy408` or `activate phy408`.
5. Create a Docker image with the required packages. From `python/docker_scripts/` run `docker build . -t autorun` (autorun will be the name). Docker will download the base jupyter/scipy virtual machine, and install a new environment idential to your phy408 environment called "submission" inside a docker machine. It will then install extra custom packages that you can specify in the `Dockerfile` (e.g. a special FFT package, pandas, xarray), disable the interactive plotting inside docker, and enable the new environment by default.
6. Test the Docker image. From `python/docker_scripts/docker_test` , run `docker run --rm -v` `` `pwd` ``:/home/jovyan -t autorun ./test_env.sh`

Should print out

```
# conda environments:
#
base                     /opt/conda
submission            *  /opt/conda/envs/submission
```

(the environment is installed)

Run

```
docker run --rm -v ‘pwd‘:/home/jovyan -t autorun  python ./test_plot.py
```

Should print out

```
    /opt/conda/lib/python3.7/site-packages/matplotlib/mpl-data/matplotlibrc
    Hello, World!
```

and create a test_plot.pdf with a straight line.

7. Test the Quercus interface. From the root (1 directory above `python`) Run `python/cli.py` and you should get

```
python/cli.py
Usage: cli.py [OPTIONS] COMMAND [ARGS]...

  CLI.

Options:
  --help  Show this message and exit.

Commands:
  assignment  Group
  groups      Get the group names
  users       Get the user information and store in the user cache file...
```

3 Running the code

3.1 User list

`python/cli.py users` exists mostly as a test, but running it will create a few files for you and output

```
           sortable_name
id
28703       Boone, Lyndon
112790    Buchanan, Mark
```

and it will make a `submission/data` directory with a `users.csv` file containing the (canvas) user id and name of every student. An empty `subs.csv` file will also be created to store the submission times.

3.2 Group list

Similarly, `python/cli.py groups` creates a file `submission/data/groups.csv` to store the group IDs that get linked to the submissions. These are not important if you don't use groups and the code should fall back to using user IDs.

3.3 Assignments

Most of the code is accessed through the assignment command `python/cli.py assignment` should output

```
Usage: cli.py assignment [OPTIONS] COMMAND [ARGS]...

  Group

Options:
  --help  Show this message and exit.

Commands:
  download              Download and assignment, DEPRECATED Options:...
  download-submission   Download the submissions for this assignment Args:...
  download-submissions  Download the submissions for this assignment Args:...
  find                  Finds an assignment and prints the assignment ID.
  find-submissions      Finds submissions to an assignment and prints information about eac
```

and you can access the sub-commands as
`python/cli.py assignment SUBCOMMAND ...`

3.3.1 Download single submission

`python/cli.py assignment download-submission ASSIGNMENT USER_ID` downloads the assignment for the matching user_id.

3.3.2 Download submissions

`python/cli.py assignment download-submissions ASSIGNMENT` downloads all submissions for the assignment.

`python/cli.py assignment download-submissions ASSIGNMENT --filter USER_ID` downloads all submissions for the assignment for users matching the USER_ID filter (name or number).

3.3.3 Find the Assignment

`python/cli.py assignment find ASSIGNMENT` prints the id of the assignment matching the name (mosty a sanity check)

3.3.4 List assignments

`python/cli.py assignment list` lists all assignment names.

`python/cli.py assignment list --filter=NAME` lists all assignment names matching the substring filter.

3.3.5 Find submissions without downloading

`python/cli.py assignment find-submissions ASSIGNMENT --filter=USER` lists all submissions for this assignment, the submission times, and group ids, optionally filtered by the user name or id.

4 Workflow when running the code in normal usage.

4.1 1. Download

Running `python/cli.py assignment download-submissions ASSIGNMENT` will create the following directories

- `submission/ASSIGNMENT`
- `submission/data`

The `submissions/ASSIGNMENT` contains a directory for each submission.

The `submission/data` directory contains caches used by the program to store user, group, and assignment information in CSV files, and logfiles for the Quercus interactions in `submission/data/downloaded` directory.

The `submission/data/subs.csv` file in particular lists the submissions that have been downloaded. If the submission is listed here, it won't be downloaded again, even if the directory is deleted. Remove the entry (e.g. the rows containing the lab name) from here to download again.

4.2 2. Running a submission

Running `./process_submission.bash ASSIGNMENT/USER` will initialize docker to run the students code, and will create the following directories

- `submission/logs`
- `submissions/run`

The `submission/logs` directory contains a directory for the ASSIGNMENT, and output and error files for each submission that has been run.

The `submissions/run` contains directory for the ASSIGNMENT, and a file for each assignment that has been run. If the appropriate file exists here, the submission won't be run again until it's deleted.