# Lab 0

Welcome to CPE 202! This first (zeroth?) lab will help you familiarize yourself with editing and running Python modules (programs) and using GitHub to retrieve starting code and submit finished code.

## Contents



## 1 Getting Started

For this quarter, you will be running your programs from the command line of your computer. Before starting on this lab, be sure that you have done the following:

- Figured out how to open a terminal.
- Downloaded and installed Python 3 on your computer (if not already available).
- Downloaded and installed an editor (`vim` is what I use, VS Code and Sublime are other good options).
- Created a GitHub account and told me your GitHub username via the form in my welcome email. It's also in the getting started page on Canvas.
- Made sure you have `git` installed.

Please refer to the getting started page on Canvas for more details. If you need help with any of the above, *please come talk to me*. The rest of the class will be difficult (or impossible) without accomplishing the above.

# 2 Using `git`/GitHub

## 2.1 Setting Up the Environment

Open your terminal and run the following at the prompt. You must, of course, replace my placeholders with the appropriate information.

- Set your name for `git` log entries. This will appear in log entries used to track changes to your repositories.

```
git config --global user.name "Your Name Here"
```

- Set your email address for `git` log entries. This should ideally be the same email address you used when you created your GitHub account (but it honestly doesn't matter).

```
git config --global user.email "your@email.address"
```

## 2.2 Authentication

**Important**: Even if you've used GitHub before, you may still need to do this setup.

Most operations involving GitHub require extra authentication beyond simply using a password. You have two options, both come with some level of tedious setup. You must setup up either:

- a personal access token, or
- an SSH key.

GitHub suggests the personal access token (it is more broadly useful), but the SSH key is (in my opinion) easier to setup and use. *Please ask me* if you need help with this. You will be unable to do the rest of the lab (and all future labs/projects) without doing this setup.

## 2.3 `git`/GitHub Basics

This quarter we will use GitHub to download starting code for our labs and for submitting finished code. GitHub is a code hosting platform for version control and collaboration. Using a version control program like `git` (and a hosting site like GitHub) is a smart way to keep your code safe. You can easily roll back to previous versions if you mess something up in your code, and your code is kept safely online in the event that your computer crashes, etc. Becoming fluent in the use of `git` will help you through your computer science career! Use the GitHub classroom link on Canvas to accept the starting code for Lab 0.

This will make a Lab 0 repository for you in our GitHub classroom. Once the repository has been created (you may need to refresh the page a few times), you can click the link to go to it.

At this point, the repository has been created, but now we need to get the code onto your computer so that you can work on it. In `git` parlance, we will *clone* the repository. To do so, click on the green Code drop-down button.

- If you created a personal access token above, copy the HTTPS link.
- If you created an SSH key, copy the SSH link.

Once you've copied the link, go back to the command line and be sure that you are in the directory you want to put coursework (in my case, a directory named `cpe202`). From there, type `git clone` and then paste the link that you just copied. It should look something like:

```
btjones:~/calpoly/cpe202 $ git clone <the copied link>
```

- If you created a personal access token above, you will be prompted for a username and password. The username is your GitHub username, but the password will be the personal access token. Rather than copy and paste your personal access token a lot, consider caching your credentials.
- If you created an SSH key, you may be prompted for a password. This is the password you used when creating the SSH key (not your GitHub password). If you don't want to type this password a lot, consider adding it to an ssh-agent.

This will make a copy of the code onto your computer. Now when you type `ls`, you should see a directory for your repository. If you `cd` into it, you can `ls` to see the contents of the lab.

# 3 Weight on Other Planets

In your new repository, add code to `planets.py` to query the user for their weight on Earth (in pounds), then display the user's weight on Mars and Jupiter. A sample run of the program is shown below for a user weighing 136 pounds. (Note: the user's input is shown in bold for clarity. Your program should not do bold text.)

```
What do you weigh on Earth? 136

On Mercury you would weigh 51.68 pounds.
On Jupiter you would weigh 344.08 pounds.
```

Important information and requirements:

- To calculate a person's weight on Mercury, multiply their weight on Earth by 0.38.
- To calculate a person's weight on Jupiter, multiply their weight on Earth by 2.53.
- You should display the resulting weights to two decimal places.
- Your output must match my sample output *exactly*. Make sure there is a blank line in between prompting for the user's weight and displaying the results.
- Your program may use a "cast" only one time. Ask me if you don't know what this means.
- Your program may only use the `print` function once.
- Your program must adhere to the style guidelines set forth in PEP 8. This includes a maximum line length of 79 characters.

Run the tests in `planets_tests.py` to check your solution for a few sample inputs.

# 4 Your Favorite Movie

Create a file called `movie.txt` and put the name of your favorite movie inside. The only purpose of this part of the lab is to force the use of another `git` command below.

# 5 GitHub Submission

This step will introduce the tool that you will use to submit assignments.

Now that you have completed the lab, you need to `commit` your changes and `push` your changes back to GitHub. You also created a new file `movie.txt` that needs to be added to the repository. To complete these tasks, follow these steps:

1. `git add movie.txt`

   This will add the new file to your repository.

2. `git commit -a -m "`**`Your descriptive message explaining what changed`**`"`

   This will make a record of all your changes to the repository with the message specified. In `git` parlance, this record is called a *commit* and the message is the *commit message*. The commit message should be a brief description of what you've accomplished in the commit.

   The `-a` part of the commit will add any file already in the repository that has been modified to this commit (in this case, that's just `planets.py`). We had to explicitly `add` the file `movie.txt` because it was not already in the repository.

   The `-m` part of the commit says that the commit message is coming next. These can (and almost always are) combined into `-am`.

3. `git push`

   This uploads the changes from your computer back to GitHub. Now if you check your repository on GitHub, you should see all your changes. **Your code is not successfully submitted until you've pushed to GitHub!**

You can resubmit your files as often as you'd like *prior* to the posted deadlines. Each subsequent submission will update the code pushed to GitHub. Once you've done step 1 once, you'll only need to repeat steps 2 and 3.