1.   Setting Up Python

The labs in this course will take place on Anchor days and offer you a chance to exercise the material in a more traditional setting. Early on in the semester we will cover the basics of the Git version control system as well as how to run Python scripts locally.

For the labs please download one of the following editors, or any other editor of your choice, to your machine:

- VScode

- Atom

- vim

- sublime 3

- GNU Emacs

Additionally, please ensure the latest version of Python is installed to your machine (3.9.x).
Please refer to the lab subsection on the course website's about page for all links. Once you have installed Python verify that it works by opening up a new terminal, typing in the command `python`, and hitting enter. You should be greeted by a prompt with three left facing carrots (i.e., >>>).

2.   Setting Up Git

a)   Installation and Settings

1) First we are going to download and install git locally on your machine using the following link: https://git-scm.com/downloads.
2) Once you have git installed we will use the `git config` command to setup your local profile variables. We will be setting the global variables so that for every repository you push too has your username and email included in the commit. Be sure to replace the text in between the quotes with your own username and email.

```
1   $ git config --global user.name "John Doe"
2   $ git config --global user.email "example@email.com"
```

In order to verify that these command executed successfully type the following commands into the terminal and observe the output.:

```
1   $ git config user.name
2   > John Doe
3   $ git config user.email
4   > example@email.com
```

What should be displayed is the username and email that you previously set.
Congrats! You have successfully installed and configured the git version control on your computer. Onto the next step.

3. Setting up GitHub

a) Making an Account

The first step is to to go to the GitHub website and make an account. Once you have that account setup you will be able to move onto the next sections which will allow you to setup your SSH key, clone, and use remote repositories.

b) Setting up an SSH key

Refer to the instructions at the following link for how to setup an SSH key on your respective operating system: `https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent`.

Once you have created that key, navigate to the GitHub website. Once there go to your `Settings` and then to `SSH keys`. You will be greeted by the following prompt. Click on the `New SSH Key` button.
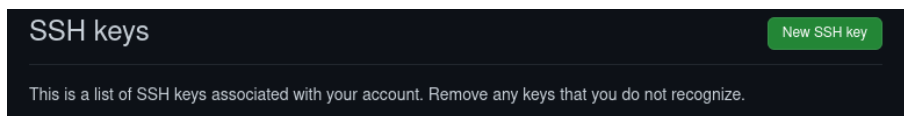

Once you've clicked on that button you will be greeted with this form. Paste the key into the `Key` section and give it a title. Once those forms are filled out hit `Add SSH key` in order to add it and you should be done!

4. Cloning a Repository

Today we will be practicing with the assignment that can be found at the following link:

```
1   https://classroom.github.com/a/hdmi4qND
```

In clicking on this link a repository will be generated on your GitHub account according to a template. In order to modify this repository on our local machine we must first **clone** the repository. To do this, navigate to the newly generated repository and (1)click on the code then (2) click on the clipboard to copy the SSH link.
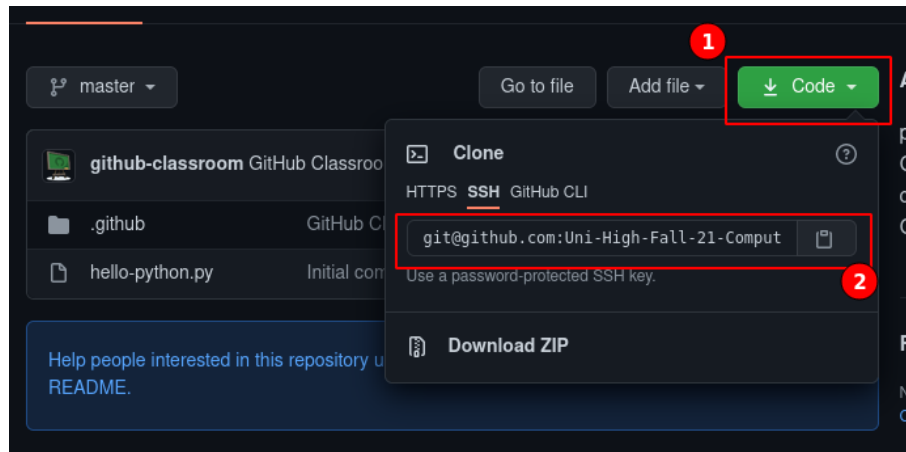
Then, using the terminal, navigate to the directory that you want to clone the repository to, and execute the following command being sure to replace it `[url]` with the URL associated with your repo.

```
1  $ cd <directory path>
2  $ git clone [url]
```

By default, this repository should have the following element:

- hello-python.py →This is the Python file that we will be modifying, adding, committing, and pushing.

5.   Modifying the Files

a)   README.md

For the final portion of the lab you will be creating the `README.md` and modifying `hello-python.py` files. Follow these instructions with regard to the README.md file:

1. Create and then open the README.md file in your text editor of choice and add a header and some information on the project. Refer to the markdown reference page for the syntax to achieve this.

2. Add this file to the index via the following command:

   ```
   1  $ git add README.md
   ```

3. Now that a snapshot of the updated file has been added to the index we now want to commit this change. In doing this we create a type of checkpoint that could be reverted to at a later date if need be:

   ```
   1  $ git commit -m "Updated README.md"
   ```

4. Now that this file has been committed locally we now want to push our changes to the remote version of our repo to (1) keep our latest changes backed up and (2) allow others to see the latest version:

   ```
   1  $ git push origin master
   ```

Now that we have updated our README.md file you can visit the repo on GitHub and view the changes.

b)   hello-python.py

Now we will apply a similar process to modify and update the hello-python file:

1. Open the `hello-python.py` file and modify it such that outputs `Hello, World!` to the terminal. This can be tested via the following command on line 1 and should produce the output displayed on the second line:

```
1  $ python hello-python.py
2  > Hello, World!
```

2. Add this file to the index.

3. Commit the file and enter an appropriate commit message.

4. Push the file to the master branch.

5. This repo is attached to an autograder on GitHub classroom. In order to determine if you have completed this stage of the assignment correctly visit the GitHub classroom website to view the autograder test cases.

6. Key Git Terms

Commands:

1. **git clone [url]:** Makes a local copy of a remote repository on your machine.

2. **git add [options] [file]:** Adds a file or a series of files and their contents to the *index*.

3. **git reset [file]:** Unstages a file or, if no file is specified, all files from the index.

4. **git status:** Shows you files that are tracked/untracked as well as those that are staged for commit.

5. **git commit -m "message":** Makes a commit to the version history with an associated message that is included in quotes after the -m tag.

6. **git pull:** Pulls changes from remote version history.

7. **git push [remote] [branch]:** Pushes the local version of your git history to a specified remote/-branch.

Terms:

1. **index:** A snapshot of the current files to be used in the next commit.

2. **unstage/staged:** When files are added to the index we refer to them as being "staged" for the next commit. When they are removed we refer to this as unstaging.

3. **tracked/untracked:** Files that are tracked are those that are being tracked for changes by git. Those files that are untracked are not currently apart of your version history and are not being tracked for changes.

7. Markdown Reference

Type	Or	… to Get
Italic	_Italic_	*Italic*
Bold	__Bold__	**Bold**
# Heading 1	Heading 1 =========	# Heading 1
## Heading 2	Heading 2 ---------	## Heading 2
[Link](http://a.com)	[Link][1] ⋮ [1]: http://b.org	Link
![Image](http://url/a.png)	![Image][1] ⋮ [1]: http://url/b.jpg	M↓
> Blockquote		┃ Blockquote
* List * List * List	- List - List - List	• List • List • List
1. One 2. Two 3. Three	1) One 2) Two 3) Three	1. One 2. Two 3. Three
Horizontal rule: ---	Horizontal rule: ***	Horizontal rule:
`Inline code` with backticks		Inline code with backticks
``` # code block print '3 backticks or' print 'indent 4 spaces' ```	····# code block ····print '3 backticks or' ····print 'indent 4 spaces'	# code block print '3 backticks or' print 'indent 4 spaces'

Link to reference page: https://commonmark.org/help/