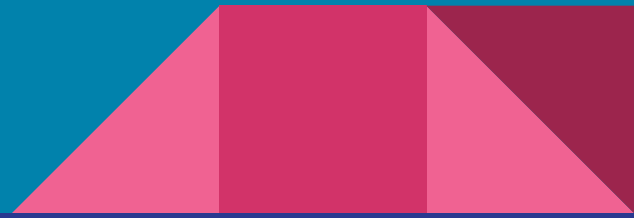
Let’s Git This Bread

Corey Fuller and Elijah Webb

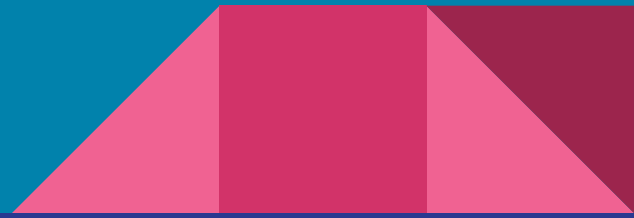
Installation/Setup

- For this Session we need to install a few tools
- Go ahead and download
  - Python 3
  - Visual Studio Code
  - Git



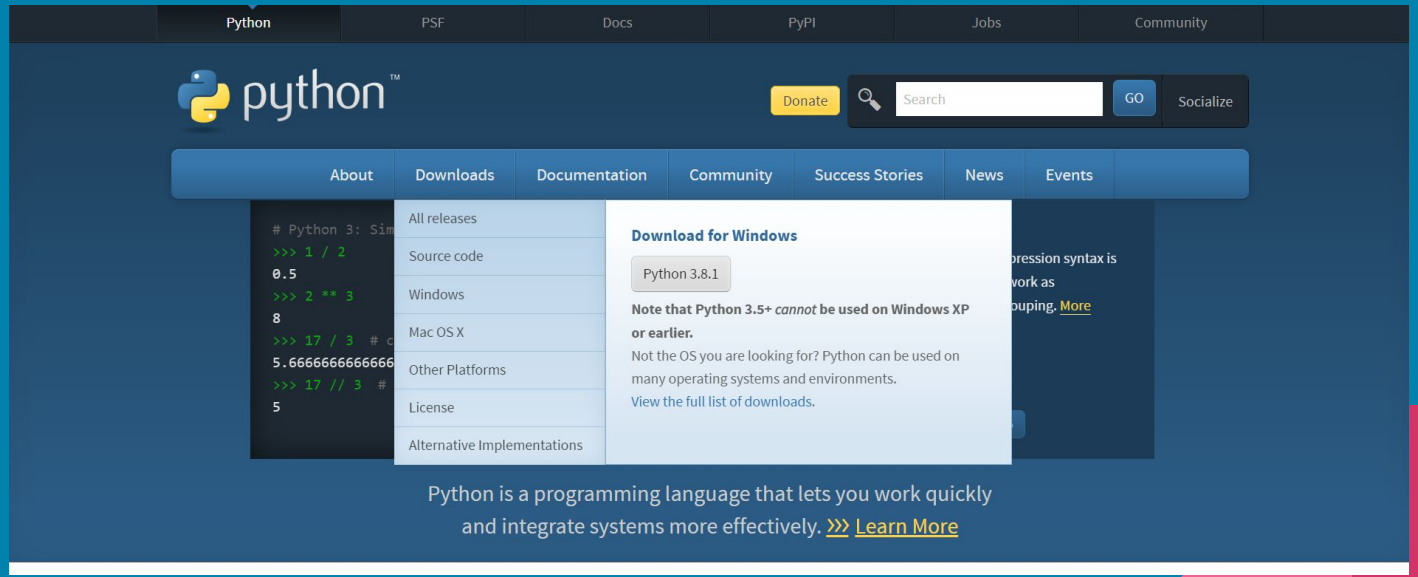
Python…?

- Python is a high level programming language
- It is also an “interpreted language”
  - a type of programming language for which most of its implementations execute instructions directly and freely, without previously compiling a program into machine-language instructions.
- Easy to use, good for introduction, widely used
  - Web development, data science. Scripting, machine learning, automation



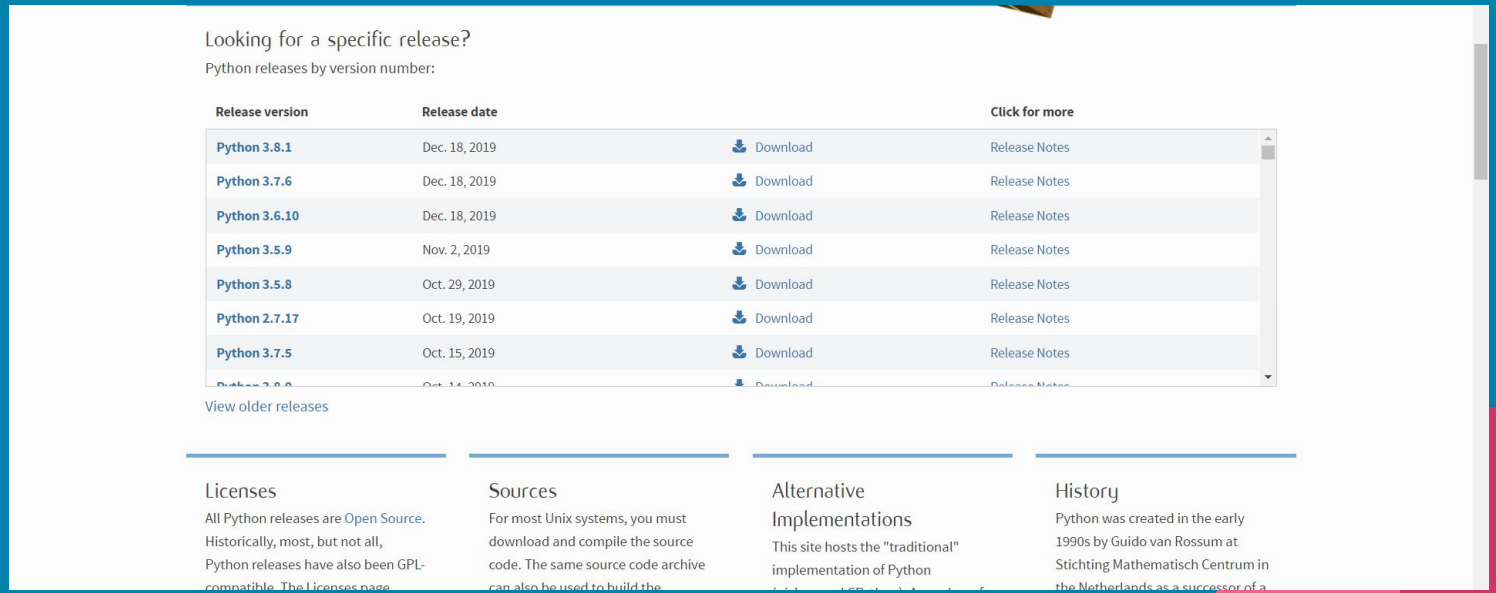
Python

- Go to python.org
- Click the Downloads menu->All releases



Python Cont.

- Scroll down until you see this menu
- Click download next to Python 3.8.1



Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.8.1	Dec. 18, 2019	Download	Release Notes
Python 3.7.6	Dec. 18, 2019	Download	Release Notes
Python 3.6.10	Dec. 18, 2019	Download	Release Notes
Python 3.5.9	Nov. 2, 2019	Download	Release Notes
Python 3.5.8	Oct. 29, 2019	Download	Release Notes
Python 2.7.17	Oct. 19, 2019	Download	Release Notes
Python 3.7.5	Oct. 15, 2019	Download	Release Notes

View older releases

Licenses

All Python releases are Open Source. Historically, most, but not all, Python releases have also been GPL-compatible. The Licenses page

Sources

For most Unix systems, you must download and compile the source code. The same source code archive can also be used to build the

Alternative Implementations

This site hosts the "traditional" implementation of Python

History

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands as a successor of a

Python cont

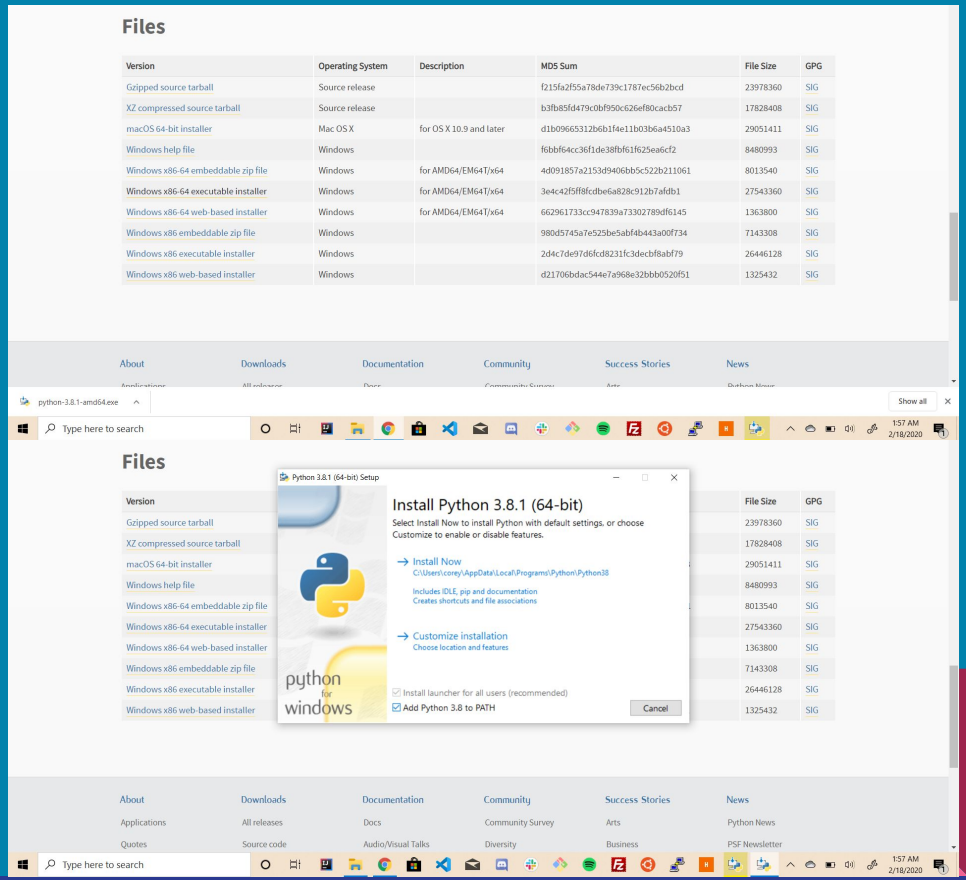
- Scroll down until you see the “Files” menu.
- Select the “executable installer” version of the file that matches your OS (macOS, Windows, Linux)

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		f215fa2f55a78de739c1787ec56b2bcd	23978360	SIG
XZ compressed source tarball	Source release		b3fb85fd479c0bf950c626ef80cacb57	17828408	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	d1b09665312b6b1f4e11b03b6a4510a3	29051411	SIG
Windows help file	Windows		f6bbf64cc36f1de38fbf61f625ea6cf2	8480993	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	4d091857a2153d9406bb5c522b211061	8013540	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	3e4c42f5ff8fcdbe6a828c912b7afdb1	27543360	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	662961733cc947839a73302789df6145	1363800	SIG
Windows x86 embeddable zip file	Windows		980d5745a7e525be5abf4b443a00f734	7143308	SIG
Windows x86 executable installer	Windows		2d4c7de97d6fcd8231fc3decbf8abf79	26446128	SIG
Windows x86 web-based installer	Windows		d21706bdac544e7a968e32bbb0520f51	1325432	SIG

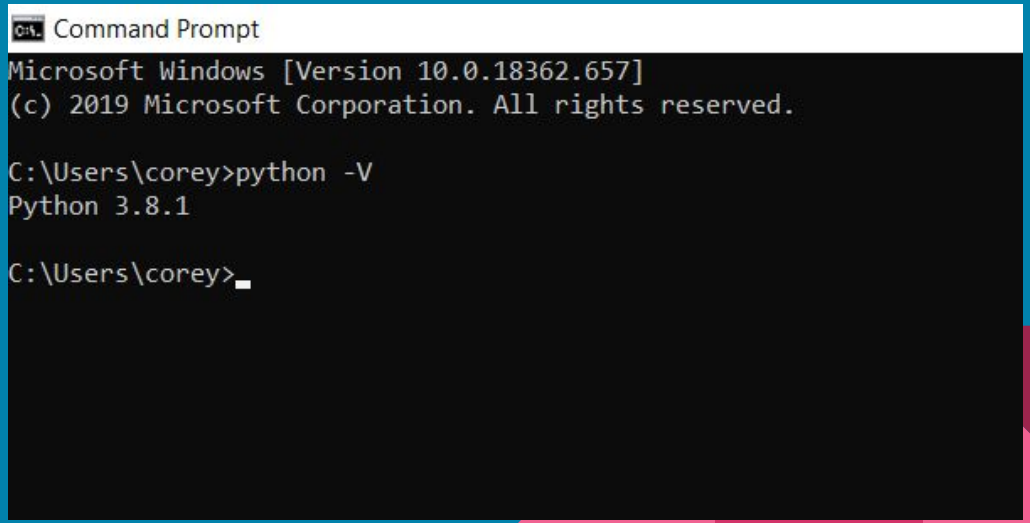
Python cont.

- Click the executable
- On the installation screen select
  - “Add Python 3.8 to PATH”
- Hit “Install Now”
- Should see a success screen



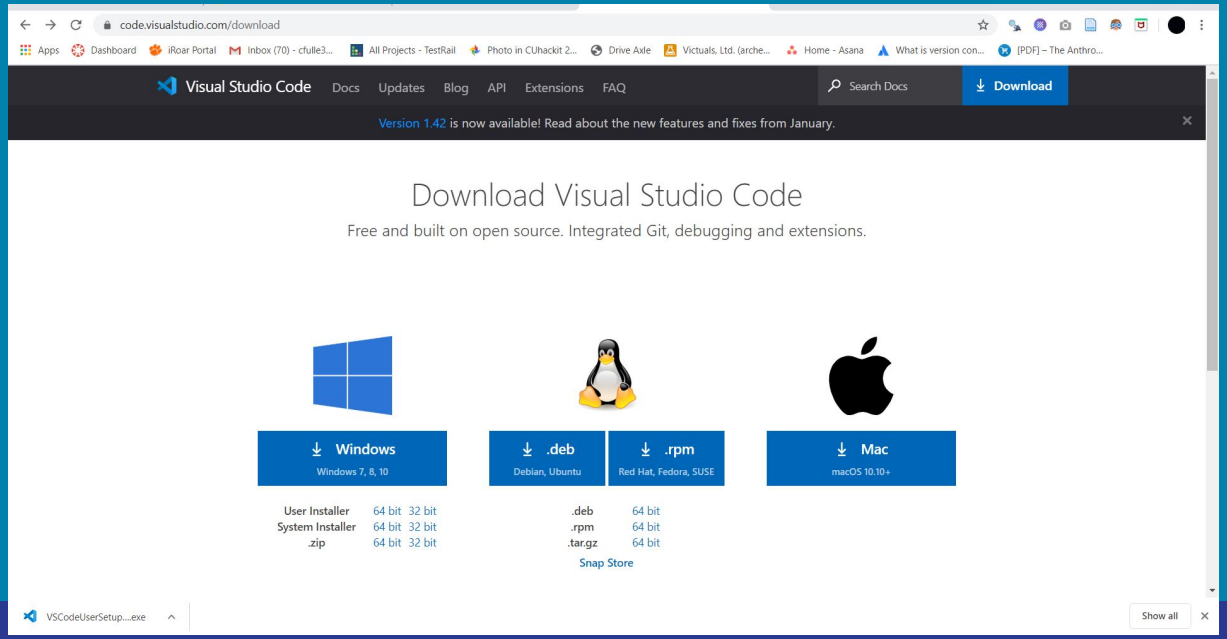
Check

- Go to your search bar and type “CMD” to open the command prompt
- Once there type the command “python -V”
- Should get a printout saying Python 3.8.1
- You’re all set with python!



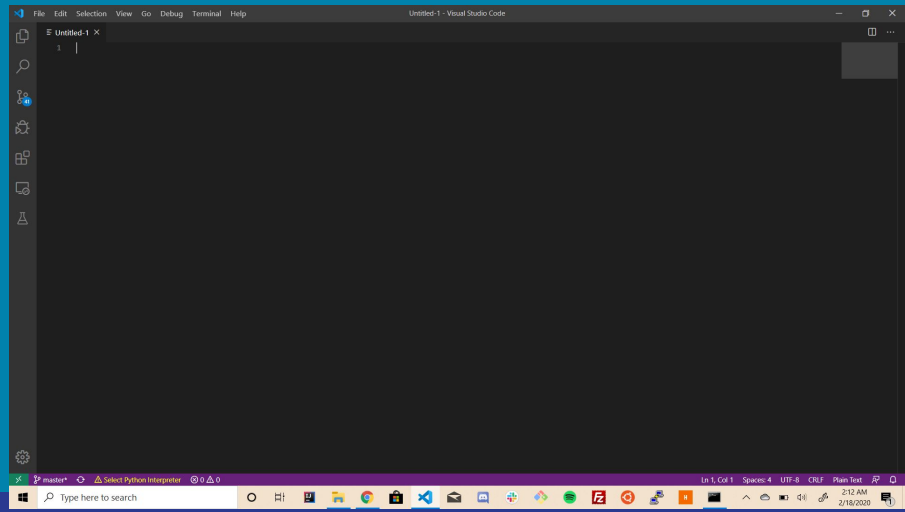
VS Code

- Go to https://code.visualstudio.com/download
- Click download next to your specific OS



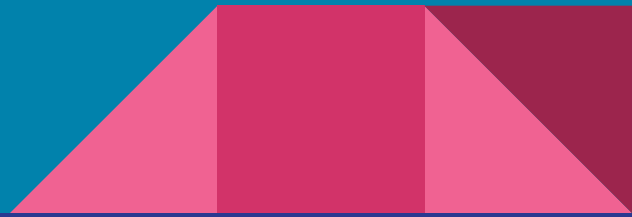
VS Code

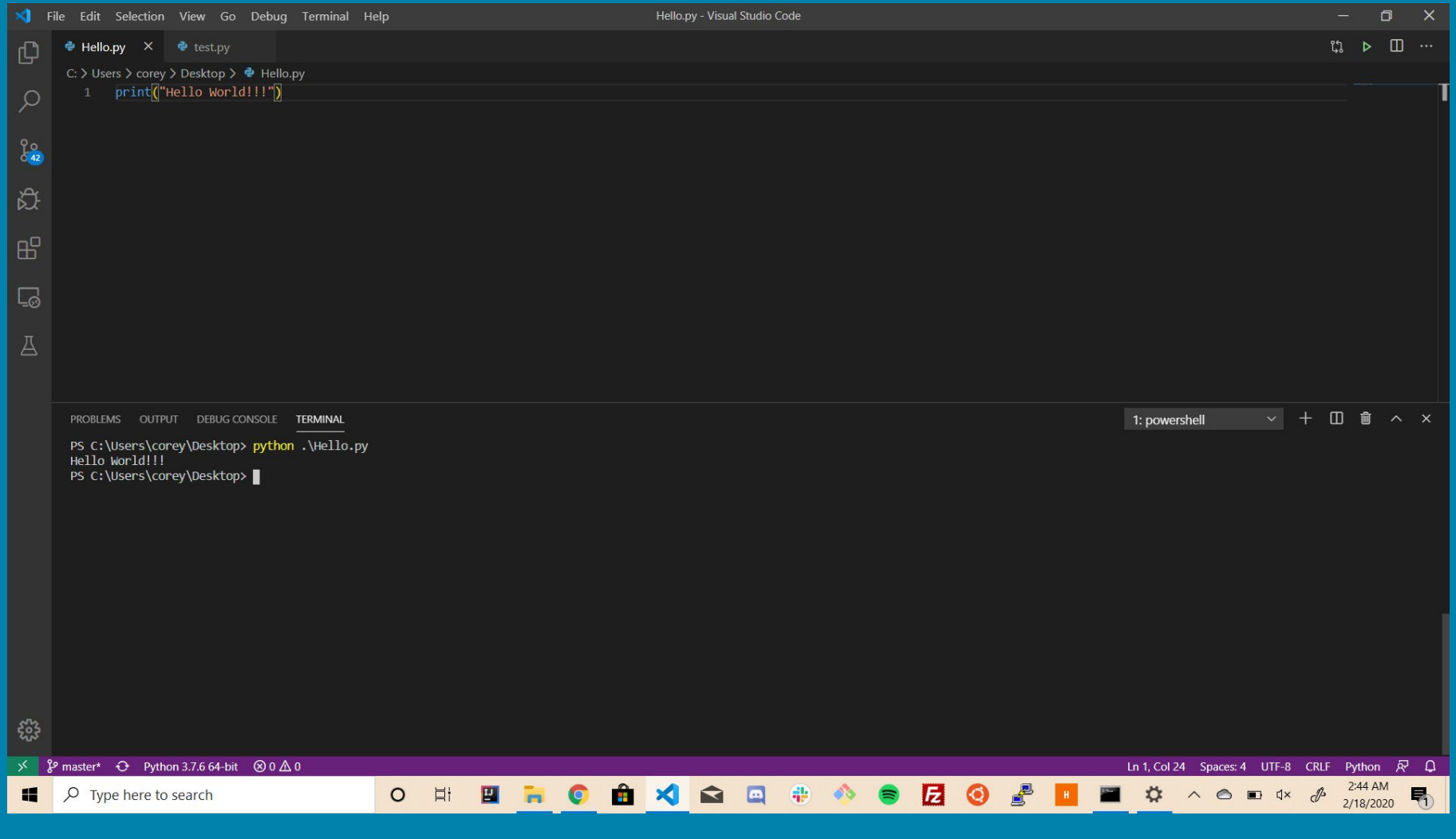
- Click on the download executable to initiate setup
- Click through all of the default settings
- Add VS Code as a desktop icon if you'd like
- Launch VS Code once install is complete
- Should be here or at a welcome page->
- If at a welcome page press ctrl + n



Python + VS Code

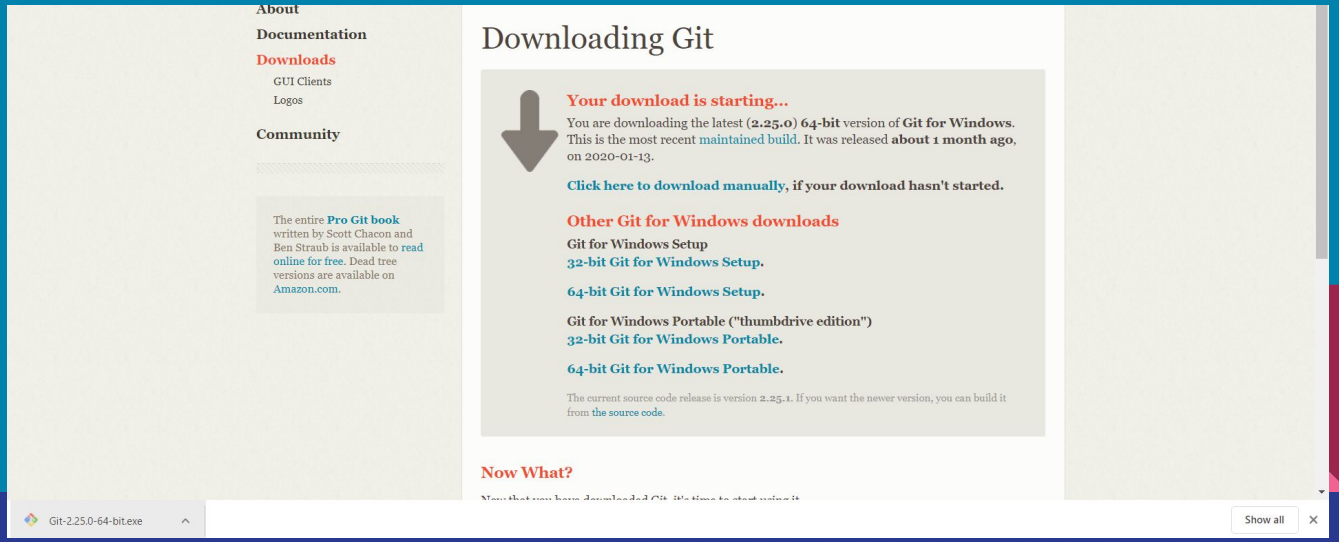
- Press ctrl + s
- Save a new file wherever you can retrieve it, Desktop recommended
- Save it as “hello.py”
- Now we can start coding
- Type print(“Hello World!!”)
- Press ctrl + ~ to open your terminal
- Navigate to where you saved the file
- Type “python hello.py”
- “Git” HYPED at the results




File Edit Selection View Go Debug Terminal Help
Hello.py - Visual Studio Code
Hello.py
test.py
C: > Users > corey > Desktop > Hello.py
1 print("Hello World!!!")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell
PS C:\Users\corey\Desktop> python .\Hello.py
Hello World!!!
PS C:\Users\corey\Desktop>
master* Python 3.7.6 64-bit 0 0
Ln 1, Col 24 Spaces: 4 UTF-8 CRLF Python
Type here to search
2:44 AM
2/18/2020

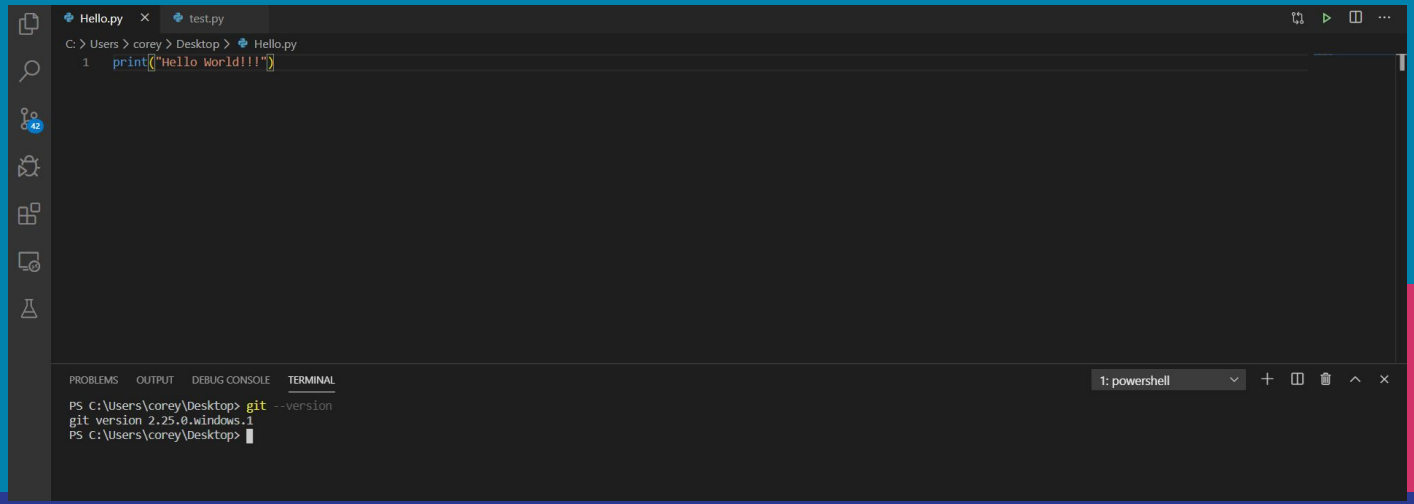
Git

- https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- Git for Windows
  - https://git-scm.com/download/win
  - Click through all the defaults



Git cont

- Go back to VS Code
- In the terminal type “git --version”
- You should see the version of git you installed
- You’re all set!!! Let’s “Git” started!!!

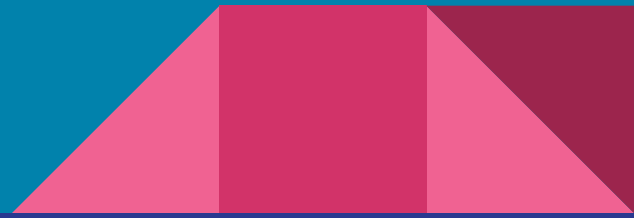


What is Git?

- Git is a version control system developed by Linus Torvalds(creator of Linux)

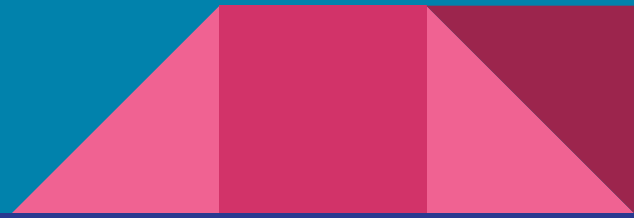
What is Version Control?

- Version Control tools are tools designed to keep track of multiple different versions of software.
- Examples of this are Git and Handin.



Why use Git?

- Git allows you to have a snapshot of a version of your code in a repository which you can revert back to at any point in time.
- Services like GitHub allow you to host this repository remotely so you(and any collaborators) can access it from any machine.



Setup

- First we need to configure some settings within Git

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git config --global user.name 'My_Name'
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git config --global user.email 'email@email.com'
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git config --global color.ui auto
```

Git Workflow

- First, we will make a directory for our project and initialize an empty Git repository in it.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents$ mkdir Python_Practice
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents$ cd Python_Practice/
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git init
Initialized empty Git repository in /mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice/.git/
```

Git Workflow

- Next, let's create a file to add to our new Git repository

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ touch hello_world.py
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ ls
hello_world.py
```

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ python hello_world.py
hello world!
```

Git Workflow

- Next, we're going to clone our empty Git repository.
- Working from a copy is very important. It allows you to make changes without consequences.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git clone /mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice
Cloning into 'Python_Practice'...
warning: You appear to have cloned an empty repository.
done.
```

Git Workflow

- Next, let's check the status of our repository

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        hello_world.py

nothing added to commit but untracked files present (use "git add" to track)
```

Git Workflow

- Next, let's tell Git to add our files to the staging area.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git add *
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   hello_world.py
```

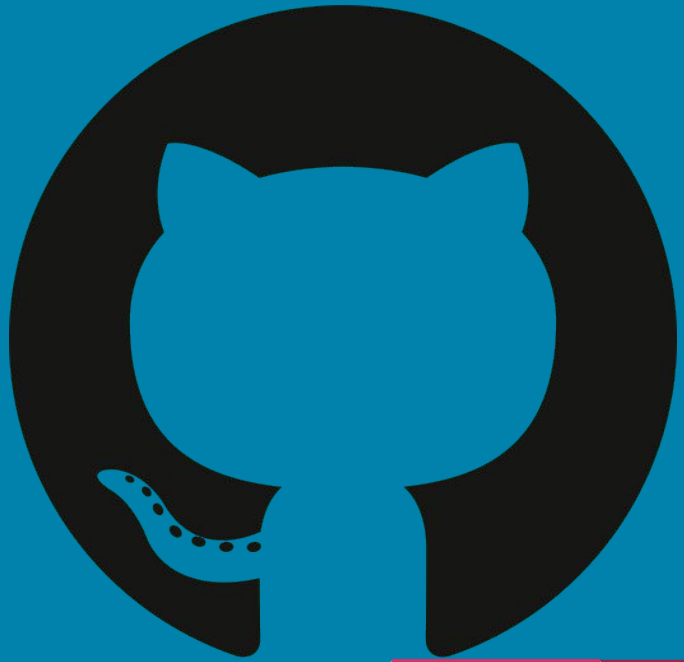
Git Workflow

- Finally, let's Git Commit our changes!

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git commit -m "Initial Commit"
[master (root-commit) 77693ac] Initial Commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello_world.py
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git status
On branch master
nothing to commit, working tree clean
```

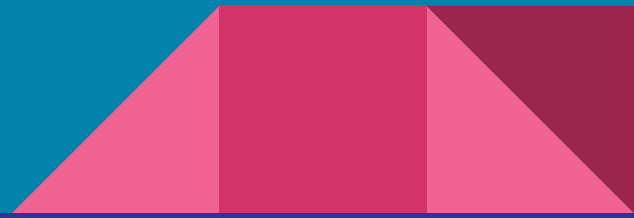
Github

- GitHub provides a place for you to host your repositories, so that you or anyone can access your code from any machine.
- Any collaborators you chose can edit the files and commit to your repository.



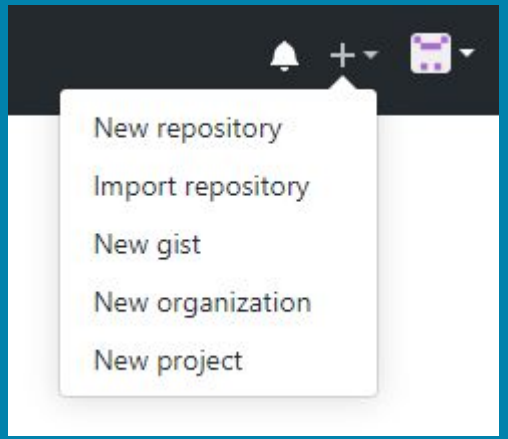
Git Remote Workflow

- First make a GitHub account.
- Choose your username wisely as your GitHub will be a large part of your resume in the future.
- Think of it like a programmer’s portfolio.



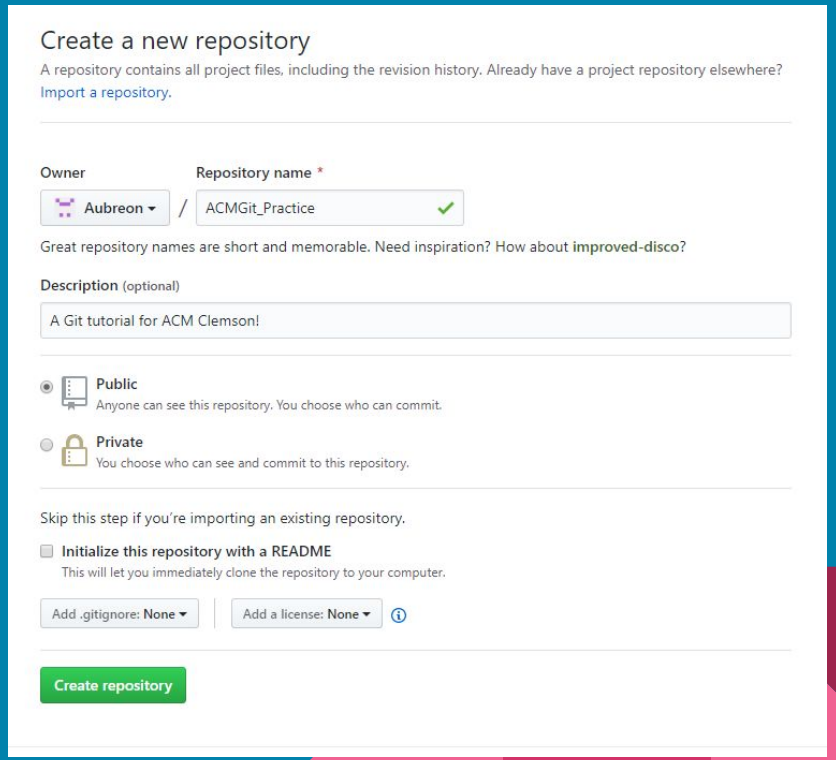
Git Remote Workflow

- Next, let's create a new remote repository.



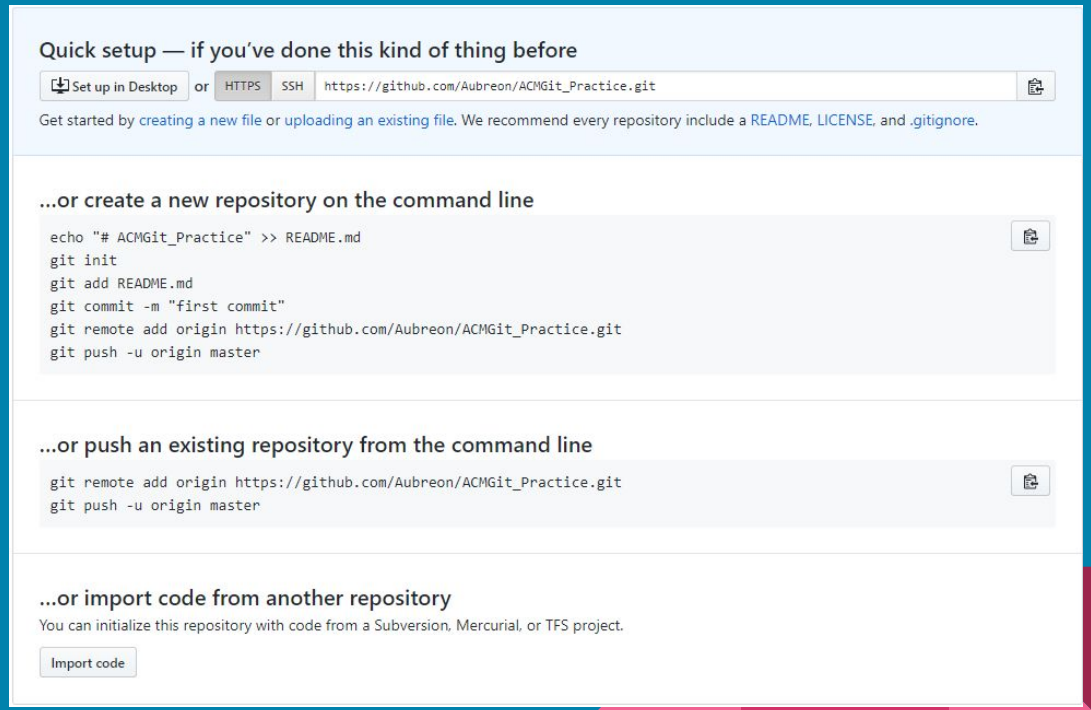
Git Remote Workflow

- Name your repository what you want the project to be called.
- Give it a description relating to the project.
- Make it Public.
- Do not initialize with a README since we are pushing an existing repository.



Git Remote Workflow

- After you create your repo, you will be taken to this page.
- Copy the link at the top.



Git Remote Workflow

- Next, we'll add our GitHub repo as the remote origin.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git remote add origin https://github.com/Aubreon/ACMGit_Practice.git
```

Git Remote Workflow

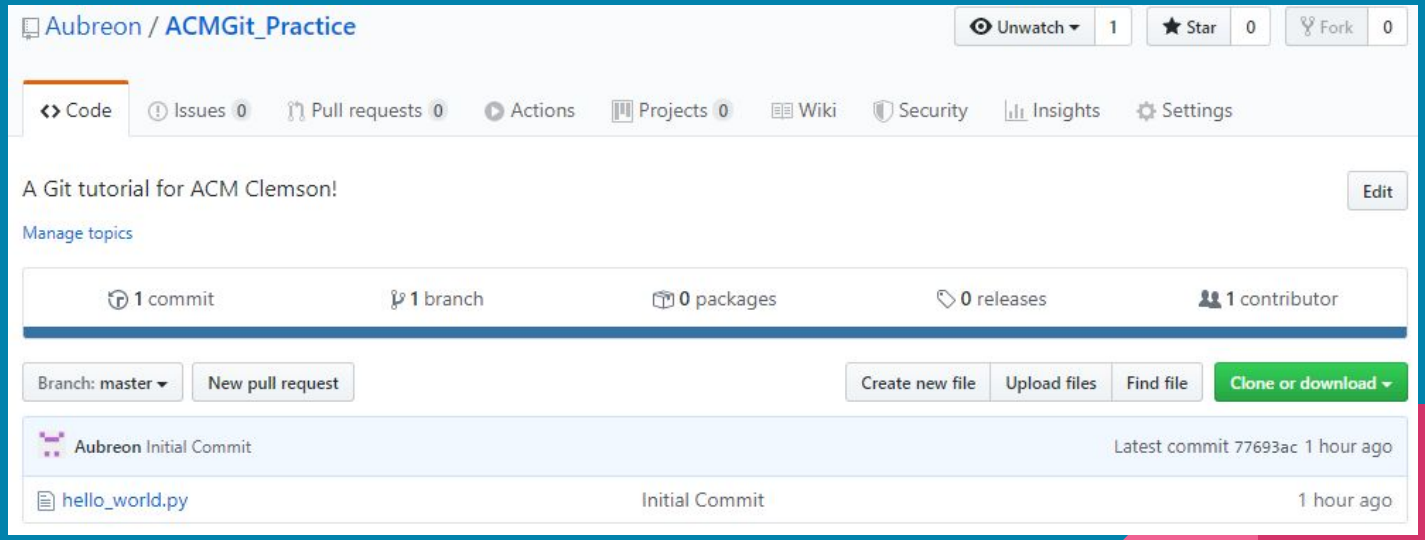
- Finally, we'll push our repo to GitHub.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git push -u origin master
Username for 'https://github.com': Aubreon
Password for 'https://Aubreon@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 238 bytes | 238.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Aubreon/ACMGit_Practice.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

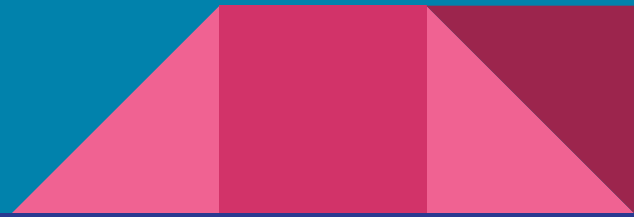
Git Remote Workflow

- If you did everything correctly, you should see your files on GitHub.



Branching

- Master branch should always be safe and useable code.
- When making changes we work on a separate branch.
- When we are sure our changes are ready and working we merge them with master.



Branch

- Git Checkout allows us to switch between branches -b makes a new branch.
- You can also create a branch using “git branch branchNameHere” , but that does not switch you to the new branch so the checkout method is safer.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git checkout -b eli
Switched to a new branch 'eli'
```

Branching

- Next we'll make a new working Python file and save it. Then we follow the same workflow as before.
- We add our file to the staging area.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git add hello_branch.py
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git status
On branch eli
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   hello_branch.py
```

Branching

- Then we commit...

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git commit -m "Added Branch"
[eli 6d2f1d0] Added Branch
 1 file changed, 1 insertion(+)
 create mode 100644 hello_branch.py
```

Branching

- Now we will move back to the master branch.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Branching

- We'll merge master and our branch.
- --no-ff retains all commit messages prior to the merge

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git merge eli --no-ff
Merge made by the 'recursive' strategy.
 hello_branch.py | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 hello_branch.py
```

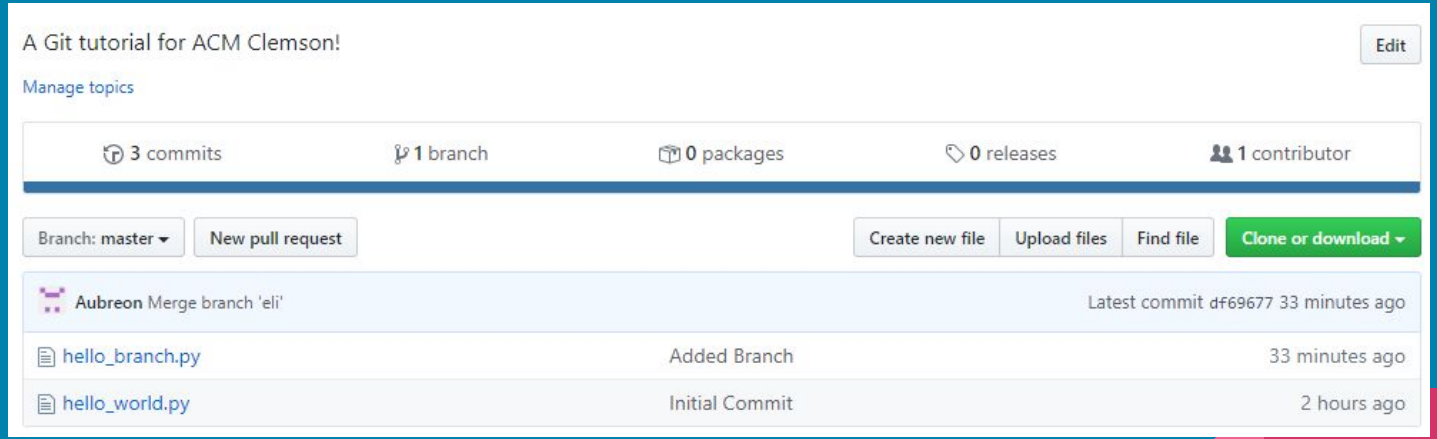
Branching

- Now that we have an updated and working master branch we will once again push to the remote repo on GitHub.

```
egwebb@MSI:/mnt/c/Users/xaubr/OneDrive/Documents/Python_Practice$ git push
Username for 'https://github.com': Aubreon
Password for 'https://Aubreon@github.com':
Counting objects: 4, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 383 bytes | 383.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Aubreon/ACMGit_Practice.git
   77693ac..df69677  master -> master
```

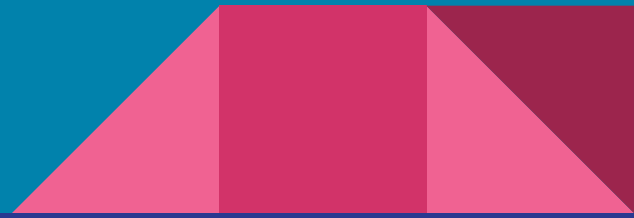
Branching

- Once again if everything was done correctly you should see your changes appear on GitHub.



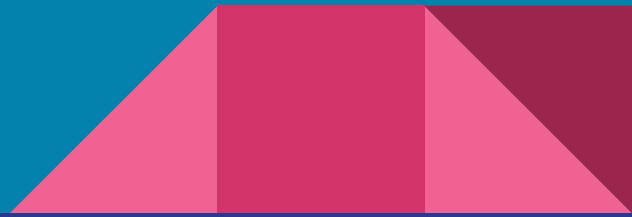
Working collaboratively with Git

- Any collaborators you add to your repo on GitHub can clone your repository to their own machine.
- Then they can make their own branch to work on, merge with master when things work, and then push to the remote repo.
- This allows a safe and easy way to collaborate with others remotely.



Helpful Reference

- https://www.hostinger.com/tutorials/basic-git-commands
- This describes basic Git commands and what they do.



Now go on, Git!