

COAPS Python Toolbelt & Tutorial

The COAPS Python Toolbelt & Tutorial is designed to get users up and running with a Python environment. Below is a list of software/IDEs that are considered industry standard. Additional integration with GitHub and additional modules are provided.

Table of Contents

Table of Contents	1
Software Utilized	2
Managing Environments	2
Integrated Development Environment (IDE)	2
Additional Tools	2
Additional Files	2
Anaconda	3
Visual Studio Code	4
Part 1: Installing Visual Studio Code	4
Part 2: Setting Up a Workspace and Terminal in VS Code.....	5
Part 3: Activating GitHub Copilot	6
PyCharm	8
Part 1: Installing PyCharm.....	8
Part 2: Installing Copilot for PyCharm	10
Jupyter Notebook	11
Python Libraries	12
Method 1: Installing Python Libraries on Mac	12
Method 2: Installing Python Libraries in VS Code	13
Method 3: Installing Python Libraries in PyCharm	13
Getting Sample Data	14
Start Coding.....	15
GitHub Integration	16
Step 1: Git Bash	16
Step 2: Create Credentials on GitHub.....	17
Step 3: Create a New GitHub Repo	19
Step 4: Quick GitHub Commands to Remember	22

Software Utilized

Managing Environments

We need some software that can manage environments and store our Python libraries. Anaconda is a powerful software bundle that allows us to create as many environments as we want and provides an additional launchpad for installing other tools! This is important when we are working on multiple projects with different libraries or wanting to share projects with other users.

- **Anaconda** - Python distribution for data science with built-in package management.

Integrated Development Environment (IDE)

We need to install an IDE to allow us to build out our scripts, projects, and eventually plots! Below are three industry standard tools that can allow us to do this. I would recommend just choosing one for now based on your comfort level. Additional details are provided below with images to help you decide!

- **Visual Studio Code** - Versatile IDE with debugging and extension support.
- **PyCharm** - Advanced IDE with code completion, debugging, testing, and project management features.
- **Jupyter Notebook** - Interactive IDE for writing and running code with visuals and notes.

Additional Tools

While it is important to understand the fundamentals of Python and the syntax it utilizes, we don't need to fly in the dark! GitHub Copilot can be natively installed via extensions in Visual Studio Code or PyCharm. This is essentially an AI coding assistant that can help the onboarding process for those just being introduced to Python. We'll also install Git Bash, which is a terminal line command tool that lets us keep our files, plots, and scripts externally for access from other users or other devices!

- **GitHub Copilot** - AI assistant that suggests and completes comments/code in real time.
- **Git Bash** - A terminal utility command-line tool that lets us keep our files externally on GitHub.

Additional Files

Some sample data files in .csv are available as well as a guide to download weather data for specific weather stations in Florida. In addition, there is a Notebook file that can be launched in an IDE or a weather_data.py file to showcase some of the amazing features that Python has to offer for categorizing and displaying data!

Anaconda

1. The first thing we need to do is download and install Anaconda. This will get all the tools we need for Python and let us set up environments

➤ **Download link** - <https://www.anaconda.com/download/success>



Important

Make sure to switch the **'Download for Mac' to Intel** with the dropdown arrow when at COAPS.

Distribution Installers



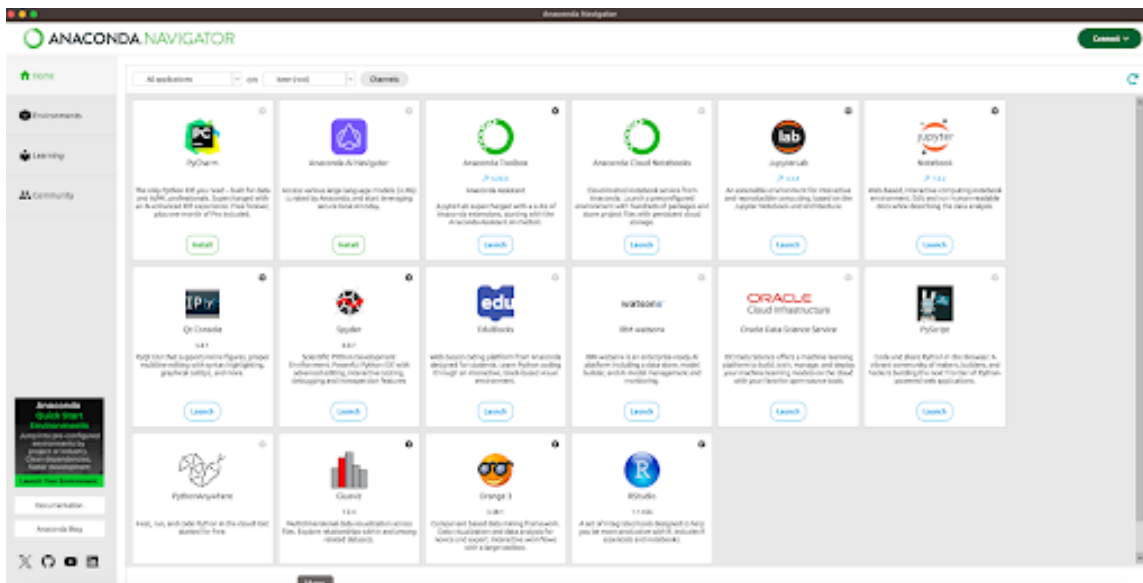
For installation assistance, refer to [troubleshooting](#).

Windows

Mac

Linux

2. Once we get that installed, we should have a screen that looks like this



3. If you click on **'Environments'** on the left, and then **'Create' on the bottom left**, you can set up a new environment (name it whatever you want).



Tip

It can take 20-30 seconds to load in but should have something like this below.

Setting up a unique environment lets us save python modules and versions.

This is important if you run a program at a later date!

Visual Studio Code

Now that we have Anaconda installed and our Python environment running, let's go get an IDE!

Part 1: Installing Visual Studio Code

1. Go to the VS Code download page: <https://code.visualstudio.com/download>
2. Under the macOS section, look for the x64 (Intel) options



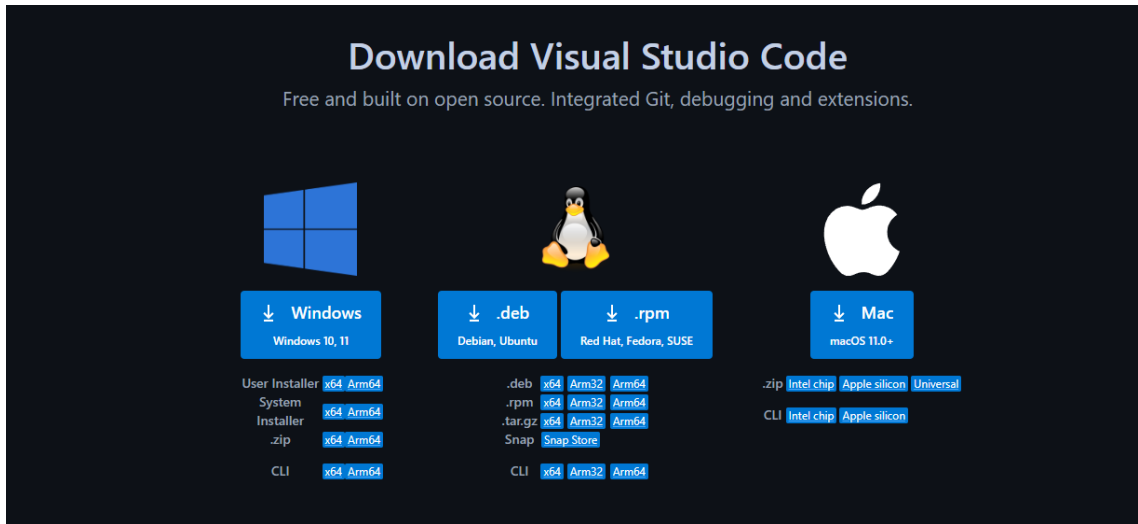
Note

You can **choose the .zip file** for a simple installation (recommended for most users) or the **User/System Installer** if you prefer a guided setup.



Important

Download the **x64 version** to ensure compatibility with your Intel Mac at COAPS.



3. Once downloaded, open your Downloads folder in Finder.
 - If you downloaded the .zip file, double-click it to extract the contents
 - This will give you the **Visual Studio Code.app** file



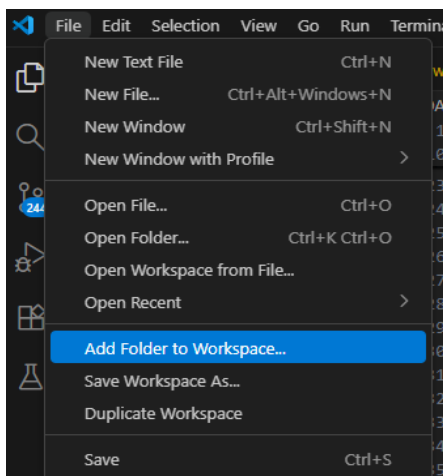
Tip

Drag the **Visual Studio Code.app** file to your Applications folder. You're good to go!

Part 2: Setting Up a Workspace and Terminal in VS Code

After launching visual studio code, there are a few things we need to do.

1. Let's setup a workspace! This is essentially adding a directory for where our files will be located. This can be done via **File → Add Folder to Workspace**

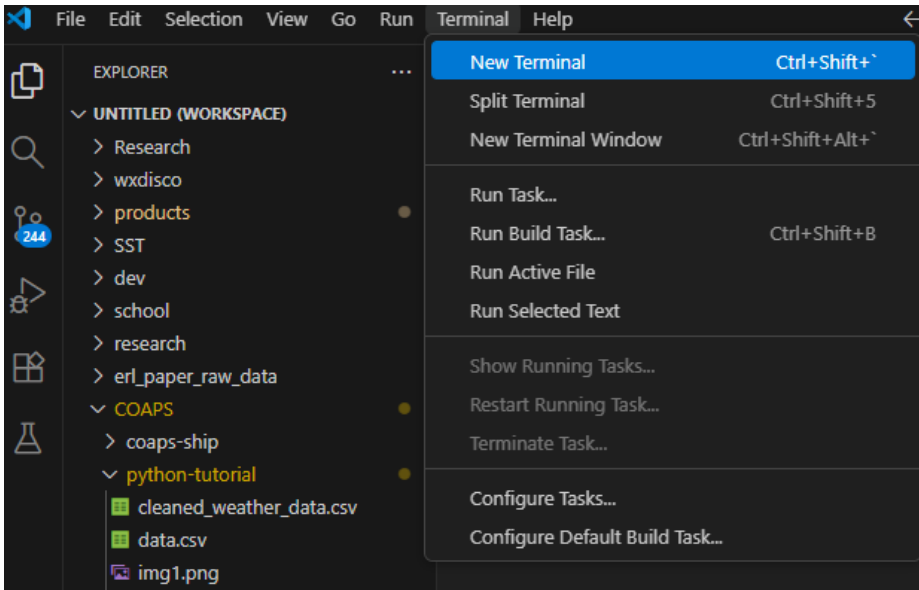


2. Let's get our terminal up and running! From the menu bar, go to **View → Terminal**, or use the shortcut **CTRL + `**

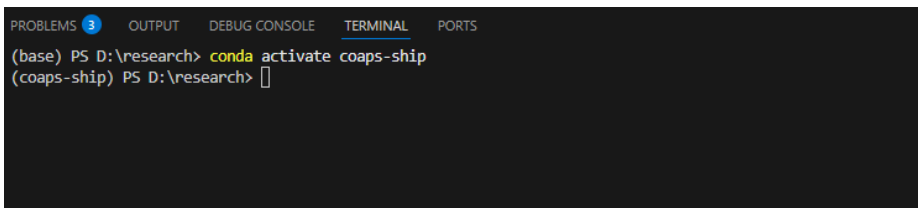


Note

In Windows, this can be done by going to **Terminal → New Terminal**.



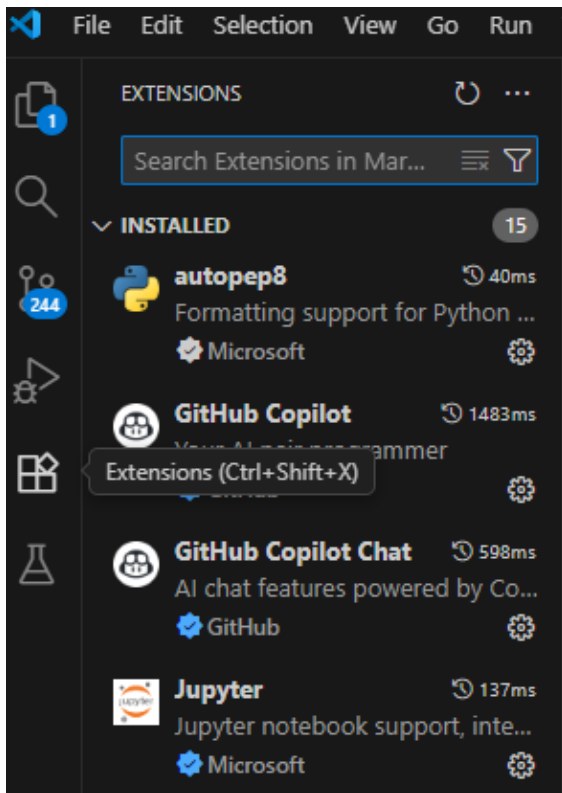
3. After our terminal has launched at the bottom of our IDE, we can activate our python environment that we created in Conda! This can be done by typing in `conda activate [ENVIRONMENT NAME]`



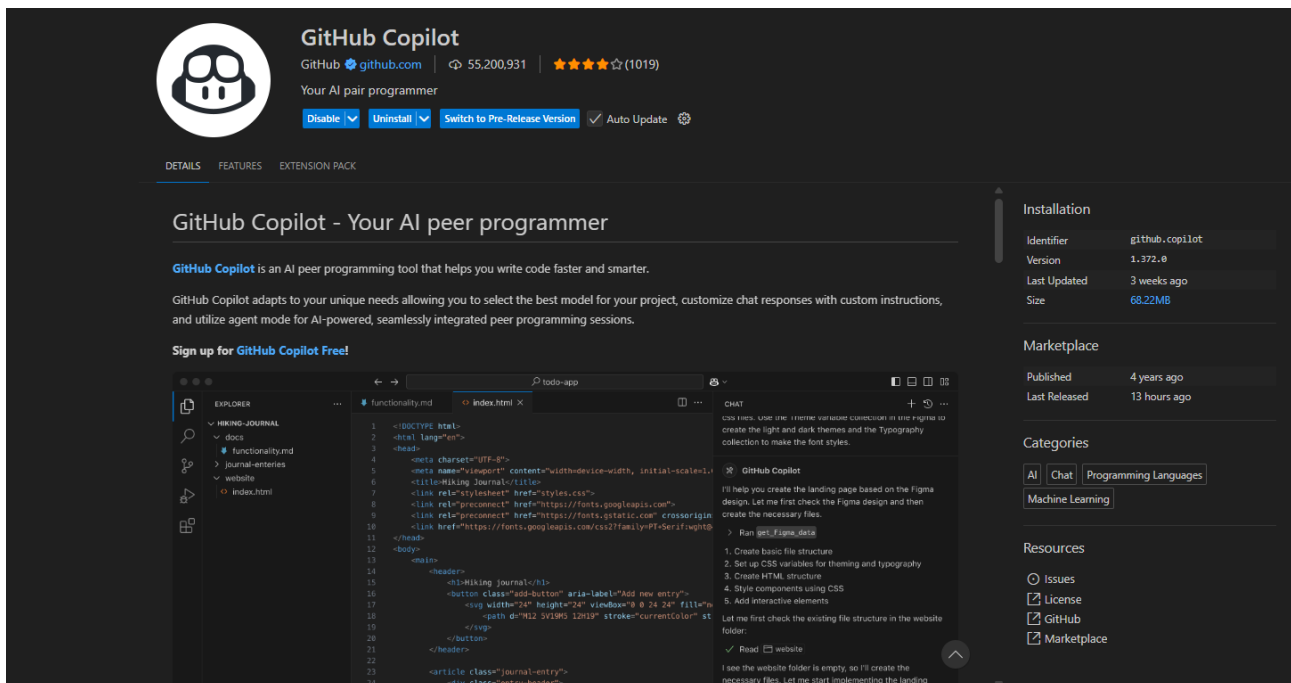
Part 3: Activating GitHub Copilot

As students (or anyone with a .edu email), we can get **GitHub Copilot for free**! This utilizes ChatGPT to provide AI assistance while coding. This is particularly useful for providing comments (or vice versa, writing a comment and having Copilot simulate the code). This can significantly reduce time required for coding or providing comments.

1. Go to the Extensions view by clicking the **Extensions icon in the Activity Bar** (square icon on the left) or using **Cmd + Shift + X**.



2. Search for **GitHub Copilot** and install the official extension from GitHub.

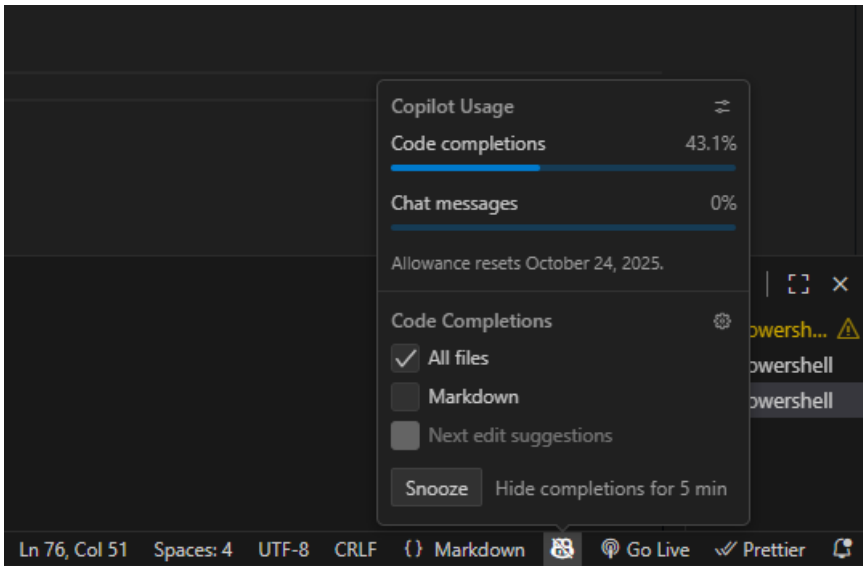


3. Sign in with your [GitHub Account](#). You'll be redirected to a browser to authorize VS Code.



Tip

You can find the **GitHub Copilot** widget at the bottom right of VS Code, near the **terminal**.



4. Once signed in, Copilot is activated!

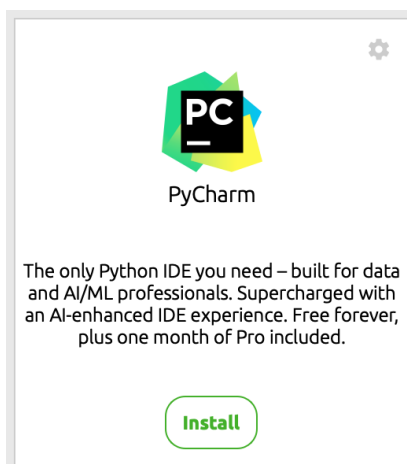
- **To test:** Open a .py file, type a comment or function signature, and Copilot will suggest code—**press Tab** to accept!

PyCharm

PyCharm provides an excellent IDE for managing large projects, as well as integrating Notebooks and scripts.

Part 1: Installing PyCharm

1. PyCharm's download location can be launched from Anaconda

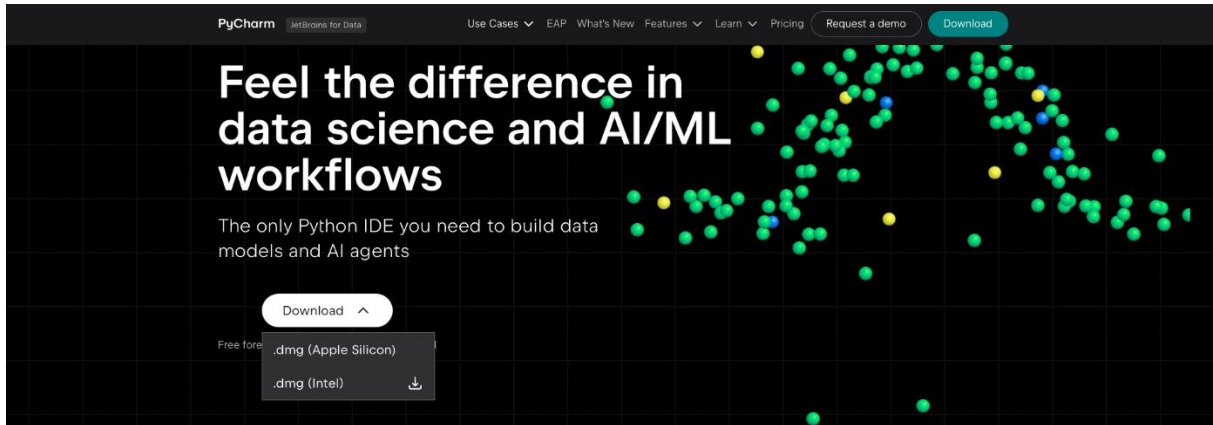




Note

Direct download link can be found here: <https://www.jetbrains.com/pycharm/data-science/?var=anaconda>

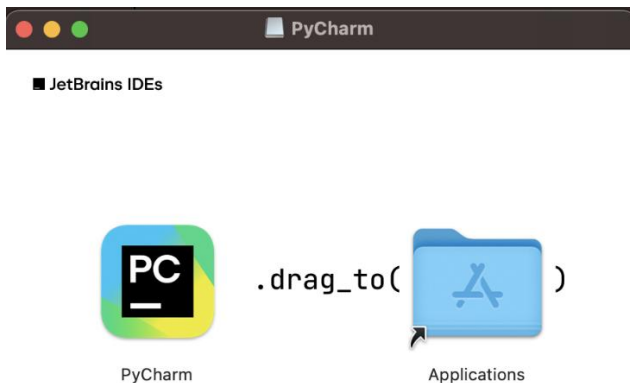
2. Click the **Download** link after the web page launches



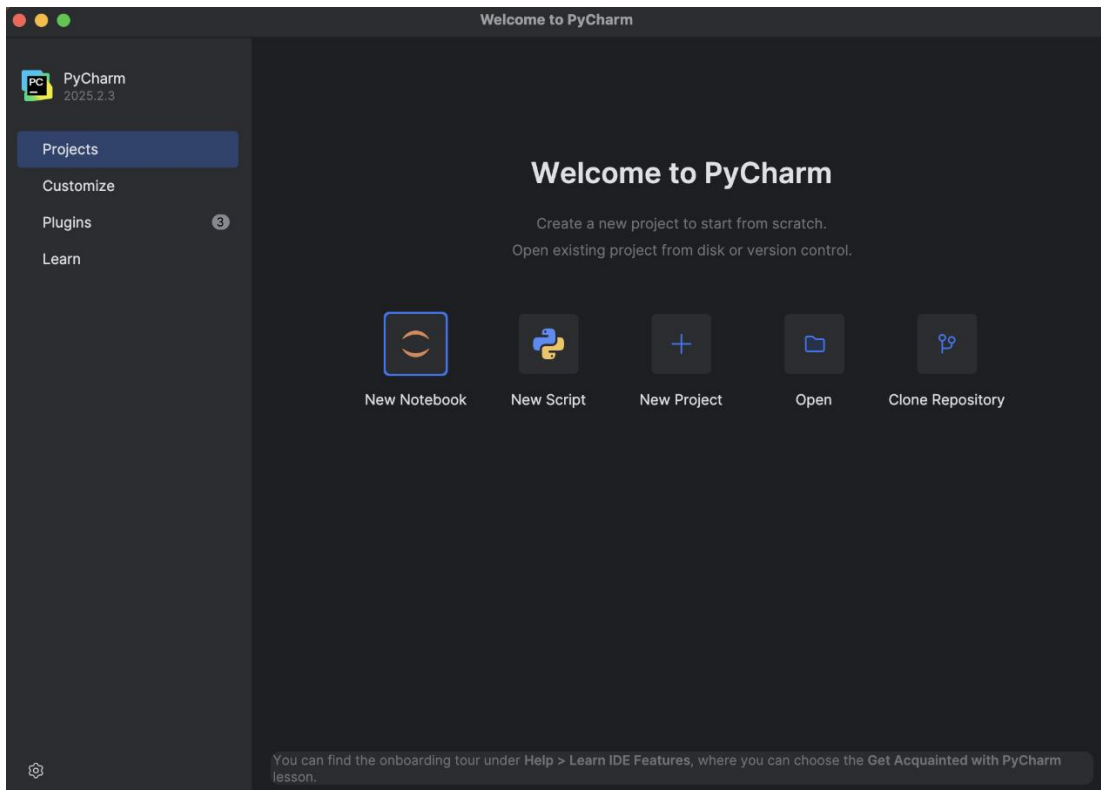
Important

At COAPS, make sure to **select .dmg (Intel)**

3. Once the .dmg file has been downloaded, launch it and drag **PyCharm** to the **Applications folder**

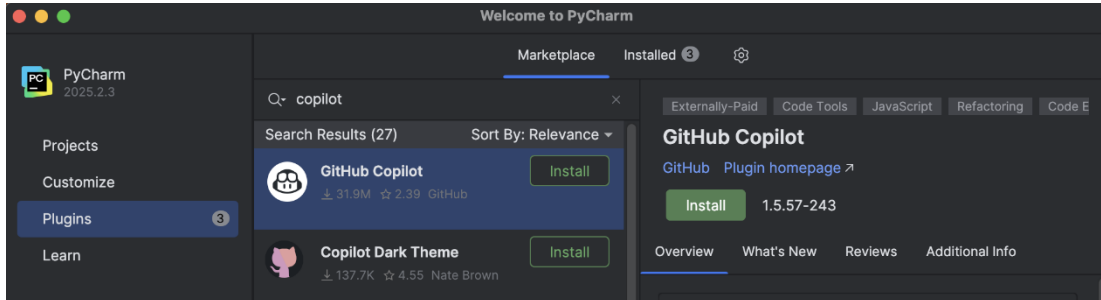


4. PyCharm is now installed and ready to go! You can **start a Notebook file** (see Jupyter below), **scripts**, or **entire projects** with PyCharm.

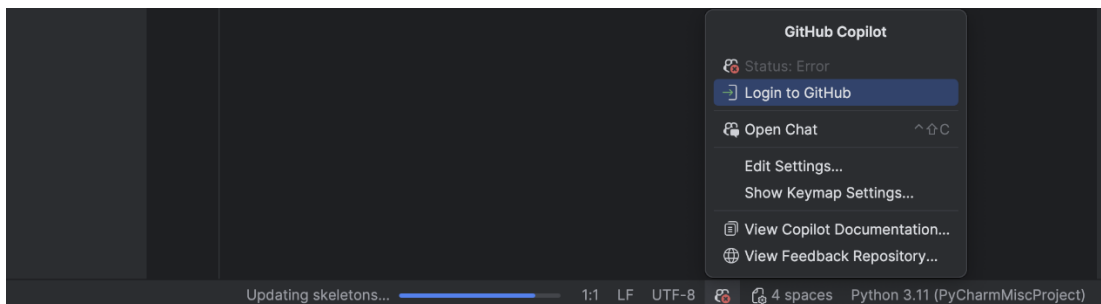


Part 2: Installing Copilot for PyCharm

1. GitHub Copilot can be installed via **Plugins**



2. After installing the plugin and restarting PyCharm, the Copilot plugin can be found on the bottom right after opening a script. Login with your GitHub account!

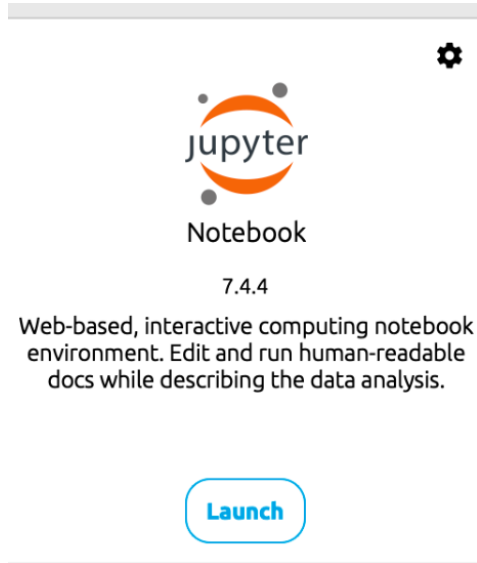


Note

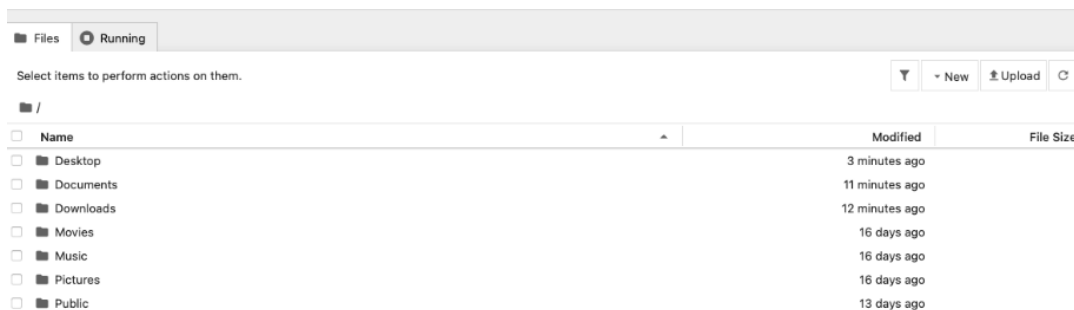
After GitHub Copilot is activated, suggestions will now appear in your script! **Hitting Tab** will accept suggestions.

Jupyter Notebook

I strongly recommend using Visual Studio Code or PyCharm, it is the industry standard IDEs for computer science and programming! Nevertheless, Jupyter Notebook provides a modular coding environment, and will be used by various professors. It has its advantages for being user friendly and to quickly test coding segments.



2. Launch it after install, and it should load something like this in your web browser

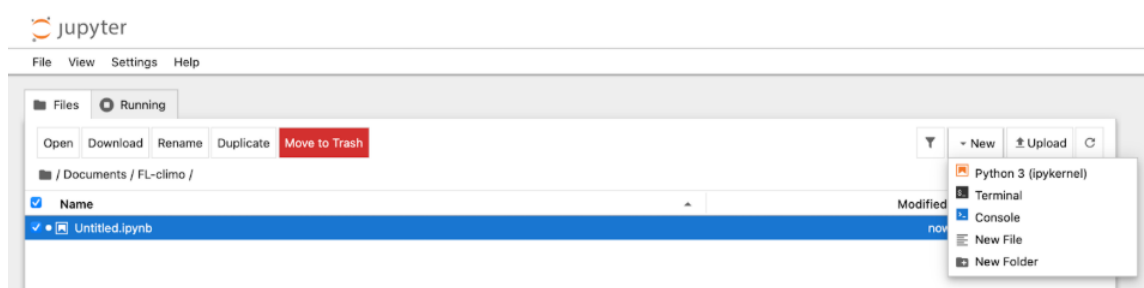


3. Create a folder anywhere you want, and inside that folder hit **'New'** and then **'Python 3 (ipykernel)'**

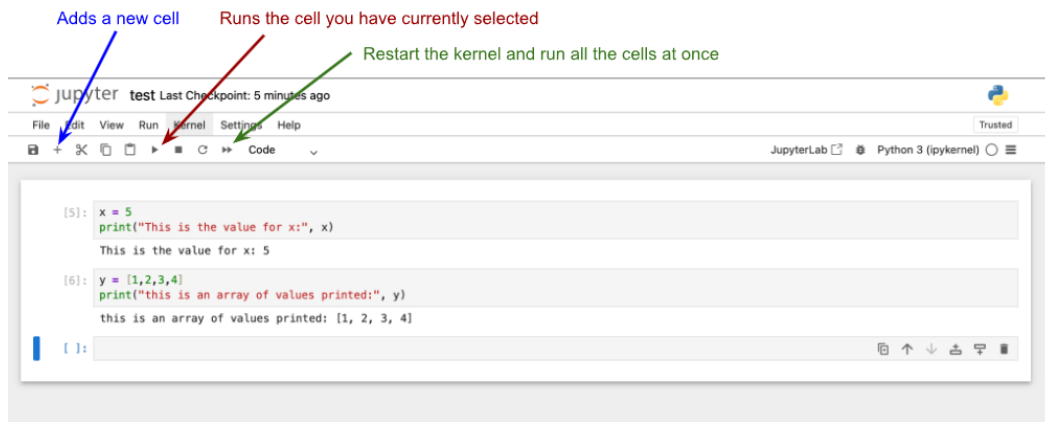


Note

This will create a new **.ipynb file** (name it whatever you want).



4. After that, you're ready to go! You can add additional cells to run different code at different times.



Python Libraries

Now that we have our Python environment and IDE set up, we need to get some Python libraries/modules. Essentially these are plugins for Python that allows us to perform various tasks without needing to code the functions/utilities ourselves!

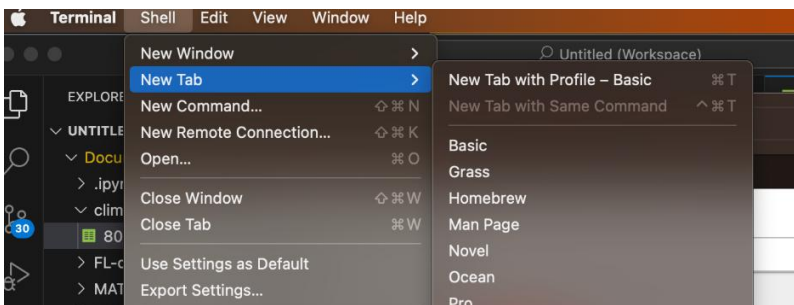
Method 1: Installing Python Libraries on Mac

1. **Open a Terminal** window in Mac from the App folder

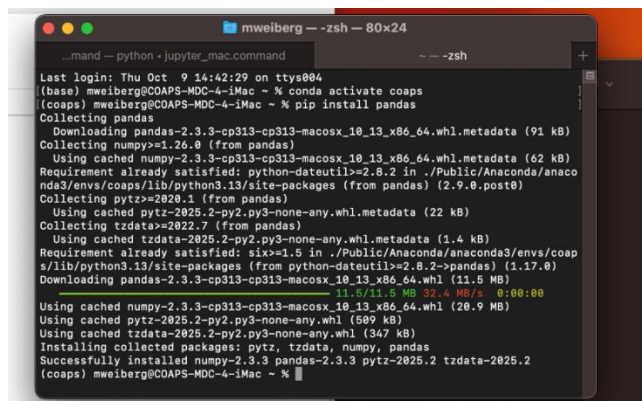


Tip

If it's already running, you can create a new tab at the top via **Shell → New Tab → New Tab with...**



2. In the **new Terminal window**, type in **conda activate coaps** (or whatever you called your environment earlier), see previous step in VS Code!
3. After our environment is active, type in **pip install pandas**. It'll run a bunch of commands like below.



```
mweiberg -- zsh -- 80x24
...mand -- python - jupyter_mac.command
-- zsh
Last login: Thu Oct 9 14:42:29 on ttys004
(base) mweiberg@COAPS-MDC-4-iMac ~ % conda activate coaps
(coaps) mweiberg@COAPS-MDC-4-iMac ~ % pip install pandas
Collecting pandas
  Downloading pandas-2.3.3-cp313-cp313-macosx_10_13_x86_64.whl.metadata (91 kB)
Collecting numpy>=1.26.0 (from pandas)
  Using cached numpy-2.3.3-cp313-cp313-macosx_10_13_x86_64.whl.metadata (62 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in ./Public/Anaconda/anaconda3/envs/coaps/lib/python3.13/site-packages (from pandas) (2.9.0.post0)
Collecting pytz>=2020.1 (from pandas)
  Using cached pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Using cached tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in ./Public/Anaconda/anaconda3/envs/coaps/lib/python3.13/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Downloading pandas-2.3.3-cp313-cp313-macosx_10_13_x86_64.whl (11.6 MB)
  11.5/11.5 MB 32.4 MB/s 0:00:00
Using cached numpy-2.3.3-cp313-cp313-macosx_10_13_x86_64.whl (20.9 MB)
Using cached pytz-2025.2-py2.py3-none-any.whl (589 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: pytz, tzdata, numpy, pandas
Successfully installed numpy-2.3.3 pandas-2.3.3 pytz-2025.2 tzdata-2025.2
(coaps) mweiberg@COAPS-MDC-4-iMac ~ %
```



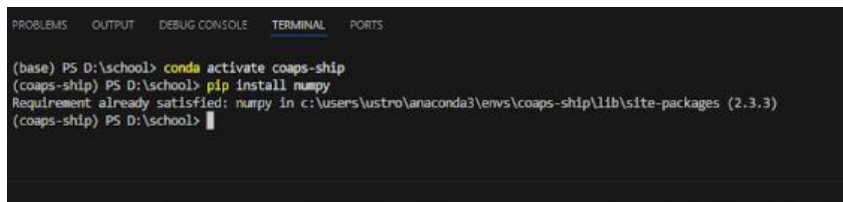
Important

While we're at it, let's **install additional Python libraries** we will need later.

- **pip install matplotlib**
- **pip install plotly**
- **pip install numpy**

Method 2: Installing Python Libraries in VS Code

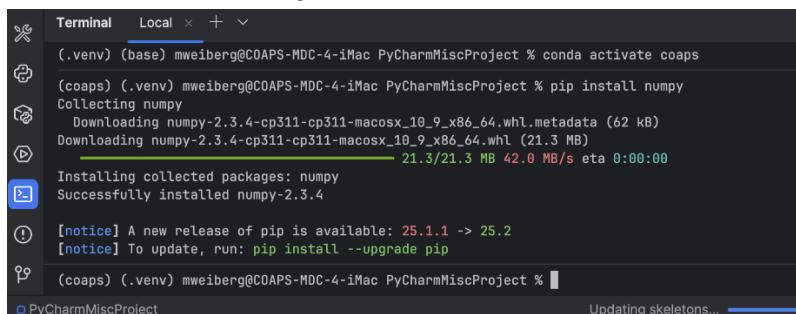
1. This can also be done in Visual Studio Code!
2. Once your conda environment has been activated, you can type in **pip install [PYTHON MODULE]**



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) PS D:\school> conda activate coaps-ship
(coaps-ship) PS D:\school> pip install numpy
Requirement already satisfied: numpy in c:\users\ustro\anaconda3\envs\coaps-ship\lib\site-packages (2.3.3)
(coaps-ship) PS D:\school>
```

Method 3: Installing Python Libraries in PyCharm

1. Much like VS Code, **PyCharm** also allows us to launch a terminal and install modules.



```
Terminal Local x + v
(.env) (base) mweiberg@COAPS-MDC-4-iMac PyCharmMiscProject % conda activate coaps
(coaps) (.env) mweiberg@COAPS-MDC-4-iMac PyCharmMiscProject % pip install numpy
Collecting numpy
  Downloading numpy-2.3.4-cp311-cp311-macosx_10_9_x86_64.whl.metadata (62 kB)
  Downloading numpy-2.3.4-cp311-cp311-macosx_10_9_x86_64.whl (21.3 MB)
  21.3/21.3 MB 42.0 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-2.3.4
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: pip install --upgrade pip
(coaps) (.env) mweiberg@COAPS-MDC-4-iMac PyCharmMiscProject %
```



Important

Make sure we activated our Anaconda environment first with **conda activate [ENVIRONMENT NAME]** and then do **pip install [PYTHON MODULE]**

Getting Sample Data

1. We can get some csv data from here for a bunch of different weather stations in Florida

<https://climatecenter.fsu.edu/climate-data-access-tools/downloadable-data>

Downloadable Data

Please select station: APALACHICOLA AP

Starting Date: January 1 1931

Ending Date: January 1 2024

Please select a variable: All

Download File



Tip

After navigating to the link above, make sure to **select 'All'** for variable and a **year range** for the station you select.

2. This will download a .csv file, which has a unique name (you can change this name to whatever convention you want!)



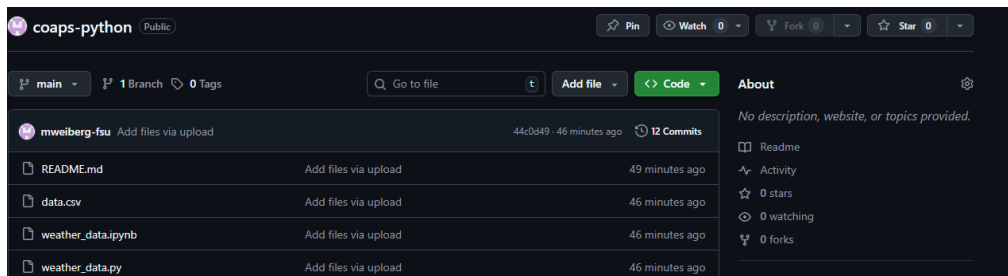
Important

Move this data file to your main directory (where your python file or .ipynb exists) to access it.

Start Coding

We're all set up and can now start coding and plotting data!

1. [Attached in this GitHub Repo](#) are some sample files and data to start with.
 - **data.csv** is the data file we downloaded earlier
 - **weather_data.py** is a sample Python file with comments that will create plots
 - **weather_data.ipynb** is a sample Jupyter file with comments that will create plots



Tip

After running **weather_data.py** in VS Code, we should get output in the terminal from our prints.

```
Temperature Statistics:
Highest Max Temp: 103.0°F
Lowest Max Temp: 30.0°F
Highest Min Temp: 86.0°F
Lowest Min Temp: 9.0°F

Final cleaned data shape: (32945, 8)

First 5 rows of final cleaned data:
      COOPID  YEAR  MONTH  DAY  PRECIPITATION  MAX TEMP  MIN TEMP  MEAN TEMP
DATE
1931-03-01   80211  1931     3     1           0.00      73.0     57.0     65.00000
1931-03-02   80211  1931     3     2           0.57      59.0     51.0     55.00000
1931-03-03   80211  1931     3     3           0.03      55.0     45.0     50.00000
1931-03-04   80211  1931     3     4           0.00      53.0     39.0     46.00000
1931-03-05   80211  1931     3     5           0.00      57.0     40.0     48.50000

Temperature plot saved as 'temperatures_over_time.png'
Precipitation plot saved as 'precipitation_over_time.png'
Mean temperature plot saved as 'mean_temperature_over_time.png'

=====
DATA PROCESSING COMPLETE
=====
Total records processed: 32,945
Date range: 1931-03-01 to 2024-01-01
```

GitHub Integration

Since we already created an account on GitHub for Copilot, we might as well utilize it! GitHub allows us to **create repositories**, **upload** (called **push** in GitHub) our files/plots, and **download** (called **pull** in GitHub). This way, we can share files among multiple devices, keep version history, and even share our GitHub repos with other people!

Step 1: Git Bash

We don't want to manually upload files, so let's get some additional software that lets us do this from a terminal!

1. Navigate to <https://git-scm.com/downloads> and click MacOS at COAPS



2. There are a few ways we can install Git, the easiest is to try this command in a terminal after activating our environment `brew install git`



Tip

Some Macs may prompt you to download git after doing `git --version` in a terminal if it's not installed.



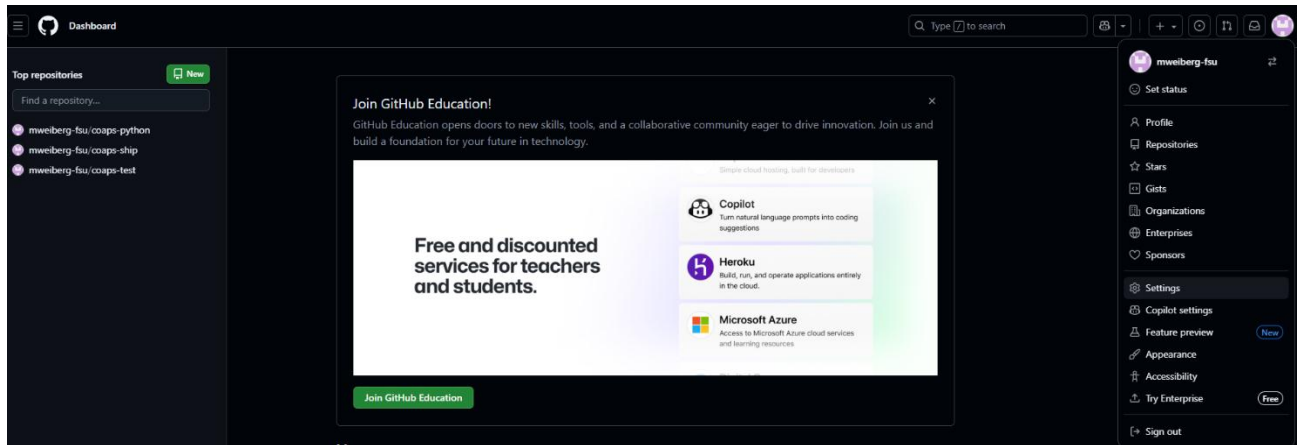
Important

If **Homebrew is not installed**, run this command in the terminal `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`

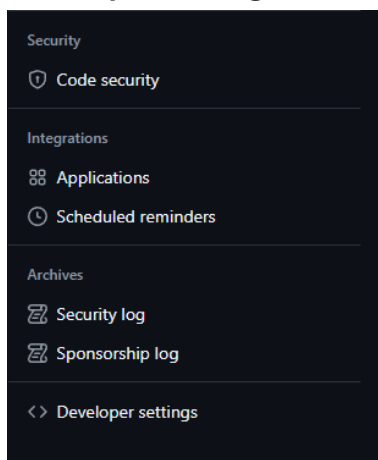
Step 2: Create Credentials on GitHub

We need to do some setting up for our new GitHub account so we can properly have credentials!

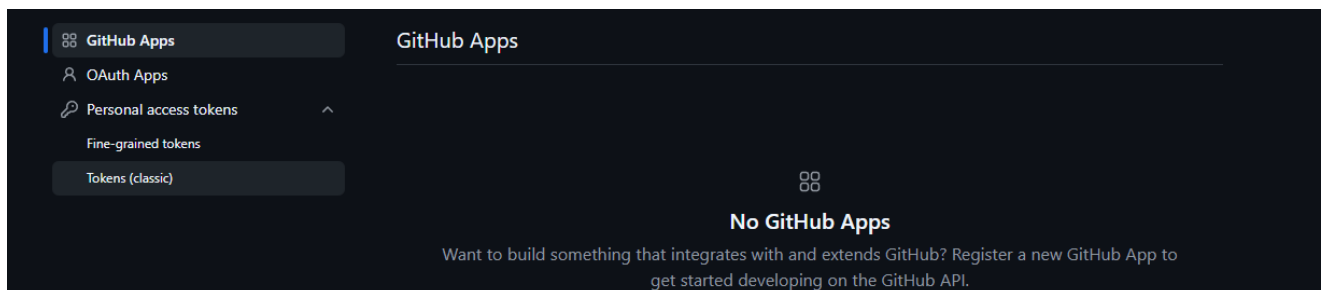
1. After logging into GitHub, **navigate to the top right image** (your profile image) and go to **Settings**



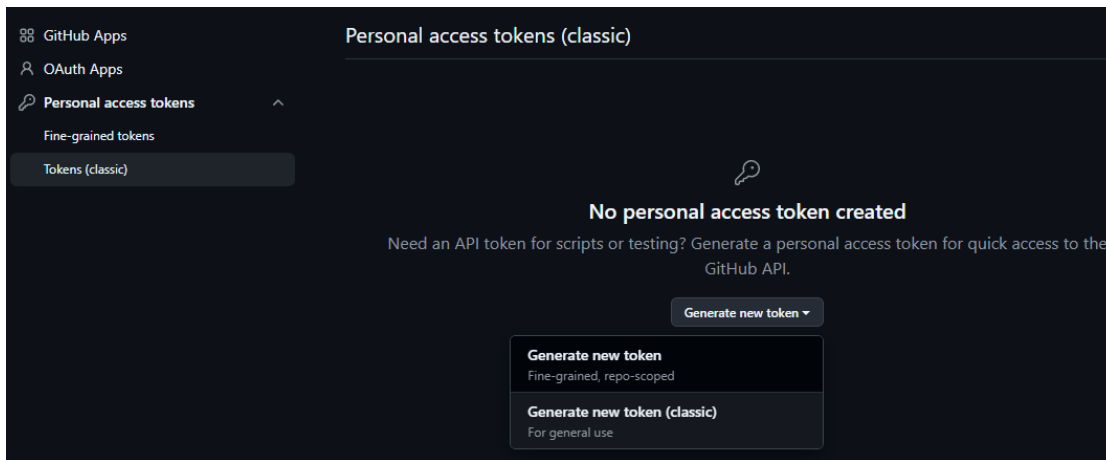
2. On the **left sidebar of options**, scroll all the way to the bottom and **select the last option, Developer settings**



3. Open **Personal Access Tokens** and select **Tokens (classic)**



4. Select **Generate new token** from the dropdown box in the middle



Important

Make sure to **select Generate new token (classic)** when creating our new token.

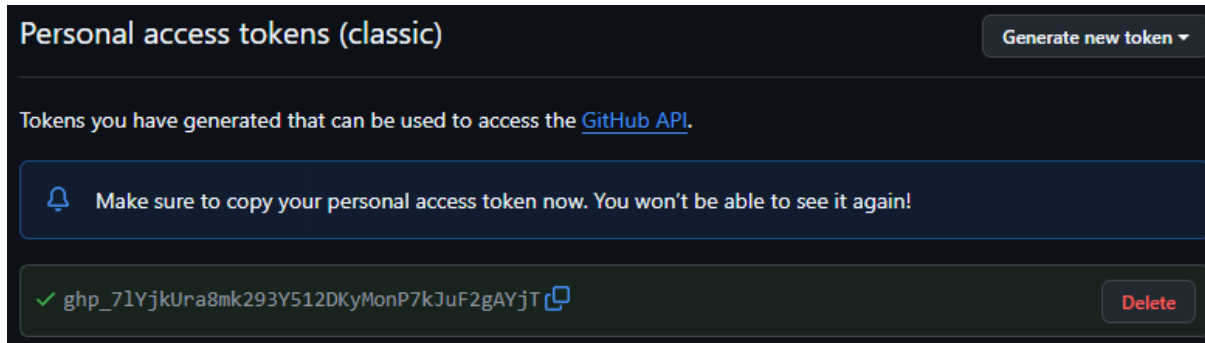
5. On the next screen, give our credentials any **Note** you want and set the Expiration to **No expiration**



Important

For scopes here, make sure to at least select **repo**. You can **select all of them** if you prefer!

- One last step here. On the next page, we will get a **Token hash** that typically starts with **ghp_....**
Make sure to **save this entire line in a file/notepad** somewhere!



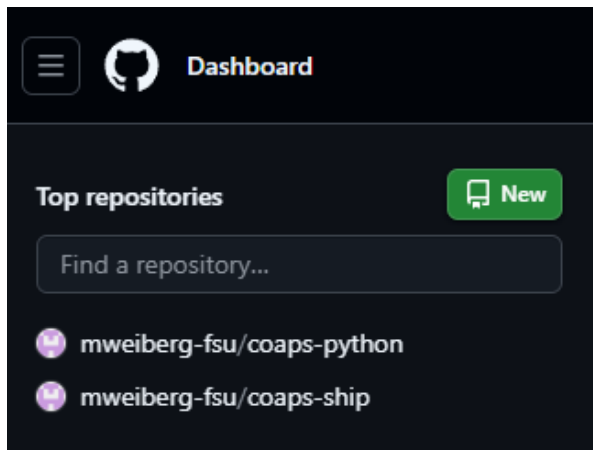
Important

The **Git username and email** should match what you made for your GitHub account earlier.

Step 3: Create a New GitHub Repo

After logging into GitHub, we can access our Dashboard on the left!

- Where it says **Top repositories** click on the green **New** button.



2. On the next screen, **give our repo a name and create it!**

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1 General

Owner * / Repository name *

mweiberg-fsu / coaps-test

coaps-test is available.

Great repository names are short and memorable. How about **effective-adventure**?

Description

0 / 350 characters

2 Configuration

Choose visibility *
Choose who can see and commit to this repository

Public

Add README
READMEs can be used as longer descriptions. [About READMEs](#)

Off

Add .gitignore
.gitignore tells git which files not to track. [About ignoring files](#)

No .gitignore

Add license
Licenses explain how others can use your code. [About licenses](#)

No license

Create repository



Tip

If you don't want your repo **to be public** to everyone, you can **set Visibility** from **Public** → **Private**. You can still invite **Collaborators** at a later date!

3. We want to run these two lines in a Mac terminal (or in your IDE terminal):

- `git config --global user.name "Your Name"`
- `git config --global user.email "your@email.com"`



Important

The **Git username and email** should match what you made for your GitHub account earlier.

4. Navigate to an empty directory on your computer (or make a new folder). Let's create our first GitHub repo locally! Run these commands in a terminal after navigating to this folder

- `echo "# coaps-test" >> README.md`
- `git init`
- `git add README.md`
- `git commit -m "first commit"`
- `git branch -M main`



Note

This will (1) **create an empty README.md file** we can add comments to, (2) **initialize our local repo**, (3) **add our new file** locally, (4) **create a commit** with a comment, and (5) **create a branch**

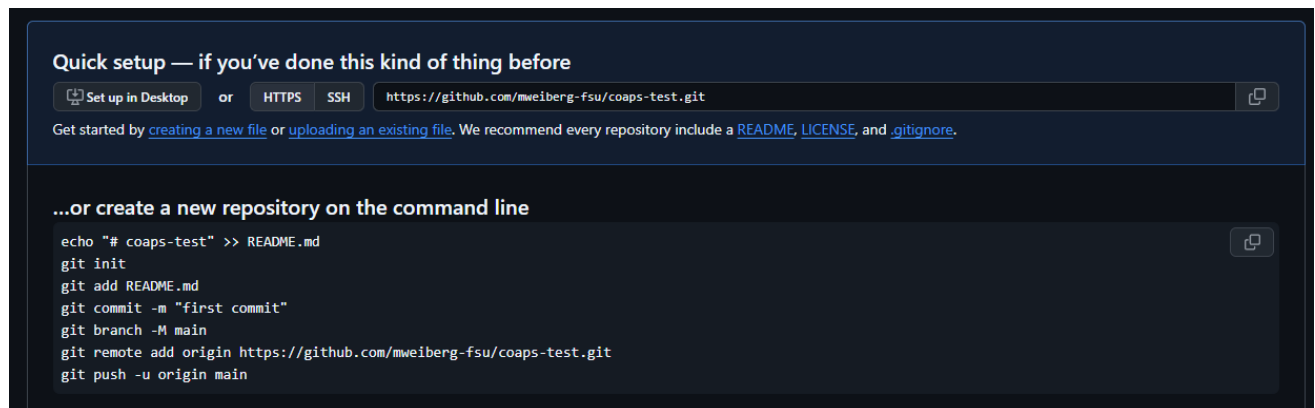


Tip

We can **make a directory** right from the terminal by using the command `mkdir [FOLDER_NAME]`

5. On the main GitHub repo page we just created, **you will have a unique url** (in addition to the commands we just ran above). **Copy that line** with the url:

- `git remote add origin https://github.com/mweiberg-fsu/change-this-to-your-repo.git`
- `git push -u origin main`

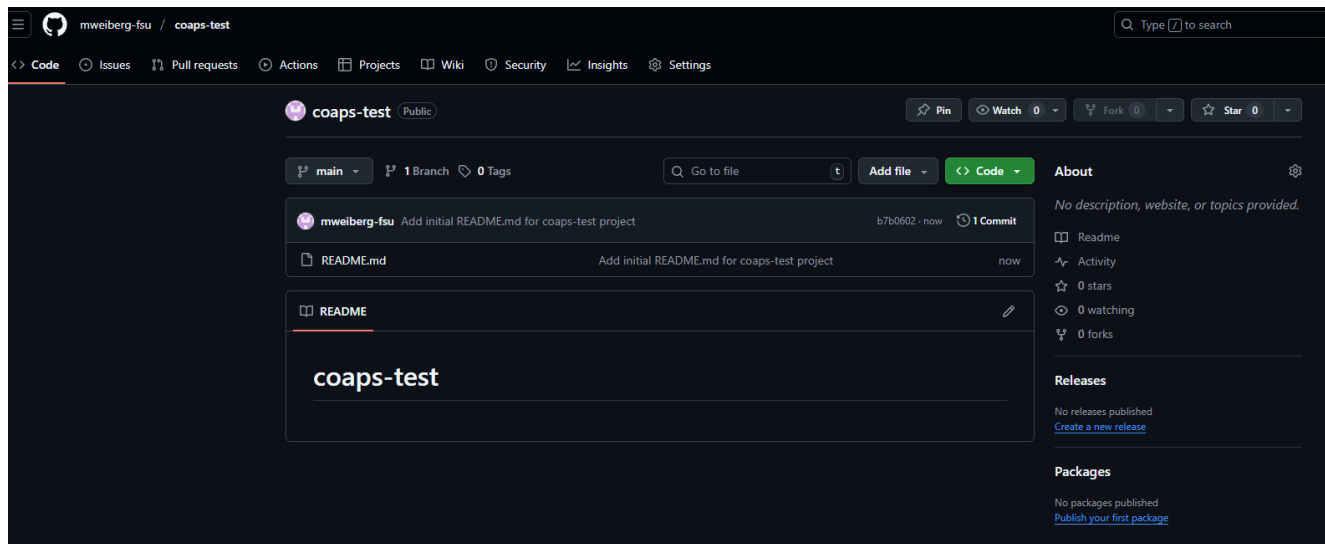


Important

The first time you send a **push** command, it will prompt you for a password. **Use that hash code** we saved earlier when we created a personal token!

```
(coaps-ship) PS D:\COAPS\python-tutorial> echo "# coaps-test" >> README.md
(coaps-ship) PS D:\COAPS\python-tutorial> git init
Initialized empty Git repository in D:/COAPS/python-tutorial/.git/
(coaps-ship) PS D:\COAPS\python-tutorial> git add README.md
(coaps-ship) PS D:\COAPS\python-tutorial> git commit -m "first commit"
[master (root-commit) cd2821a] first commit
1 file changed, 339 insertions(+)
 create mode 100644 README.md
(coaps-ship) PS D:\COAPS\python-tutorial> git branch -M main
(coaps-ship) PS D:\COAPS\python-tutorial> git remote add origin https://github.com/mweiberg-fsu/coaps-test.git
(coaps-ship) PS D:\COAPS\python-tutorial> git push -u origin main
```

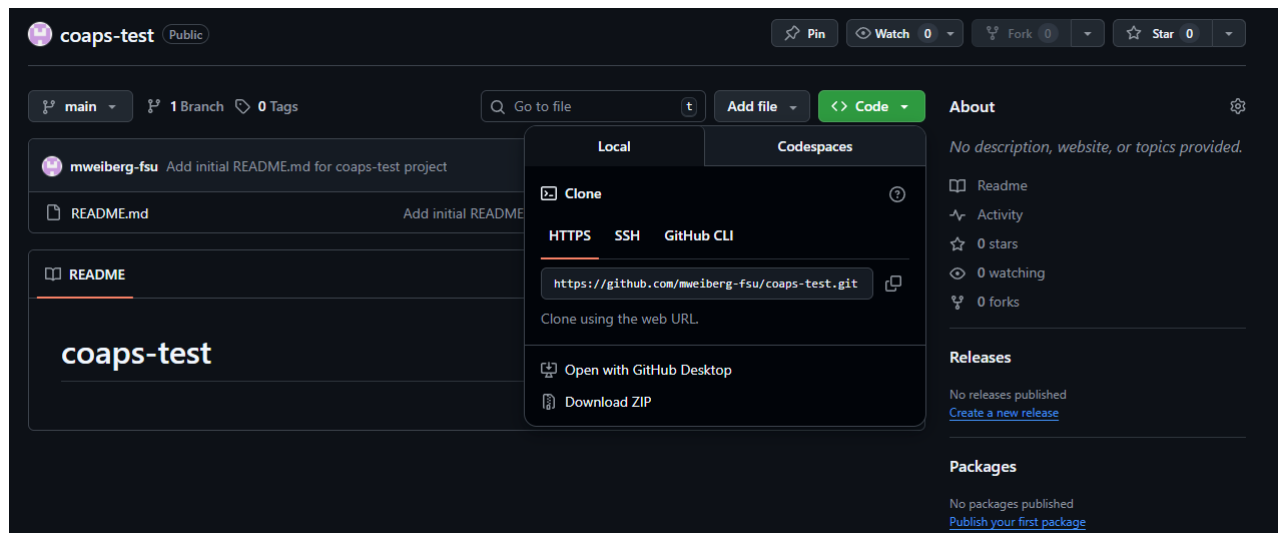
6. That's it, we're all set! **Navigate back to our GitHub repo** in a browser window, and you should see your files!



Step 4: Quick GitHub Commands to Remember

Here are some quick Git commands to use when **pushing and pulling** files from different devices.

1. Let's say we want to download our files on a laptop. If we **navigate back to our GitHub page**, we can select the **green <> Code button**, this will give us a URL to copy so we can clone our repo!



2. After **navigating to a directory** you want to save your repo at (see steps above), **run this command in a terminal**

➤ `git clone https://github.com/mweiberg-fsu/change-this-to-your-repo.git`



Tip

Make sure to change the url above to your unique url.

3. This should **download all the files** and **create a local GitHub repo** on your laptop now!
4. If you ever want to update your files and **push** them to your external GitHub repo, run these commands in succession

➤ `git add .`
➤ `git commit -m "add some comments on what you changed"`
➤ `git push`

That's it! You now can **keep your external and local GitHub repos up-to-date and share them** with other people or devices. This is great for putting projects on your resume at a later date as well!