ASTRON 98 Python DeCal
The Installation Guide

Created by: The Python DeCal Staff

Last Updated: August 24, 2025

Contents

1 Foreword

Welcome to the Python DeCal Installation Guide! This document will walk you through how to install all the software you'll need for the semester.

While we've tried to make every step as clear as possible, you might still run into a bug or error. This is totally normal. No stress! If anything goes wrong, please feel free to reach out to a Python DeCal staff member in class, on EdStem, or during office hours.

We are a bit short on time, so please try to get through these steps as soon as your can. This guide may look long, but that's mostly because we included lots of pictures to guide you through the installation process. Best of luck!

2 Mac or Windows?

The two most common operating systems people use are **Microsoft Windows** or **Apple macOS**, in other words, Windows users or Mac users. While both are under the category of operating system, I hate to say it, that's pretty much where the similarities end.

From here on out, a heads-up for Windows users but most of the tools we'll be using this semester are more Mac-friendly. But don't worry! We've included Windows-specific instructions when needed. Just be sure to follow the steps for you operating system, and you'll be fine.

3 The Terminal

First up on the menu is the **terminal**. The terminal is the most direct way to communicate with your computer and tell it exactly what you want it to do. However, it requires you to learn a specific kind of programming language called a **shell language**. If you've never used the terminal before, these language and feel pretty confusing and overwhelming at first.

"But wait", I here you say while reading this document, "I thought this course was called the Python DeCal, why am I learning a shell language?". Great question! Python is often run from the terminal, and as I mentioned before, the terminal gives you much more control over how your programs interact with your computer. Learning to navigate your computer through the terminal is a great way to become a more efficient programmer.

Over time, developers have created tools to make using the terminal easier. The most common of these is the **graphical user interface (GUI)**. See Figure 1 for an example of a Mac GUI and Figure 2 for a Windows GUI.

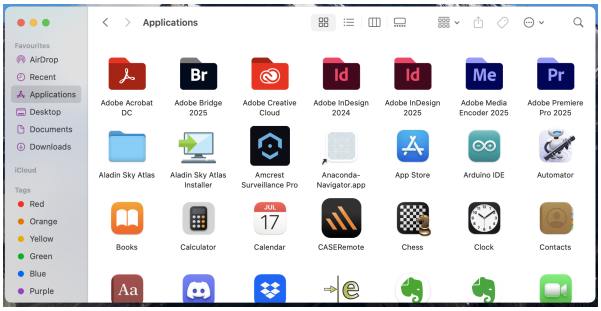

Figure 1: Mac GUI

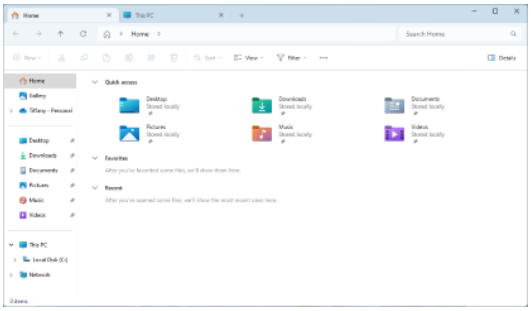

Figure 2: Windows GUI

The GUI lets you move through apps, folders, and files by clicking and scrolling. It's great for everyday use. But now that you're learning to program, we want more control over how we access and manipulate information on your computer.

What if you wanted to copy 1,000 files all at once? Rename every file in a folder based on a pattern? Or run a complex simulation with one command? Honestly, I'm not even sure how to do all that using just a GUI, but it is absolutely possible with the terminal. See Figures 3and 4 for what a Mac and Windows terminal look like.

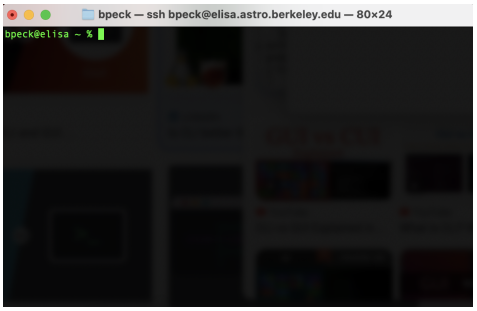

Figure 3: Mac Terminal

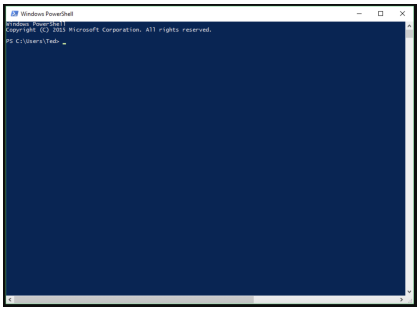

Figure 4: Windows Terminal

The Mac and Windows terminals might look pretty similar at first glance, but they actually speak two completely different shell languages. That's important to keep in mind, especially for Windows users, because you'll be installing a tool that creates a terminal which "speaks" the same language as a Mac terminal. That way, the Python DeCal staff only need to teach one shell language per semester instead of two and we can focus more on Python!

4 The Shell

Just to be clear: the **terminal** is the application you open, while the **shell** is the language the terminal speaks. Based on what kind of computer you're using, read the appropriate section below.

4.1 Windows Users: PowerShell

On a Windows computer, the default shell language is called **PowerShell**. No shade towards Microsoft, there's nothing inherently wrong with PowerShell, but we won't be working with it in this class. You're absolutely welcome to explore it on your own (with the help of Google or ChatGPT), but for consistency, we'll be using a Unix/Linux-based shell instead (which is what Mac and Linux terminals use by default).

In the next section, we'll walk you through how to install and use that shell on your Windows computer.

4.2 Mac Users: Zsh (Z Shell)

On a Mac, the default shell language is called **zsh (Z Shell)**. Good news: this is the exact shell we'll be using in the Python DeCal. That means your terminal is already set up to follow along with all the examples and shell scripting we'll do this semester, no extra work needed!

4.3 (BONUS) Linux Users: Bash

If you're on a Linux machine, haha cool. Your default shell is most likely **bash**, another Unix-based shell that works perfectly with everything we'll cover in this course. No additional setup should be needed.

(**Windows users:** bash is actually the shell language you'll be using this semester once your setup is complete. And for the purpose of this class, we'll treat zsh and bash as basically the same language.

5 Installing Git

Alright! So far, we've covered that the terminal is the application you open, and the shell is the language it speaks. Now let's introduce one more important term: the **command line**. This is the space inside the terminal where you actually type your commands, it's how you communicate with your computer using the shell language.

If you're on a Mac, you'll be using the command line in this section. Windows users, don't worry, you will get to use it very soon. And if you're generally feeling confused about what the terminal/shell/command line and how to use them, now worries, we will cover all of that in class.

The best analogy I have for **Git** is that it's like **Google Drive**, but for code. Git is a version control system that helps you track changes in your code and collaborate with other people. Just like Google Drive, it lets you see what was changed and when, go back to earlier versions and manage multiple versions of the same project. And if you're working on that project with other people, you can review their edits before running the program.

This might sound like overkill, but writing and collaborating on code is much harder than collaborating on a Google Doc. But Git makes it more manageable.

Windows users: You'll need to install a program called **Git Bash**. This gives you access to the Git version control system *and* installs a Unix-like shell language (bash) on your computer.

Mac users: Your computer already comes with a Unix-like shell (zsh), so all you need to do is install Git itself. If you are lucky, you might already have Git pre-installed and then you are actually done for this section! But read ahead to figure out if you do have Git pre-installed.

5.1 Windows Users

Follow the instruction below to download and set up Git Bash on Windows.

1. Go to the Git Bash installation page and click the **Download** button. *(See Figure 5)*

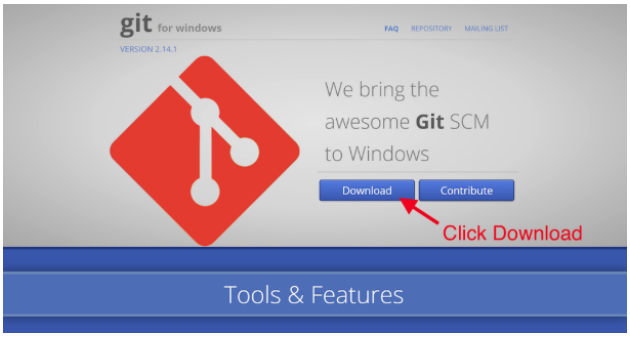

Figure 5:

2. Once Git Bash has finished downloading, run the `.exe` (executable) file by clicking on it. *(See Figure 6)*

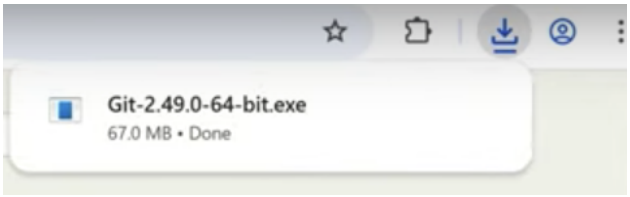

Figure 6:

3. When prompted, allow the application to make changes to your computer. *(See Figure 7)*

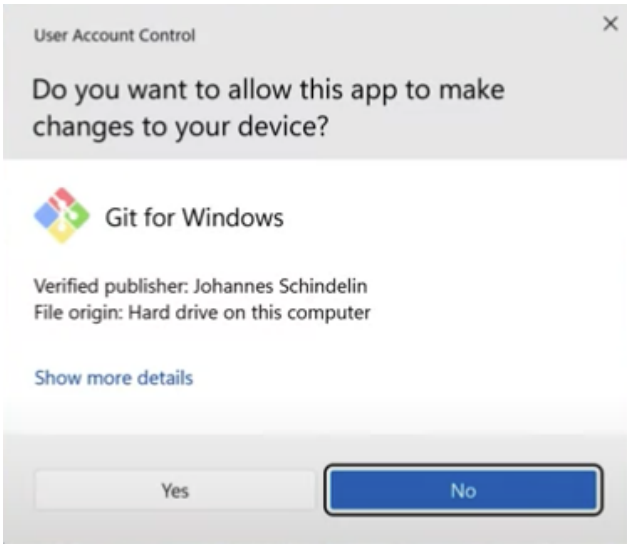

Figure 7:

4. Follow the default installation steps by repeatedly clicking **Next**, and finally **Finish**. *(See Figure 8)*

Figure 8:

5. To open Git Bash, go to the Windows Start Menu and type "Git Bash". Click on the Git Bash icon when it appears.
   (See Figure 9)

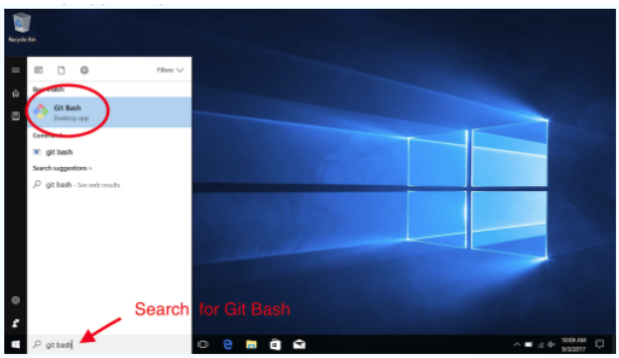

Figure 9:

6. A new window will open, this is your Git Bash terminal!
   (See Figure 10)

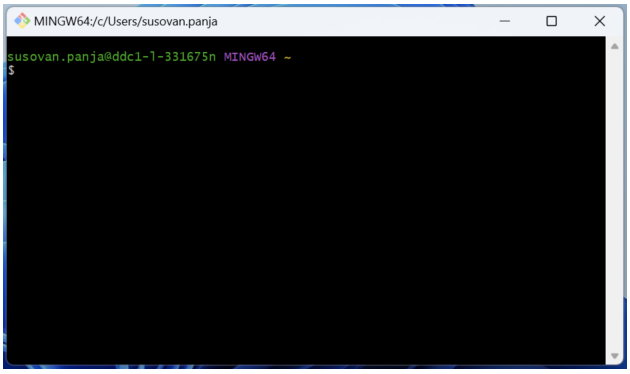

Figure 10:

7. **ATTENTION WINDOWS USERS!** Anytime we're working with the terminal in this class, make sure to open the Git Bash window, not PowerShell.

With Git Bash, you'll be using the same shell language as Mac users and will already have Git installed.

5.2 Mac Users

Follow the instructions below to install Git on a Mac. First, we'll check if Git is already installed, this is where you will need to use the command line!

1. Open the Terminal by going to **Finder** (the GUI we talked about earlier), clicking **Applications**, then **Utilities**, and finally opening the **Terminal** icon.
   (See Figure 11)

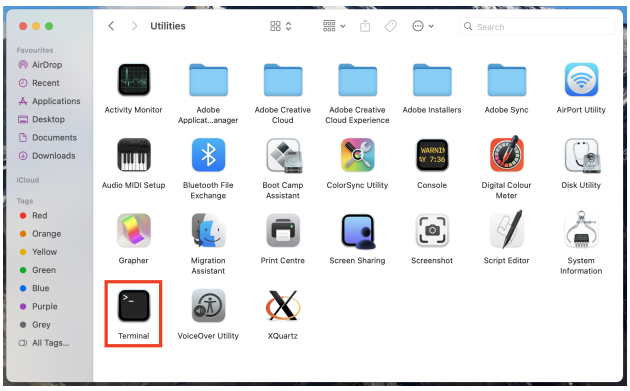

Figure 11:

2. A window will pop up. Type the following command and press **Return**.

```
git --version
```

   (See Figure 12)

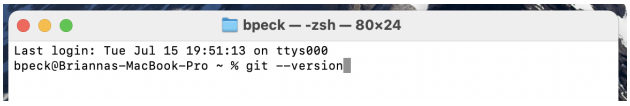

Figure 12:

3. If Git is already installed, the terminal will show the version number, which means that you are good to go! You can move on to the next section.
   (See Figure 13)

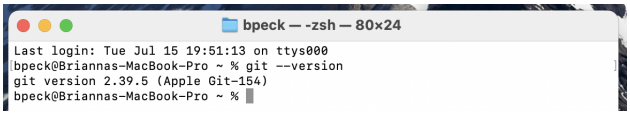

Figure 13:

4. If Git is *not* installed, you'll see an error message. No worries, just follow the next steps to install it.
   (See Figure 14)

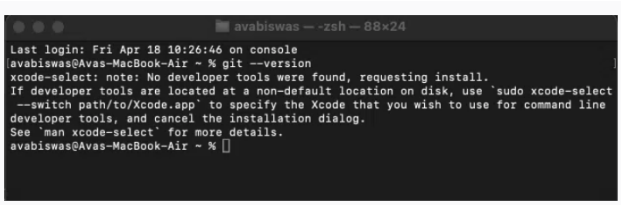

Figure 14:

5. We'll be installing Git using Homebrew. So we will install Homebrew first and Homebrew will install Git for you. Go to the Homebrew installation page.
   (See Figure 15)

Figure 15:

6. Click the clipboard (highlighted in red in Figure 16) to copy the installation command.
   (See Figure 16)

Figure 16:

7. Open your Terminal window again. (If you accidentally closed it since the last section, just go back to **Finder** → **Applications** → **Utilities** → **Terminal.**)

8. Paste the copied command into the command line and press **Return**.
   (See Figure 17)

Figure 17:

9. The terminal will ask you for your Mac password. When you start typing, you will notice that **NOTHING WILL SHOW UP!** That's totally normal, it just hiding your password for privacy. Once you've typed it, press **Return**.
(See Figure 18)

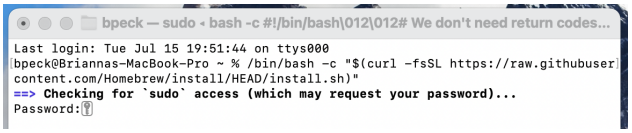

Figure 18:

10. After entering your password, press **Return** to begin installing Homebrew.
(See Figure 19)

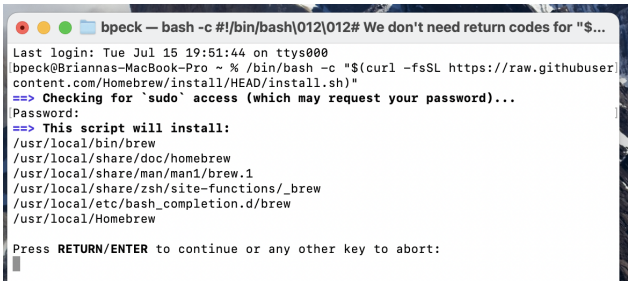

Figure 19:

11. The installation may take a while. You'll know it's done when your terminal looks like this:

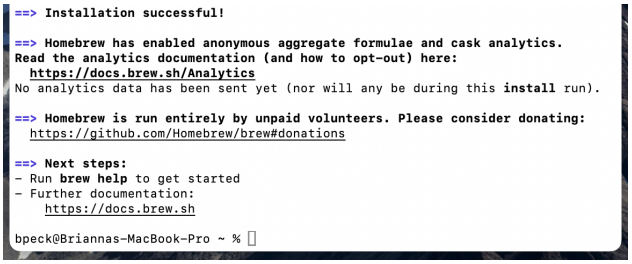

Figure 20:

12. To confirm that Homebrew was successfully installed, type:

```
brew --version
```

(See Figure 21)

Figure 21:

13. Now it's time to install Git. In the terminal, type:

```
brew install git
```

This may also take a few minutes.

14. Once the % prompt appears again, verify that Git is installed by running:

```
git --version
```

(See Figure 22)

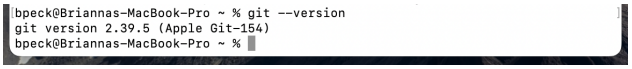

Figure 22:

6 Setting up GitHub

GitHub is a popular hosting platform that lets you store your personal Git coding projects in the cloud (on the internet). Github makes it much easier to back up your code, share it with others, and explore open-source projects created by other developers. There are lots of services like GitHub, but GitHub is free, widely used, and trusted by developers around the world.

Since this is just the installation guide, we won't cover how to use GitHub (or even the command line) day-to-day, like how you will use it to access homework or turn in assignments. That's because GitHub and the command line are very detail-oriented so instruction for how to use them deserve their own separate space. We've created a separate guide that walks you through those steps.

Please pay extra attention as you follow this guide, sometimes just a single typo can cause unexpected issues. I'm not trying to scare you, if you make a mistake, it's totally fine! Just ask for help, that's what the course staff is here for.

To connect Git (on your computer) to GitHub (on the internet), we'll use a much more secure method than passwords: **SSH keys.** SSH keys work in a **pair:**

- A **private key** is stored on your computer.
- A **public key** is stored on GitHub.

GitHub will only let your computer connect if they keys match, like a secret handshake. Though really this is done through the process of encryption.

Follow the instructions below to get everything set up.

Windows users: If you're a Windows user, make sure you're using Git Bash, not PowerShell. To open Git Bash, go to the **Windows Start Menu**, type **Git Bash**, and click the icon. A new Git Bash terminal window will open.

1. Go to `https://github.com/` and create an account with your **@berkeley.edu** email address. If you already have an account, you can skip this step. *(See Figure 23)*

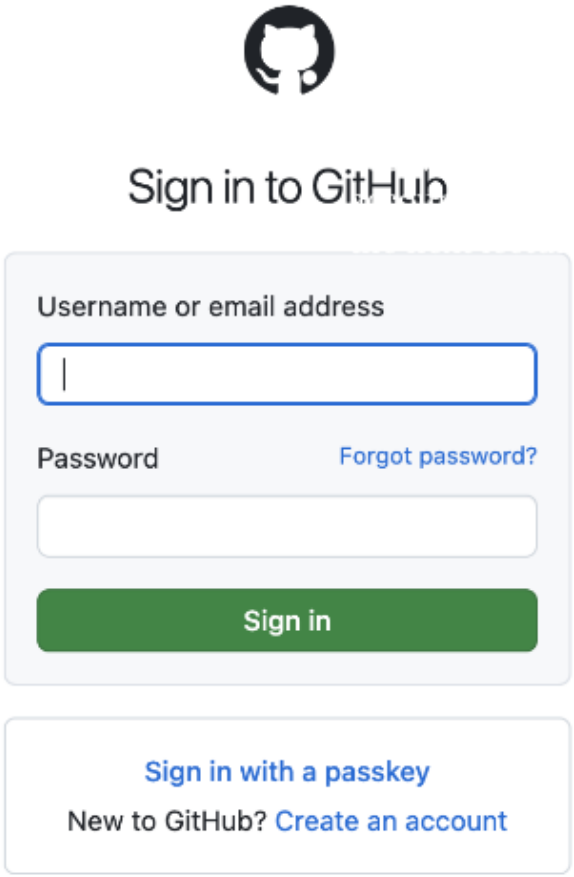

Figure 23:

2. After creating your GitHub account, open the terminal.
   Windows users: make sure you're using Git Bash, not PowerShell.

3. In the terminal, copy and paste the following command. Replace the example email with your GitHub email address (your @berkeley.edu email).

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

(See Figure 24)

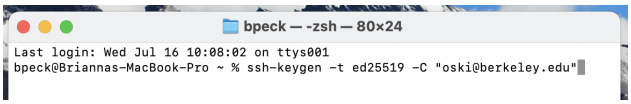

Figure 24:

4. When prompted to "`Enter a file in which to save the key`", just press **Return** to accept the default location.
   (See Figure 25)

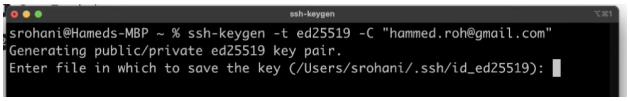

Figure 25:

5. Next, you'll be asked to create a passphrase (a password).
   Please remember your secure password. You'll have to enter it multiple times.
   Nothing will appear as you type, that's normal!
   (See Figure 26)

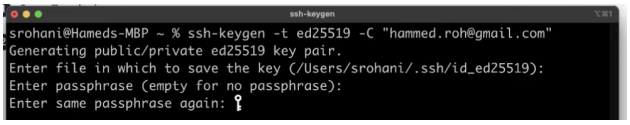

Figure 26:

6. After entering you password, you'll see a bunch of random symbols. That means your SSH key was successfully generated.
   (See Figure 27)

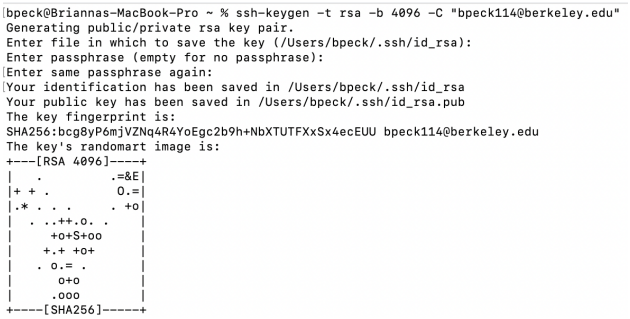

Figure 27:

7. Now, start the SSH agent in the background by running the following command in the terminal:

```
eval "$(ssh-agent -s)"
```

(See Figure 28)

Figure 28:

8. Add your new SSH key to the ssh-agent and store your passphrase in the keychain:

```
ssh-add ~/.ssh/id_ed25519
```

(See Figure 29)

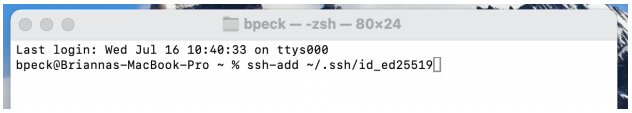

Figure 29:

9. Copy your SSH public key to your clipboard.

   On Windows:

   ```
   clip < ~/.ssh/id_ed25519.pub
   ```

   On Mac:

   ```
   pbcopy < ~/.ssh/id_ed25519.pub
   ```

   Nothing will appear after you run this, all good! Your key is now copied, just as if you called **Ctrl + C**.
   (See Figure 30)

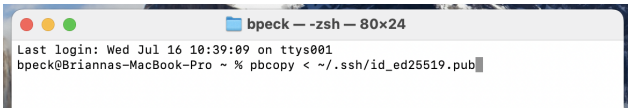

Figure 30:

10. In your web browser, open GitHub. Click your profile photo in the upper right corner.
    (See Figure 31)

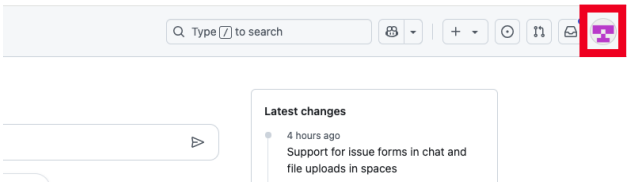

Figure 31:

11. Click **Settings** (a bit more than halfway down the menu).

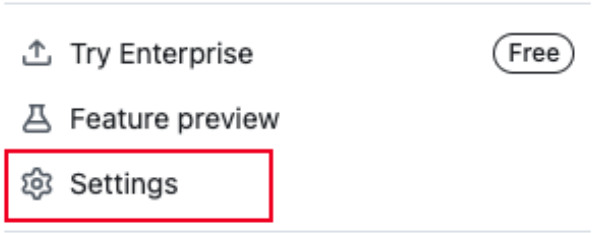

Figure 32:

12. In the left sidebar of the settings page, under **Access**, click **SSH and GPG keys**. *(See Figure 33)*

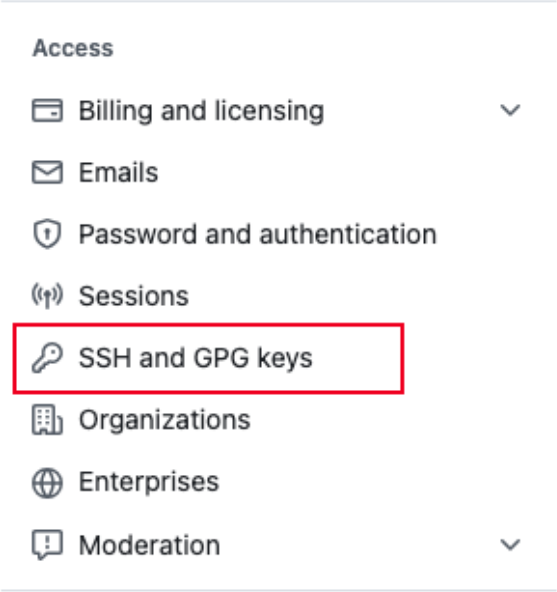

Figure 33:

13. Click **New SSH key** or **Add SSH key**. *(See Figure 34)*

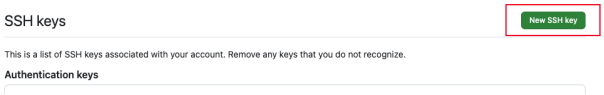

Figure 34:

14. In the **Title** field, give your key a name (e.g., "[your name]'s Personal Laptop).
    In the **Key** field, paste the key you copied earlier.
    (See Figure 35)

Figure 35:

15. Click **Add SSH key** to save it.
    (See Figure 36)

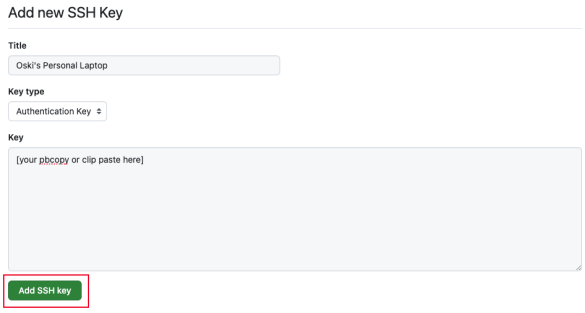

Figure 36:

16. Finally, test that your SSH key is connected by opening up the terminal and running:

    ```
    ssh -T git@github.com
    ```

    The terminal might ask: "`Are you sure you want to continue connecting (yes/no)?`"
    Type: `yes`
    (See Figure 37)

Figure 37:

17. Lastly, I would recommend setting up your git username and git email locally. Call:

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

    Replace `"Your Name"` with your name and `"your.email@example.com"` with your @berkeley.edu email address.

7 Anaconda (Jupyter Notebooks)

We are done installing Git and GitHub! Only a couple of installation to go, but these should be a bit easier.

Anaconda is a big software package that includes a lot of useful tools for this class and for scientific computing in general. One of the most important tools it comes with is **Jupyter Notebook**: a web-based, interactive environment where you can write and run Python code. We will also work with **Jupyter Lab** which provides a more interactive environment than Notebooks alone.

This is the software where you might be writing a majority of your Python code. Many assignments throughout the course will rely on Jupyter Notebooks.

Mac users: Before beginning, you need to check you Mac's chip type. In the top-left corner of your screen, click the Apple logo, then click **About this Mac**.
(See Figure 38)

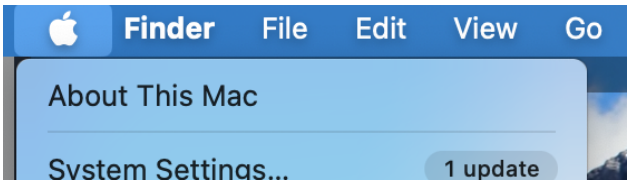

Figure 38:

Look at the line that says **Processor:**

- If it says something like "Apple M1" or "M2", you have a **Silicon** Mac.
- If it doesn't have an "M" (e.g., says "Intel"), you have an **Intel** Mac.

Keep this in mind as we go through the next steps.
(See Figure 39)

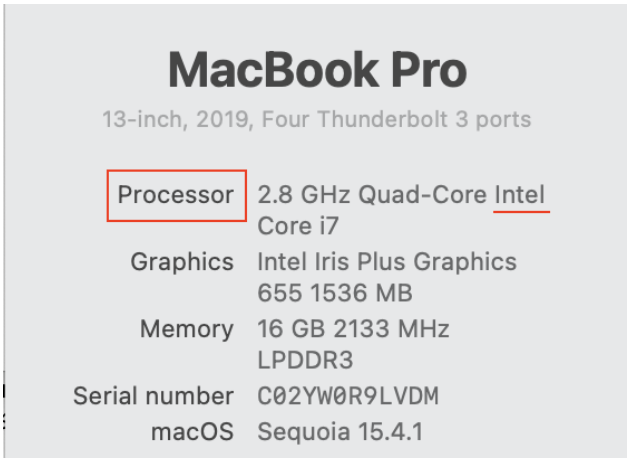

Figure 39:

1. Make sure you're in a place with strong WiFi. Otherwise, downloading and installing Anaconda might take a while. This is especially true if you're using eduroam during peak hours in a crowded location.

2. Go to the Anaconda Installation page.

3. Click **Skip registration** under "Provide email to download Distribution." *(See Figure 40)*

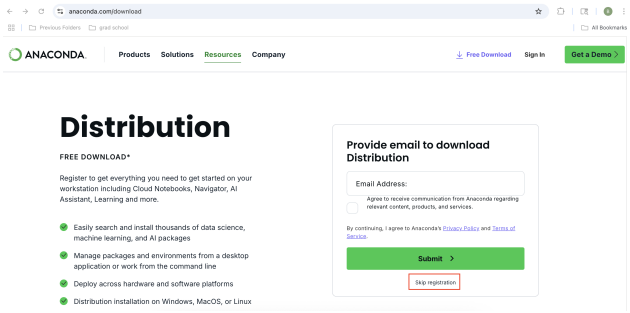

Figure 40:

4. Scroll down to the section labeled **Distribution Installers**.

5. **Mac users:**

   - Click **Download for Mac**
   - Then, based on your computer's processor type:
     - If you have an Apple Silicon Mac (M1, M2, etc.), click **Download for Apple Silicon**
     - If you have an Intel Mac, click **Download for Intel**

   Windows users:

   - Click the **Windows** tab (might be below **Download for Mac**).
   - Then click on **64-Bit Graphical Installer (914M)** to beging downloading Anaconda.

   (See Figure 41)

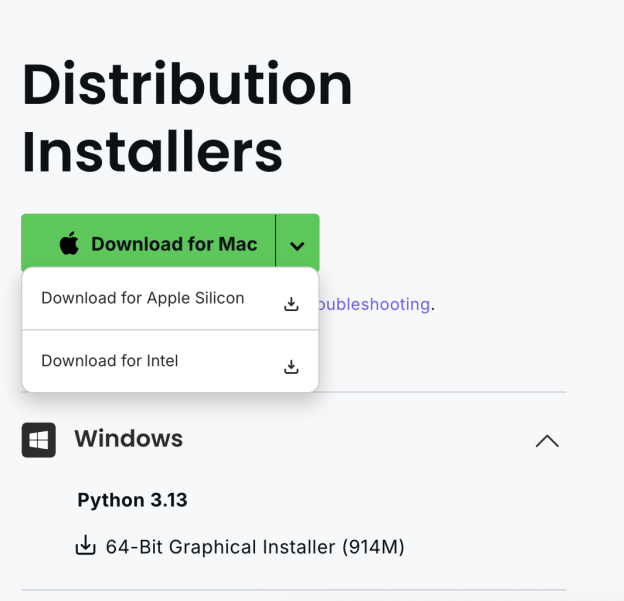

Figure 41:

6. Follow the installation instructions for your operating system (Mac or Windows).
   Heads-up: the download and setup might take a while, especially on show WiFi.

7. Once the installation is complete, open the **Anaconda Navigator** app.
   (See Figure 42)

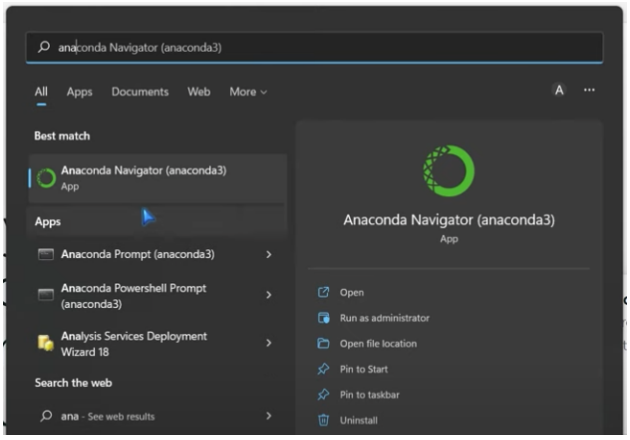

Figure 42:

8. Inside Anaconda Navigator, search for the app called **Jupyter Notebook** and click **Launch**.
   (See Figure 43)

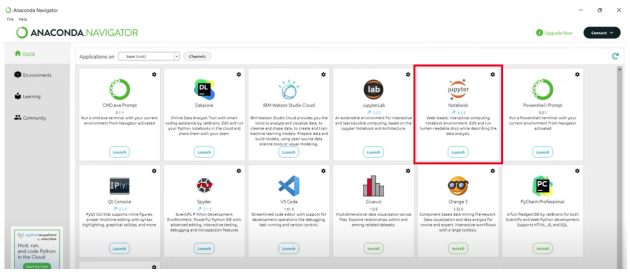

Figure 43:

9. If you see the screen below open in your browser, then you've successfully installed Jupyter Notebook!
   (See Figure 44)

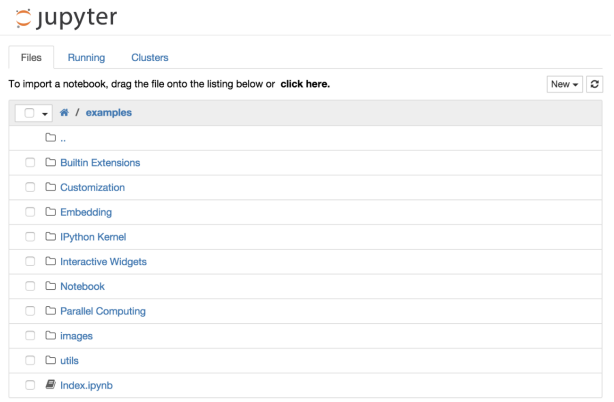

Figure 44:

8 Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is a code editor that's great for writing and running Python scripts. It's especially useful for larger programs or assignments that are too complex for a Jupyter notebook. For this course, you'll use VS Code for both assignments and projects.

1. Go to Visual Code Studio page.
2. Click the **Download** button and follow the installation instructions for your operating system. *(See Figure 45)*

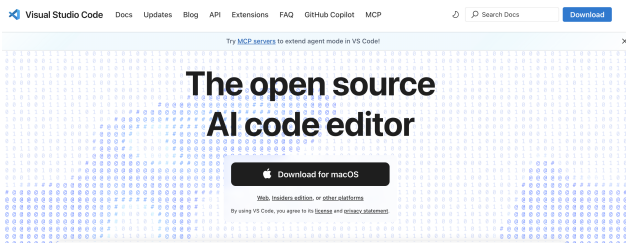

Figure 45:

3. Once installed, open the **VS Code** application. If you see the screen below, that means the installation was successful!
(See Figure 46)

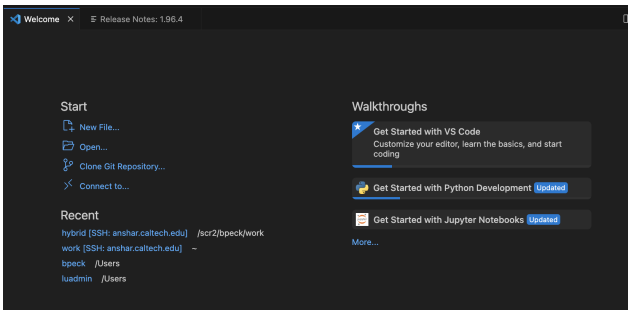

Figure 46:

4. Now we will install an extension. Click on **Extensions** (cube icon) on the left sidebar. *(See Figure 47)*

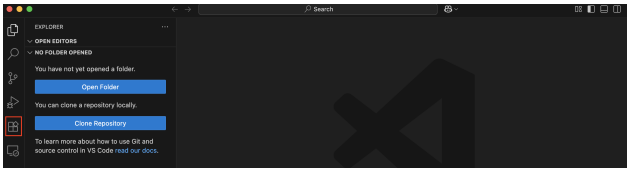

Figure 47:

5. Search **Python** and click **Install**. *(See Figure 49)*

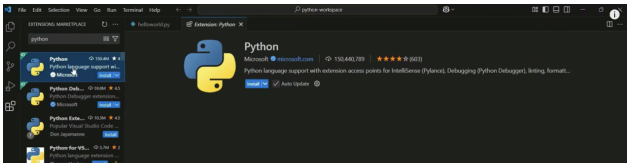

Figure 48:

6. **Windows users:** Change the language of your terminal in VS Code by calling **Ctrl + Shift + P**. A new search bar will open: *(See Figure 49)*

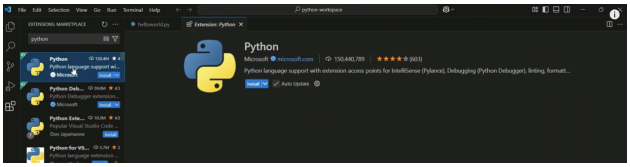

Figure 49:

Good job :) you are done with installations.