

# Walkthrough of Installing Everything Needed for AI-Assisted-Coding-In-Python Workshop

2025-11-04

## Table of contents

1 Checklist of Everything Needed to Install for AI-Assisted-Coding-In-Python .....	1
2 Installing GitHub .....	2
2.1 Optional: Two-Factor Authentication (2FA) .....	3
3 Sign Up for GitHub Copilot (THIS SECTION NEEDS WORK) .....	8
3.1 Free Version .....	8
3.2 Copilot Pro (Instructions for Verification) .....	8
3.3 Section on Getting Copilot Set Up (NEEDS WORK -> WAITING ON ACCESS) .....	16
4 Installing Python .....	17
4.1.a Mac .....	20
4.1.b Windows .....	20
4.1.c Linux .....	20
5 Install VSCode .....	21
5.1.a Mac .....	21
5.1.b Windows .....	21
5.1.c Linux .....	21
5.2 VSCode Documentation .....	23
6 Setting up Python in VSCode .....	23
7 Download Zip File .....	36

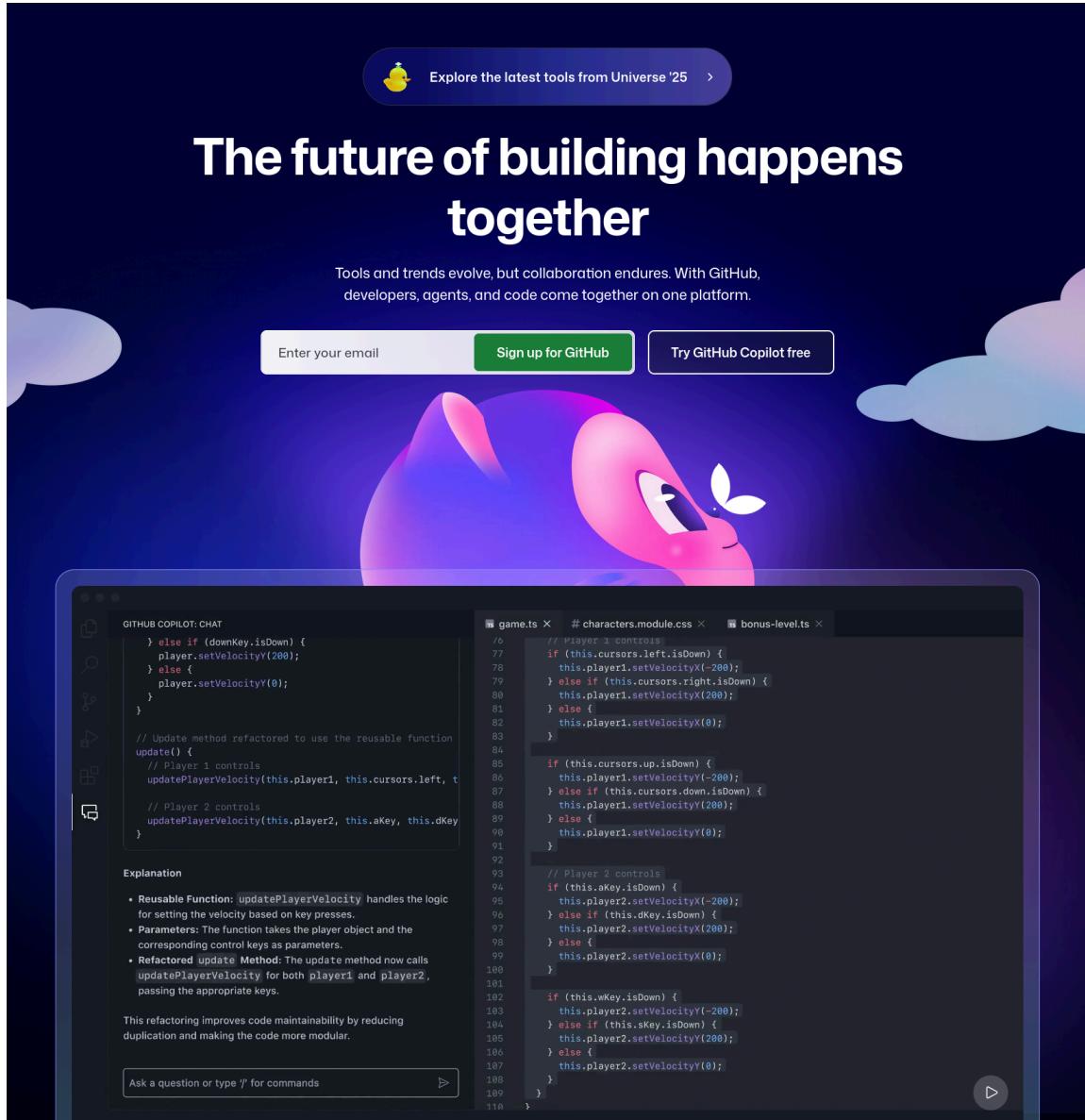
## 1 Checklist of Everything Needed to Install for AI-Assisted-Coding-In-Python

- [] Install GitHub
- [] Sign up for GitHub Copilot
- [] Install Python
- [] Install Visual Studio (VS) Code
- [] Adjust VSCode to work with Python
- [] Download Zip file

When these are completed, you are ready for your AI-Assisted-Coding-In-Python workshop.

## 2 Installing GitHub

Go to [GitHub](#) and sign up for a GitHub account.



```
GITHUB COPILOT: CHAT
    } else if (downKey.isDown) {
      player.setVelocityY(200);
    } else {
      player.setVelocityY(0);
    }

    // Update method refactored to use the reusable function
    update() {
      // Player 1 controls
      updatePlayerVelocity(this.player1, this.cursors.left, t
      // Player 2 controls
      updatePlayerVelocity(this.player2, this.aKey, this.dKey
    }

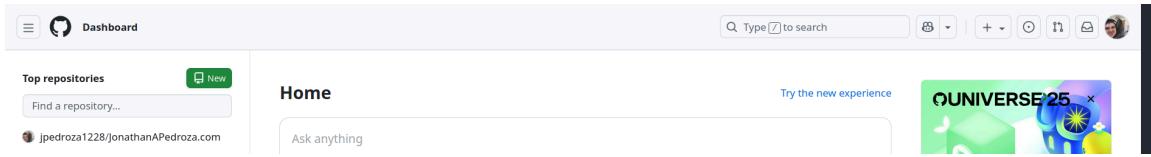
    Explanation
    • Reusable Function: updatePlayerVelocity handles the logic
      for setting the velocity based on key presses.
    • Parameters: The function takes the player object and the
      corresponding control keys as parameters.
    • Refactored update Method: The update method now calls
      updatePlayerVelocity for both player1 and player2,
      passing the appropriate keys.

    This refactoring improves code maintainability by reducing
    duplication and making the code more modular.

    Ask a question or type '?' for commands
```

At this page, you will sign up for GitHub with your @berkeley.edu email account. Follow the directions to verify your account. Below are some recommendations for creating a username (inspired by <https://happygitwithr.com/github-acct>).

- Use part of your real name so it is easier for people to know who you are
- Try and keep it short, you may have to type it a lot
- Keep everything lowercase. If you really want to separate words, use a hyphen (-) or an underscore (\_)

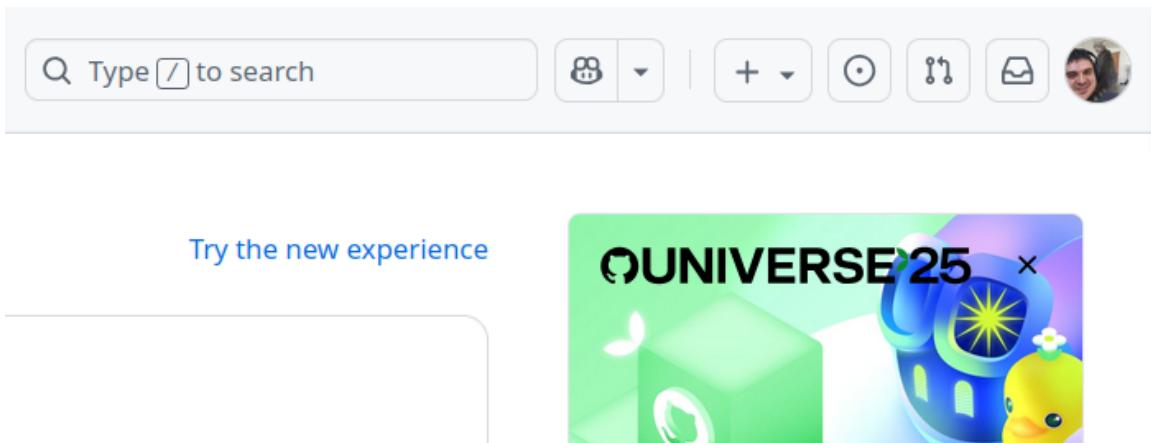


Once you sign in, you will be at your dashboard. You have now downloaded GitHub! Congrats!

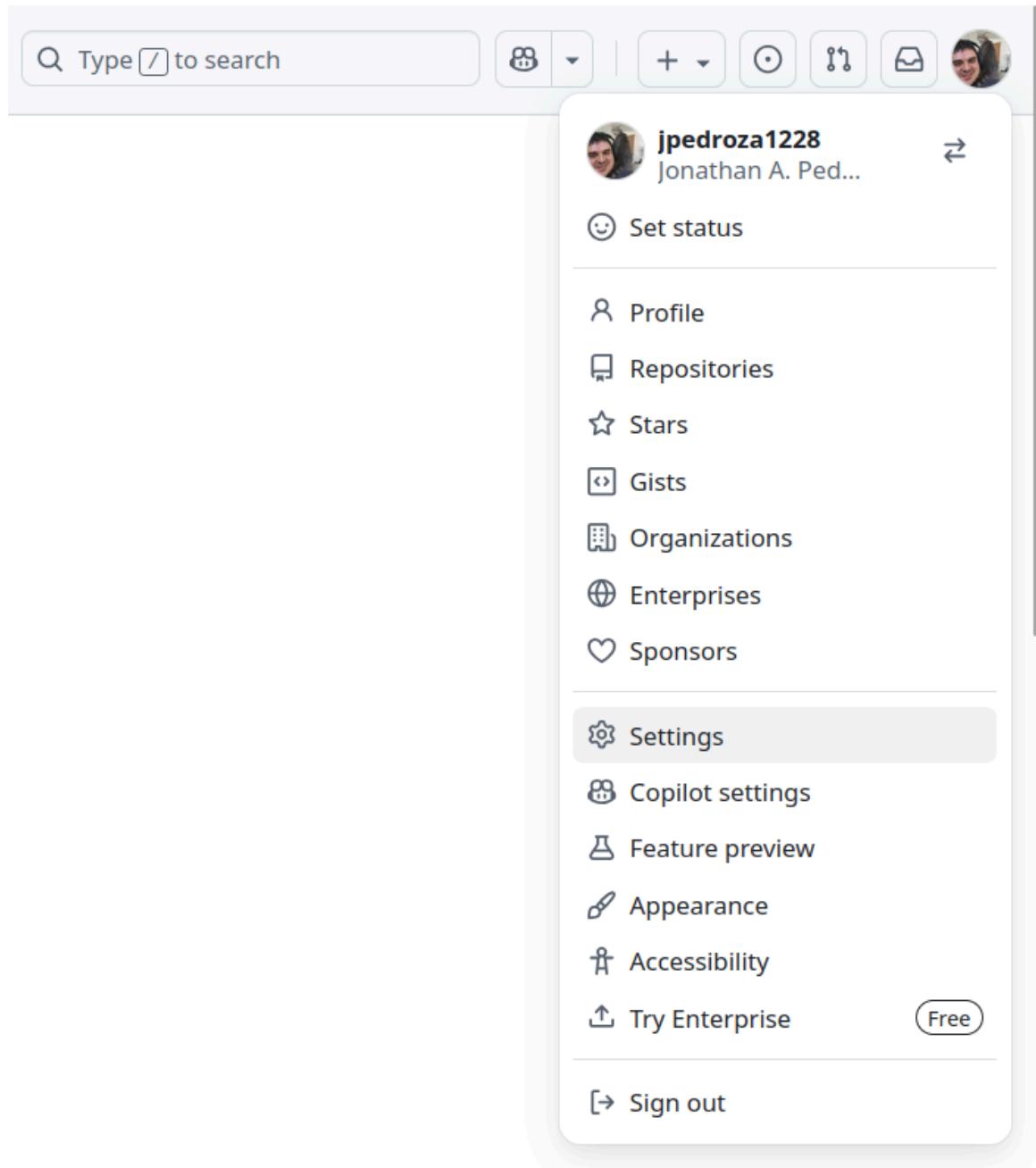
## 2.1 Optional: Two-Factor Authentication (2FA)

While not part of the tutorial, GitHub can house public and private data. If you are planning on continuing to use GitHub, please think about creating additional safeguards by setting up your Two-Factor Authentication (2FA). Below are some resources and a quick look into 2FA.

For more information on 2FA, you can find resources [here \(About 2FA\)](#) or [here \(Securing account with 2FA\)](#).



From your dashboard, you will want to go to your profile. If you just created your account, you will not have a profile picture. You'll then click on your profile circle to show a dropdown menu.



At this dropdown menu, you will go to your **Settings**.

 **Jonathan A. Pedroza (JP) (@jpedroza1228)**  
Your personal account

[Go to your personal profile](#)

**Public profile**

-  [Account](#)
-  [Appearance](#)
-  [Accessibility](#)
-  [Notifications](#)

**Access**

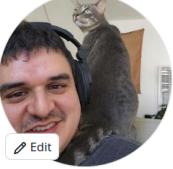
-  [Billing and licensing](#)
-  [Emails](#)
-  [Password and authentication](#)
-  [Sessions](#)
-  [SSH and GPG keys](#)
-  [Organizations](#)
-  [Enterprises](#)
-  [Moderation](#)

**Public profile**

**Name**

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

**Profile picture**



 [Edit](#)

---

**Public email**

 Remove

You can manage verified email addresses in your [email settings](#)

---

**Bio**

**Pronouns**



---

**URL**



---

**Social accounts**

 <https://bsky.app/profile/jonathanpedroza.bsky.social>

 <https://www.linkedin.com/in/jonathan-a-pedroza-phd-5721a7120/>

 [Link to social profile 3](#)

 [Link to social profile 4](#)

---

**Company**

You can @mention your company's GitHub organization to link it.

---

**Location**

**Display current local time**  
Other users will see the time difference from their local time.

---

**ORCID ID**

You have a connected ORCID ID 0009-0000-5276-0835 for the account @jpedroza1228.

**Display your ORCID ID on your GitHub profile**

Disconnecting your ORCID ID may affect areas of your profile where your ORCID ID is displayed.

 [Disconnect your ORCID ID](#)

At your settings, there will be a lot of information.

5

 **Public profile**

---

-  Account
-  Appearance
-  Accessibility
-  Notifications

---

**Access**

-  Billing and licensing 
-  Emails
-  Password and authentication
-  Sessions
-  SSH and GPG keys
-  Organizations
-  Enterprises
-  Moderation 

---

**Code, planning, and automation**

-  Repositories
-  Codespaces
-  Models 
-  Packages
-  Copilot 
-  Pages
-  Saved replies

---

**Security**

-  Code security

---

**Integrations**

-  Applications
-  Scheduled reminders

---

**Archives**

-  Security log
-  Sponsorship log

---

 Developer settings

On the left sidebar, you will want to go to **Password and authentication**. Here you can customize how you would like to sign into GitHub.

## Two-factor authentication

...

Two-factor authentication adds an additional layer of security to your account by requiring more than just a password to sign in. [Learn more about two-factor authentication](#).

**Preferred 2FA method**

Set your preferred method to use for two-factor authentication when signing into GitHub.

**Two-factor methods**

**Authenticator app** Configured

Use an authentication app or browser extension to get two-factor authentication codes when prompted.

**SMS/Text message** Less secure

Get one-time codes sent to your phone via SMS to complete authentication requests. We strongly advise against using SMS because it is susceptible to interception, does not provide resistance against phishing attacks, and deliverability can be unreliable. It is recommended to use an Authenticator app instead of SMS.

**Security keys**

Security keys are webauthn credentials that can only be used as a second factor of authentication.

**GitHub Mobile**

GitHub Mobile can be used for two-factor authentication by installing the GitHub Mobile app and signing in to your account.

**Recovery options**

**⚠ Your two-factor authentication recovery codes have not been downloaded or printed in the last one year. Make sure your recovery codes are up-to-date by viewing and downloading or printing them again.**

**Recovery codes** Viewed

Recovery codes can be used to access your account in the event you lose access to your device and cannot receive two-factor authentication codes.

While there are options for 2FA, I would recommend using an authenticator app. So every time you sign in (among other actions on GitHub), you will sign in with your username and password then verify it with a code from your authenticator app.

- 
- Install GitHub
  - Sign up for GitHub Copilot
  - Install Python
  - Install Visual Studio (VS) Code
  - Adjust VSCode to work with Python
  - Download Zip file

## 3 Sign Up for GitHub Copilot (THIS SECTION NEEDS WORK)

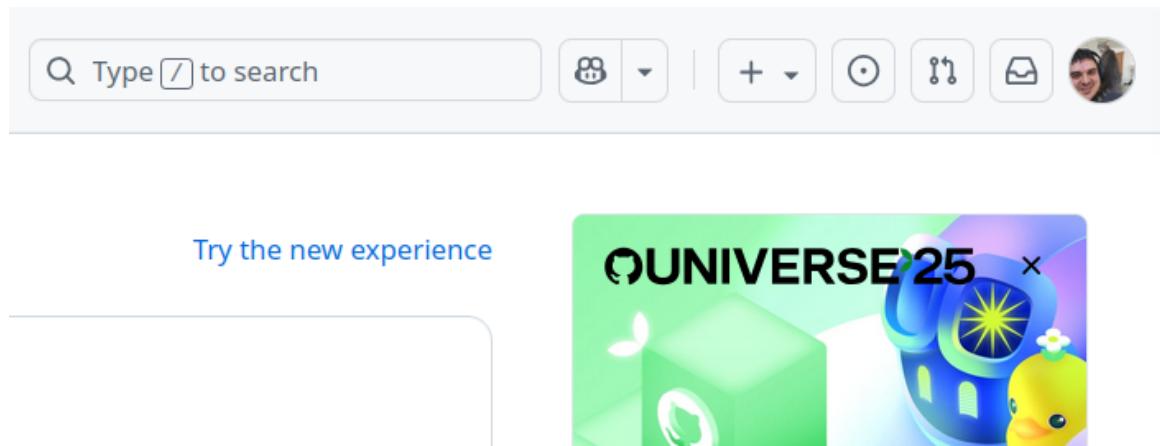
Signing up for GitHub Copilot will depend on whether you plan to sign up for the free version, CoPilot Pro using verified information, or pay for a Pro plan ([see pricing information here](#)).

### 3.1 Free Version

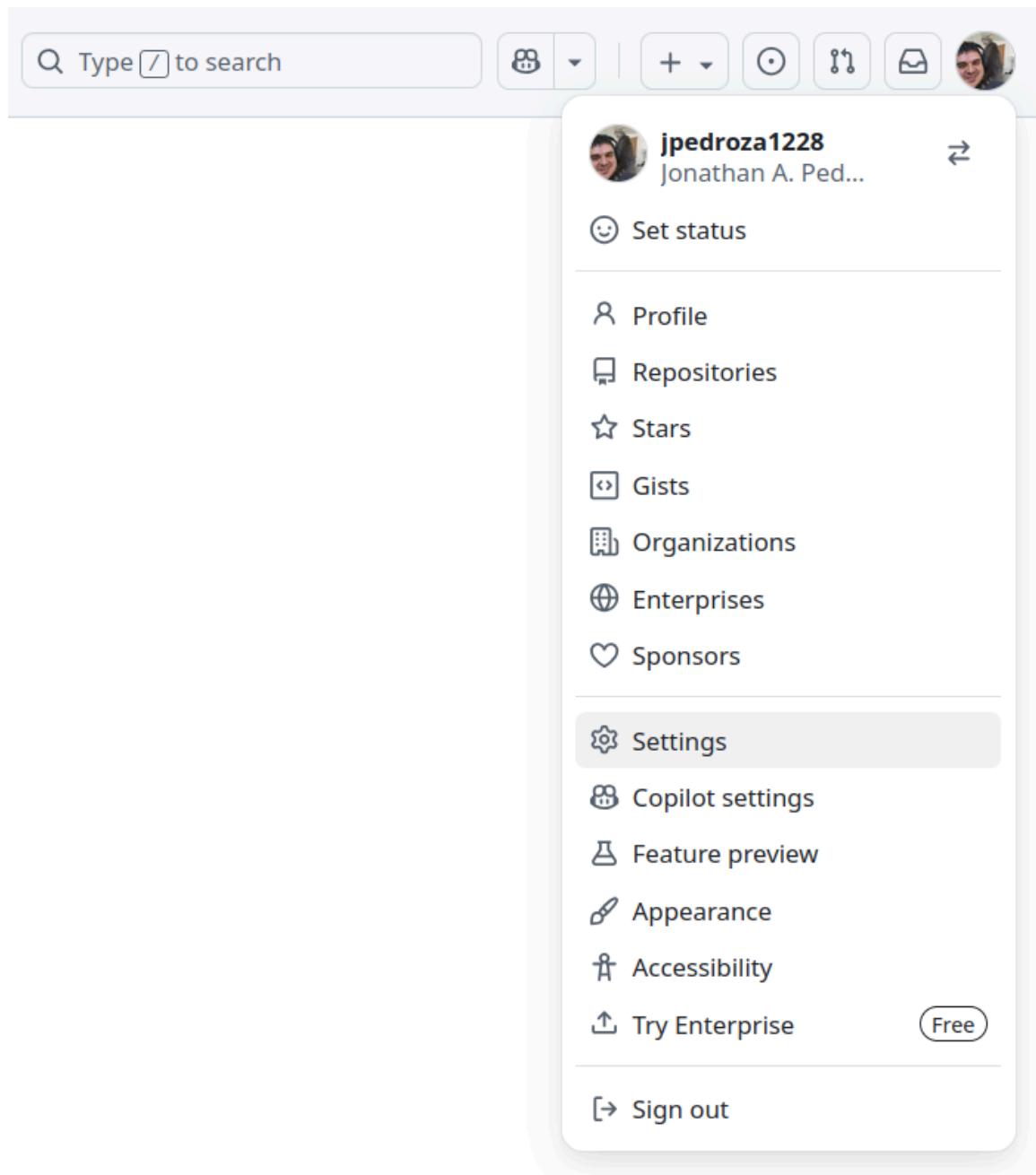
The free version of GitHub Copilot comes with VSCode. You can install the necessary extensions in the following section. You can also try out a [30-day trial here](#).

### 3.2 Copilot Pro (Instructions for Verification)

To get Copilot Pro for teachers and students (for free), you will need to follow the following steps. Below are some steps from the optional 2FA section above.



From your dashboard, you will want to go to your profile. If you just created your account, you will not have a profile picture. You'll then click on your profile circle to show a dropdown menu.



At this dropdown menu, you will go to your **Settings**.

 **Jonathan A. Pedroza (JP) (@jpedroza1228)**  
Your personal account

[Go to your personal profile](#)

**Public profile**

-  [Account](#)
-  [Appearance](#)
-  [Accessibility](#)
-  [Notifications](#)

**Access**

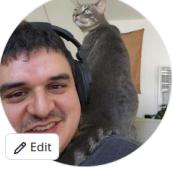
-  [Billing and licensing](#)
-  [Emails](#)
-  [Password and authentication](#)
-  [Sessions](#)
-  [SSH and GPG keys](#)
-  [Organizations](#)
-  [Enterprises](#)
-  [Moderation](#)

### Public profile

**Name**

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

**Profile picture**



 [Edit](#)

---

**Public email**

 Remove

You can manage verified email addresses in your [email settings](#)

---

**Bio**

**Pronouns**



---

**URL**



---

**Social accounts**

 <https://bsky.app/profile/jonathanpedroza.bsky.social>

 <https://www.linkedin.com/in/jonathan-a-pedroza-phd-5721a7120/>

 [Link to social profile 3](#)

 [Link to social profile 4](#)

---

**Company**

You can @mention your company's GitHub organization to link it.

---

**Location**

**Display current local time**  
Other users will see the time difference from their local time.

---

**ORCID ID**

You have a connected ORCID ID 0009-0000-5276-0835 for the account @jpedroza1228.

**Display your ORCID ID on your GitHub profile**

Disconnecting your ORCID ID may affect areas of your profile where your ORCID ID is displayed.

 [Disconnect your ORCID ID](#)

At your settings, there will be a lot of information. This time, you will click on the dropdown menu for **Billing and licensing**.

10



**Jonathan A. Pedroza (JP)**

Your personal account

**Public profile**

Account

Appearance

Accessibility

Notifications

**Access**

Billing and licensing ^

Overview

Usage

Premium request analytics

New

Budgets and alerts

Licensing

Payment information

Payment history

Additional billing details

Education benefits

Emails

Password and authentication

Sessions

From the dropdown menu, you will click on **Education benefits**.

The screenshot shows the GitHub account settings page for user Jonathan A. Pedroza (JP). The left sidebar includes links for Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and licensing (which is expanded to show Overview, Usage, Premium request analytics (New), Budgets and alerts, Licensing, Payment information, Payment history, Additional billing details, and Education benefits, the latter of which is highlighted with a blue bar). The main content area is titled "GitHub Education" and features a "Education Benefits" section with a graduation cap icon, describing it as "Complete a teacher or student application to unlock tools and resources for your educational journey." A green button labeled "Start an application" is visible.

Clicking on **Education benefits** will take you to GitHub Education. There you will start your application to get additional benefits, including GitHub Copilot Pro.



# Education Benefits Application

X

Select your role in education: \*

Teacher

Student



You have verified the email address on your GitHub account.  
That academic domain is associated with the school **University of California, Berkeley**.

Select this school

What is the name of your school? \*



If your school is not listed, then enter the full school name and continue. You will be asked to provide further information about your school on the next page. A minimum of two characters is required to find your school.

What is your school email address? \*

jpedroza1228@berkeley.edu



Have a different email address you use with your school? [Add it here.](#)

[Privacy Policy](#)

[Share Location](#)

[Continue](#)

Once you start your application, you will have the option of choosing your role At UC Berkeley. Below are general instructions for teachers and students; however, the instructions below will start to shift toward specific instructions for teachers.

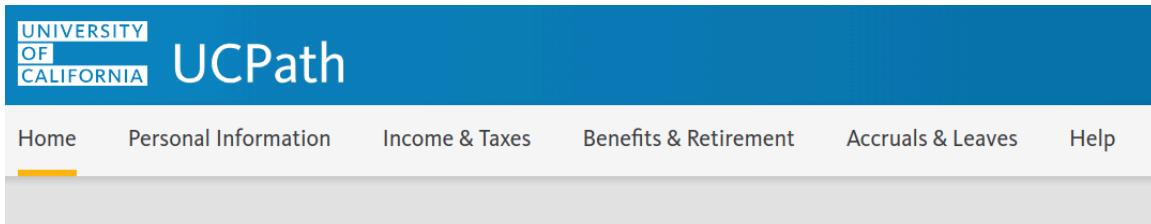
On the application, you can include the full name of the university (University of California, Berkeley) and when given an option to Select a school, click Select this school. You will also include your school email address in the dropdown menu. Make sure you are including your @berkeley.edu email address since verification relies on that email address.

The screenshot shows a web-based application titled "Education Benefits Application". At the top, there is a graduation cap icon and the title. A close button "X" is in the top right corner. Below the title, the text "Select your role in education: \*" is displayed. Two radio button options are shown: "Teacher" (selected, indicated by a blue outline) and "Student" (unselected, indicated by a grey outline). A green callout box contains the message: "You have verified the email address on your GitHub account. That academic domain is associated with the school **University of California, Berkeley**". Below this message is a button labeled "Unselect this school". At the bottom left, there is a button labeled "✓ Location shared". At the bottom right, there is a green button labeled "Continue".

You will also be asked to share your location, which will then allow you to click on Continue and finish this section of the application. You will then be asked for additional information in the form of proof of you being associated with UC Berkeley. The following information will be for teacher roles.

You will need to provide proof of your affiliation. **This proof must have your name, a current date, and the name of your institution on it.** If your application is rejected, it is most likely because your proof of affiliation was missing one of these three things (but, you can always apply again with new proof!). For example, some UC Berkeley student IDs do not have a date on them, so they will not be accepted. You may also have to follow some additional steps to verify your GitHub account, make sure to check your email for instructions. **Note** We have also heard of a bug that results in .png files not being accepted while .jpeg files are.

If you are a UC Berkeley student, the most straightforward way to get proof is to download a certificate of enrollment verification by going to CalCentral -> My Academics -> Enrollment Verification (under Academic Records) -> View or Print Enrollment via Self Service -> Obtain an enrollment certificate. This will give you a PDF enrollment certificate which you can screenshot and submit for proof of affiliation (you need to use a screenshot because the application does not accept PDFs).



For teacher roles, you will need to go to UCPath.

Payroll Information	Payroll Resources	Forms	Tax Statements
Paychecks	Expedited Pay Through Pay Card	Pay Card Consent	Federal Withholding (W-4)
Direct Deposit	Employee Calendars	Wage Payment Consent	CA State W-4 (DE-4)
Verification of Employment	Salary Overpayment Portal	Foreign Source Income eForm	Out-of-State Tax eForm
<a href="#">View Pay Record Via AYSO</a>			Online 1095-C Consent
			View Online 1095-C
			View Online W-2/W-2c
			Enroll to receive W-2/W-2c

From the main page, you will want to click on **Income & Taxes** and go to **Verification of Employment under Payroll Information**.

For your convenience, the University of California (UC) provides a simple method for employment verification. If you are applying for a loan, an apartment or job, your employment verifier (e.g. bank, leasing agent, or employer) accesses your employment information through The Work Number website.

**Please Note**  
Employees who opted out from sending their information to The Work Number must contact UCPath for assistance.

**The Work Number**  
The Work Number is a third-party provider of employment and income verification. All verifiers (banks, employers or leasing agents) must access your information through its website.

**How to Provide Proof of Your Employment and Income**  
Please provide your employment verifier the following information:

- Inform them that UC uses The Work Number
- Provide them the University of California Employer Code: 15975
- Provide them your Social Security Number

**Employment verification summary for employees only**  
If you simply need your employment information for your records, you may download a summary below.

**Employees who Opted Out**  
Employees who opted out from sending their information to The Work Number must refer verifiers to UCPath to complete employment & income verifications. Employment & income verifications for this population must be completed manually by UCPath.

**Verifiers may contact UCPath via**

- Email: ucpath@universityofcalifornia.edu
- Phone: 1-855-982-7284 (Monday through Friday, 8 a.m.– 5 p.m.)
- Fax: 1-855-982-2329

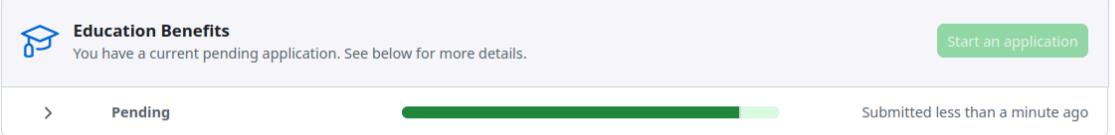
**UCPath data at The Work Number**  
UCPath data sent to The Work Number excludes Employees who opted out from sending their information. For all other populations, UCPath demographic data such as job record information is submitted daily and income information is submitted after each pay date.

[Generate Summary Report](#)

To get verification of your employment, you will then go to the bottom of the page and click on **Generate Summary Report**. You will be showed a pdf with your title, the current date of when

you generated report (today's date), and other information. Since you cannot submit PDFs, make sure you get a screenshot of the report and save it as a .jpg file.

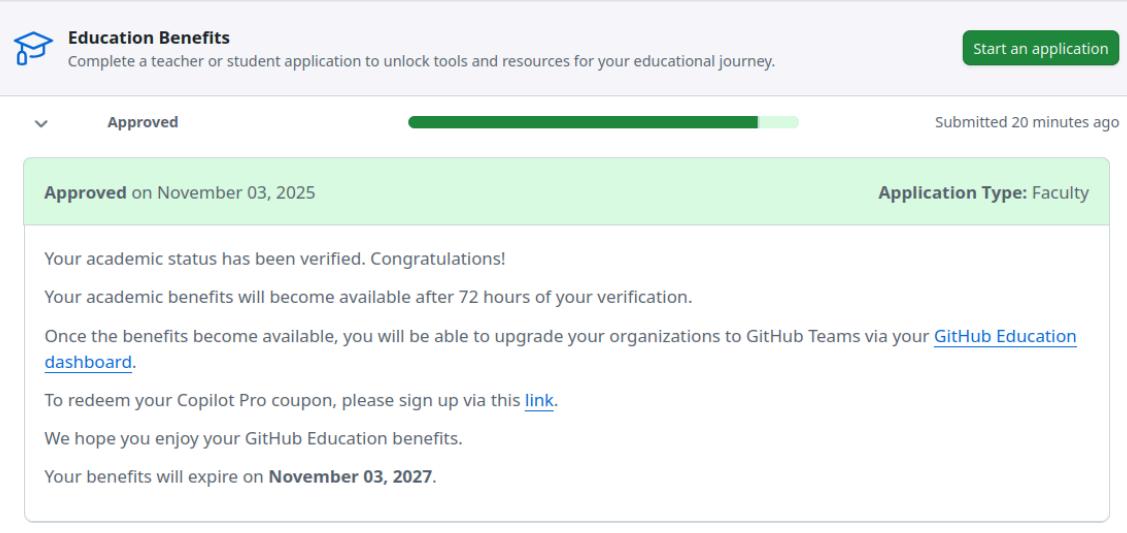
## GitHub Education



A screenshot of a GitHub Education application status page. At the top, there is a blue graduation cap icon followed by the text "Education Benefits". Below this, a message says "You have a current pending application. See below for more details." To the right is a green button labeled "Start an application". In the center, the word "Pending" is displayed next to a progress bar that is mostly filled with a dark green color. Below the progress bar, the text "Submitted less than a minute ago" is shown. There is also a small navigation arrow pointing left.

Once you have submitted your proof, your application will show that it is Pending.

## GitHub Education



A screenshot of a GitHub Education application status page showing an approved status. At the top, there is a blue graduation cap icon followed by the text "Education Benefits". Below this, a message says "Complete a teacher or student application to unlock tools and resources for your educational journey." To the right is a green button labeled "Start an application". In the center, the word "Approved" is displayed next to a progress bar that is almost entirely filled with a dark green color. Below the progress bar, the text "Submitted 20 minutes ago" is shown. A dropdown menu is open, showing "Approved" and "Rejected". On the right side of the page, under "Application Type: Faculty", it says "Approved on November 03, 2025". The main content area contains several messages: "Your academic status has been verified. Congratulations!", "Your academic benefits will become available after 72 hours of your verification.", "Once the benefits become available, you will be able to upgrade your organizations to GitHub Teams via your [GitHub Education dashboard](#).", "To redeem your Copilot Pro coupon, please sign up via this [link](#).", "We hope you enjoy your GitHub Education benefits.", and "Your benefits will expire on **November 03, 2027**".

After some time, it will state that you are approved. This is not a full approval, and full approval will take some time to gain access to Copilot Pro.

### 3.3 Section on Getting Copilot Set Up (NEEDS WORK -> WAITING ON ACCESS)

NEED TEXT HERE

- 
- Install GitHub
  - Sign up for GitHub Copilot
  - Install Python
  - Install Visual Studio (VS) Code

- [] Adjust VSCode to work with Python
- [] Download Zip file

## 4 Installing Python

1. Go to [anaconda to download Python](#)
2. Sign up (can go through Google, entering the information, or GitHub)
3. Verify your Email Address
4. Choosing installer

### Distribution Installers

[Δ Download](#)

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

**Windows**



**Mac**



**Linux**



### Miniconda Installers

[Δ Download](#)

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

**Windows**



**Mac**



**Linux**



 Free Download

Sign In

Get Demo >

## Download Now

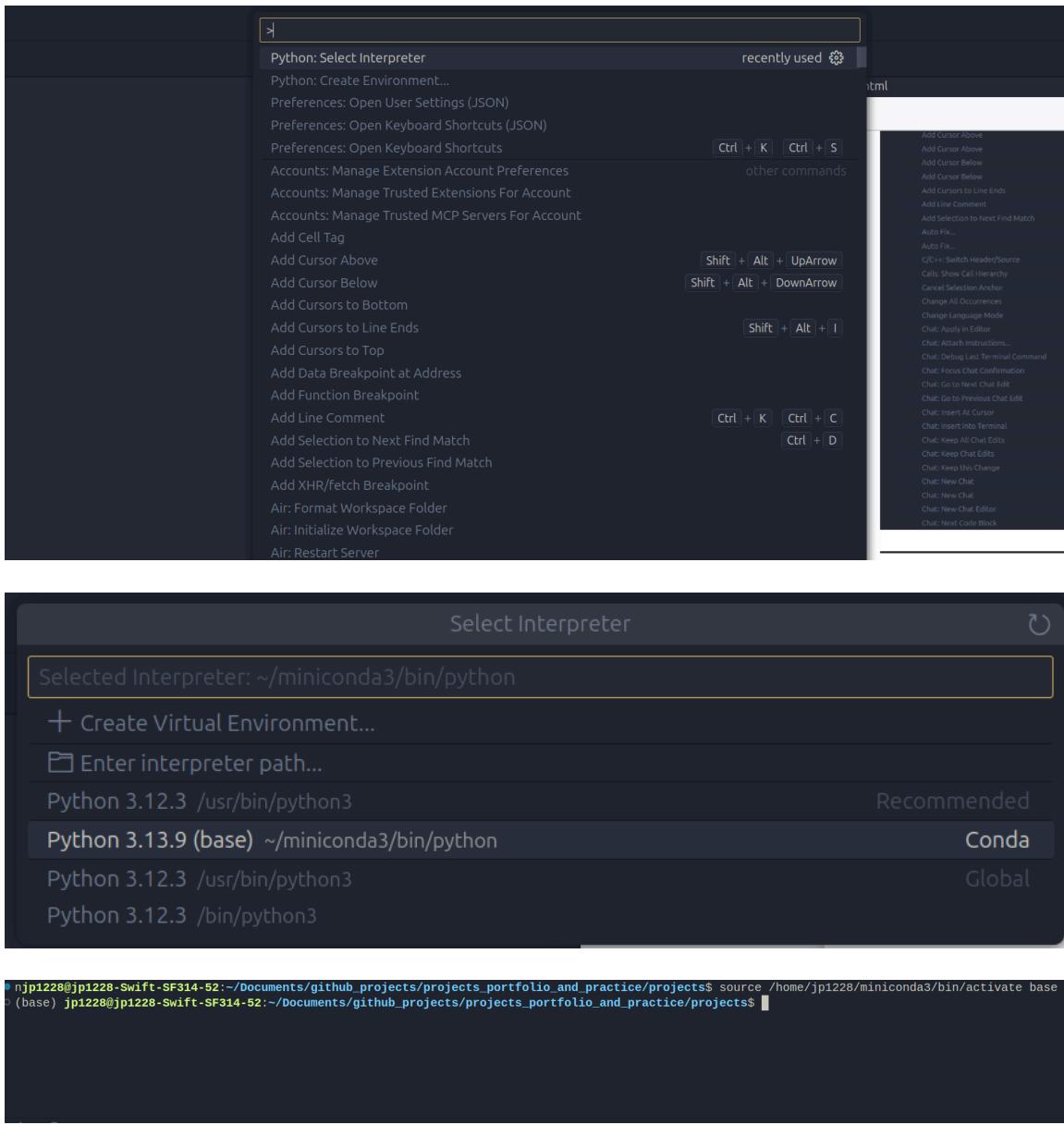
Get access in 30 seconds. Completely free.\*

Get Started >

Returning Users >

\*Subject to our [Terms of Service](#). Use of Anaconda's offerings at an organization of more than 200 employees/contractors requires a paid business license unless your organization is eligible for discounted or free use. See [Pricing](#).

```
jp1228@jp1228-Swift-SF314-52:~$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
--2025-11-03 16:08:52-- https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 2606:4700::6810:bf9e, 2606:4700::6810:20f1, 104.16.32.241, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|2606:4700::6810:bf9e|:443... [
```



## Linux Installation

Go here (<https://www.anaconda.com/docs/getting-started/miniconda/install>)

Find the linux code to install miniconda

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

You can then install the file by running the following code and answering the prompts in the terminal. You can answer yes and press enter to keep the default values.

```
bash ~/Miniconda3-latest-Linux-x86_64.sh
```

Closing out of your terminal, you will be ready to use Python in VSCode.

Choosing an interpreter

Go to the command palette (Windows/Linux: Ctrl + Shift + P; Mac: Cmd + Shift + P) and search for Python: Select Interpreter.

Now when you open a bash terminal, you will see that it starts with:

```
source <miniconda-path-location>/bin/activate base
```

That means you are now using conda as your Python interpreter. This should translate to using the Jupyter notebook for the workshop materials.

More information can be found on [environments here](#), as well as [general start up assistance with Python in VSCode](#).

To install Python, you will go to [The Comprehensive R Archive Network \(CRAN\) website](#). There you can download Python for your operating system.

#### 4.1.a Mac

Follow the directions for installing Python. You can keep the defaults for everything. When installed, you should see a message stating The installation was successful.

#### 4.1.b Windows

Follow the directions for installing Python. You can keep the defaults for everything. When installed, you should see a message stating The installation was successful.

#### 4.1.c Linux

There are some really good instructions on how to [install R here](#).

```
sudo apt update  
sudo apt install -y build-essential libcurl4-openssl-dev libssl-dev libxml2-dev  
  
# optionally  
# sudo apt install -y libfontconfig1-dev libharfbuzz-dev libfribidi-dev  
libfreetype6-dev libpng-dev libtiff5-dev libjpeg-dev
```

- 
- [] Install GitHub
  - [] Sign up for GitHub Copilot
  - [] Install Python
  - [] Install Visual Studio (VS) Code
  - [] Adjust VSCode to work with Python
  - [] Download Zip file

## 5 Install VSCode

Let's move forward with installing VSCode. You can install [VSCode here](#) for your operating system. Below are detailed instructions on how to install VSCode.

### 5.1.a Mac

Link: <https://code.visualstudio.com/docs/setup/mac>

Follow the directions to install VSCode. Below, I will include some helpful tips for VSCode extensions that may help when using Python.

### 5.1.b Windows

Link: <https://code.visualstudio.com/docs/setup/windows>

Follow the directions to install VSCode. Below, I will include some helpful tips for VSCode extensions that may help when using Python. **For Windows users, it is recommended to check “Save version number in registry” during installation so that the R extension can find your R installation automatically. If you have not done this you may need to add the location of your R to your PATH manually (see FAQ 3.1 I am using windows and my VS Code can't find R!).**

### 5.1.c Linux

**Note:** Everything below shows installation using Linux Mint.

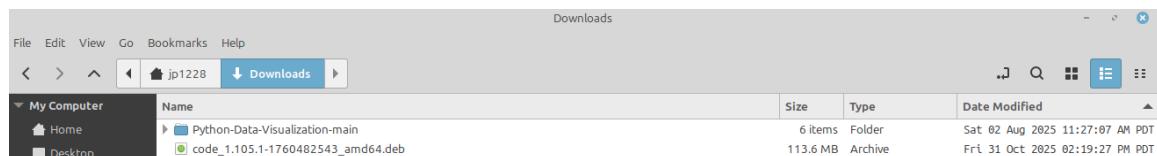
#### System:

```
Host: jp1228-Swift-SF314-52 Kernel: 6.8.0-87-generic arch: x86_64 bits: 64
Desktop: Cinnamon v: 6.4.8 Distro: Linux Mint 22.1 Xia
```

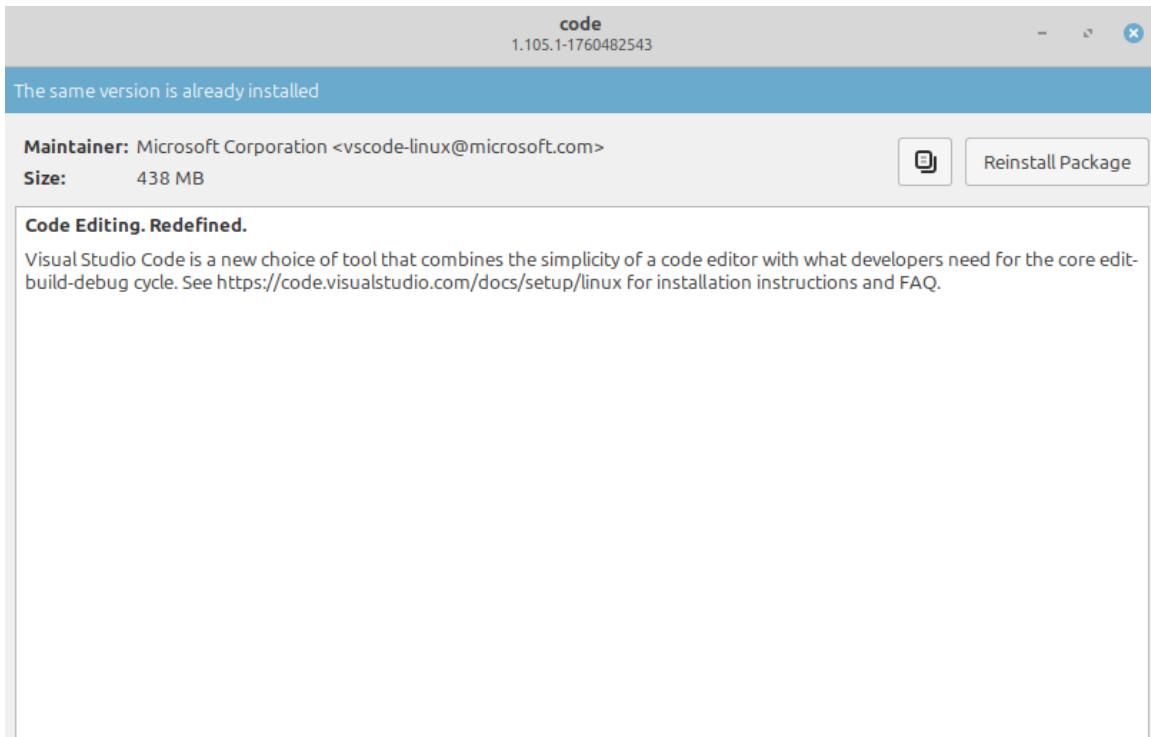
Link: <https://code.visualstudio.com/docs/setup/linux>

#### Option 1: Use the Link

1. Click on the link above and open the download file.



2. Click Install Package to start the install.



## Option 2: Download Using Terminal

1. Update your programs.

A screenshot of a terminal window with a black background and white text. The prompt shows "jp1228@jp1228-Swift-SF314-52:~". The user has typed the command "sudo apt update" and is waiting for the output.

2. Then you will install the file that you downloaded from [this page](#) as shown below. Your file will look different, depending on the version and differences in your linux distribution, but it should start downloading after running the code below.

```

sudo apt install ./<file_name>.deb
# include the name of your file and change <file_name> to the name of your file

```

```

Hit:4 https://cloud.r-project.org/bin/linux/ubuntu noble-cran40/ InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 https://packages.microsoft.com/repos/code stable InRelease
Get:7 https://s3.amazonaws.com/repo.deb.cyberduck.io stable InRelease [3,245 B]
Hit:8 http://archive.ubuntu.com/ubuntu noble InRelease
Get:9 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:11 https://packages.cloud.google.com/apt cloud-sdk InRelease
Hit:12 https://repository.spotify.com stable InRelease
Get:13 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Err:7 https://s3.amazonaws.com/repo.deb.cyberduck.io stable InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY FE7097963FEFBET2
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main i386 Packages [545 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1,573 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:18 http://archive.ubuntu.com/ubuntu noble-updates/universe i386 Packages [1,498 kB]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [940 B]
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [7,144 B]
Get:22 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7,144 B]
Get:23 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:24 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.0 kB]
Get:25 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:26 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:27 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:28 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.3 kB]
Get:29 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Reading package lists... Done
W: GPG error: https://s3.amazonaws.com/repo.deb.cyberduck.io stable InRelease: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY FE7097963FEFBET2
E: The repository 'https://s3.amazonaws.com/repo.deb.cyberduck.io stable InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
jp1228@jp1228-Swift-SF314-52:~$ sudo apt install ./code_1.185.1-1768482543_amd64.deb

```

## 5.2 VSCode Documentation

When you installed VSCode, it should have brought you to the documentation page. If not, you can find all the [documentation here](#). [This tutorial](#) also provides an in-depth tutorial on getting started with VSCode.

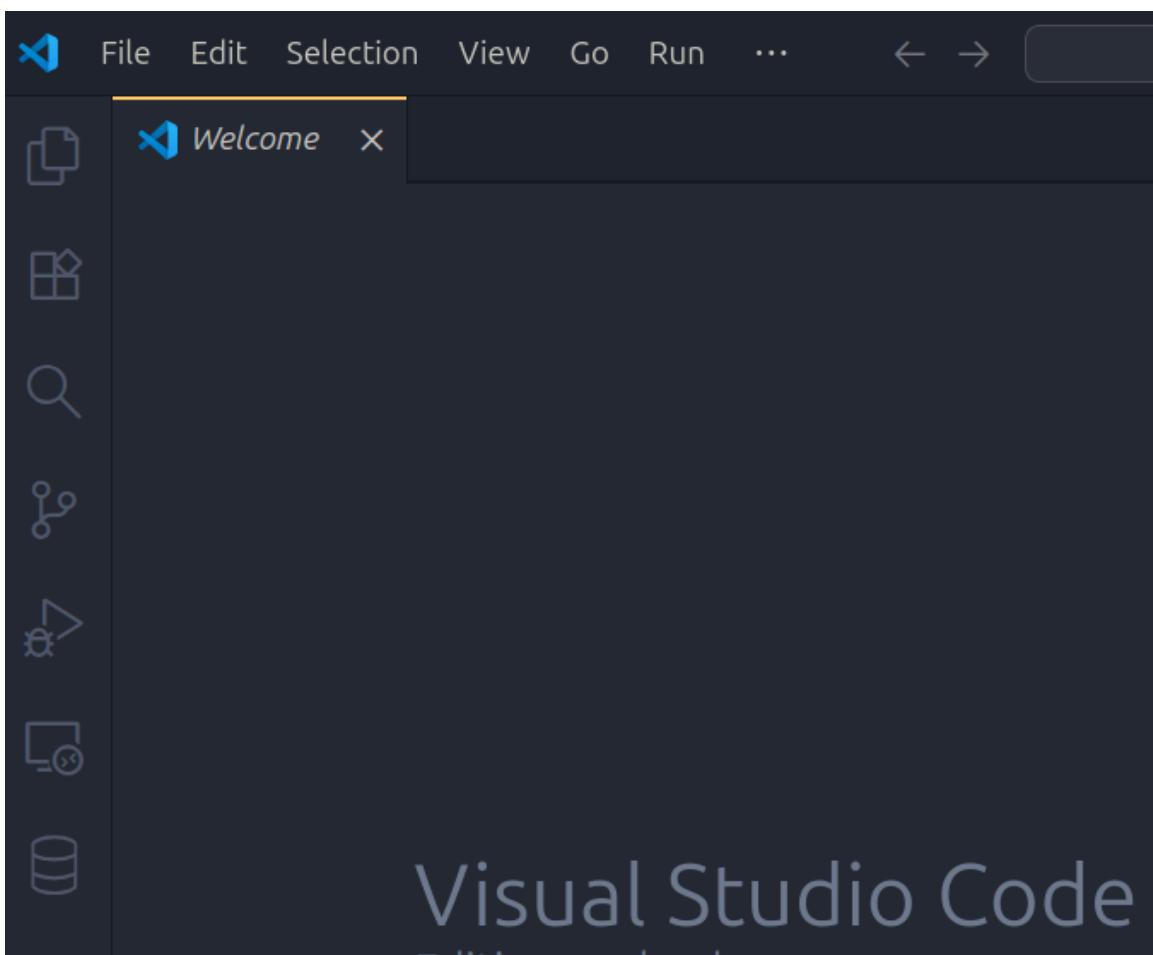
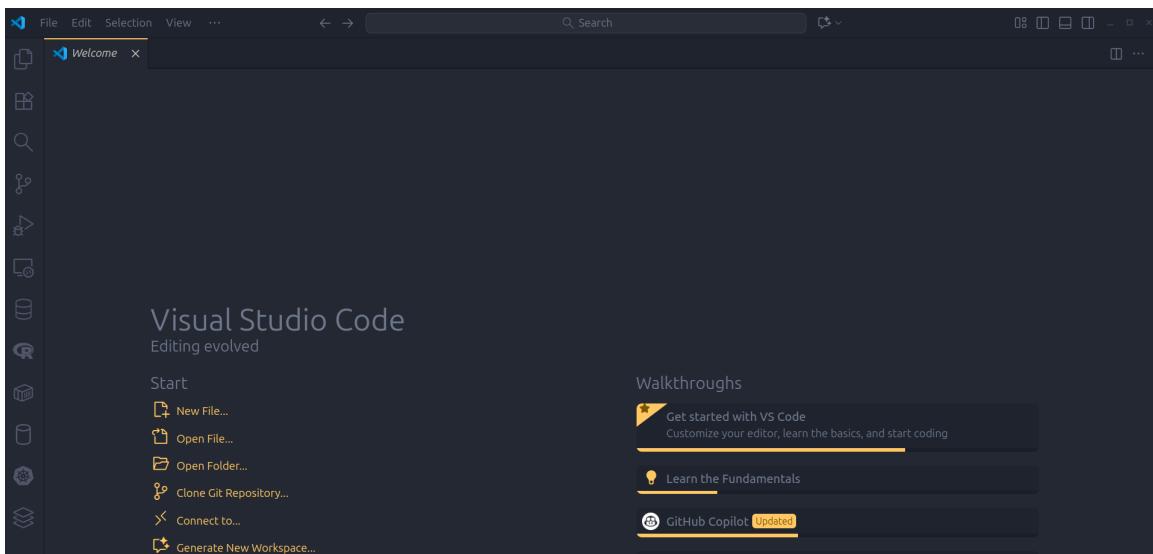
- 
- [] Install GitHub
  - [] Sign up for GitHub Copilot
  - [] Install Python
  - [] Install Visual Studio (VS) Code
  - [] Adjust VSCode to work with Python
  - [] Download Zip file

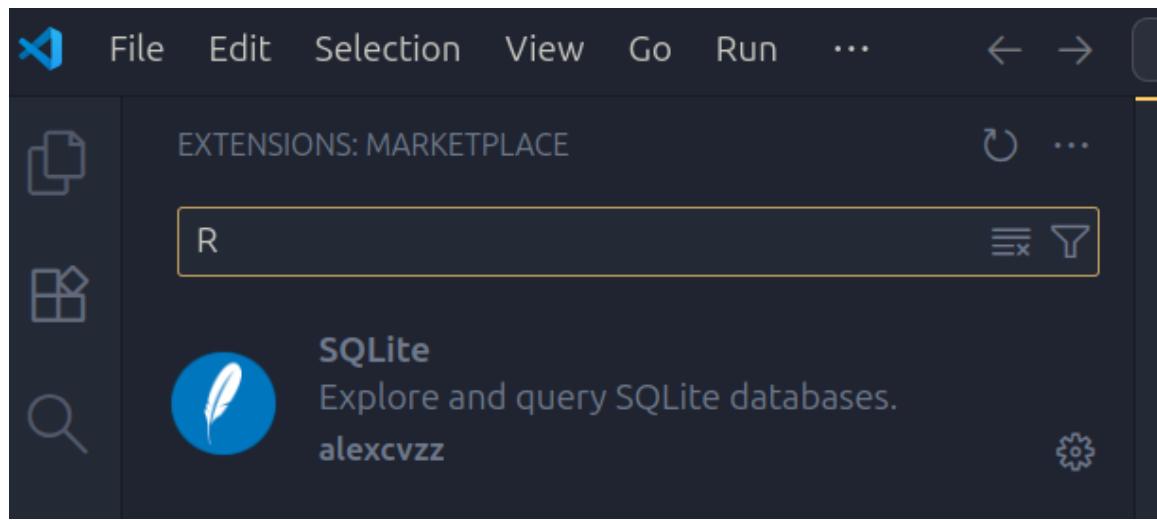
## 6 Setting up Python in VSCode

**Note** Your VSCode will look slightly different, as the screenshots are from a custom VSCode theme. Additionally, the ordering and number of tabs on the sidebar may be different from the VSCode default.

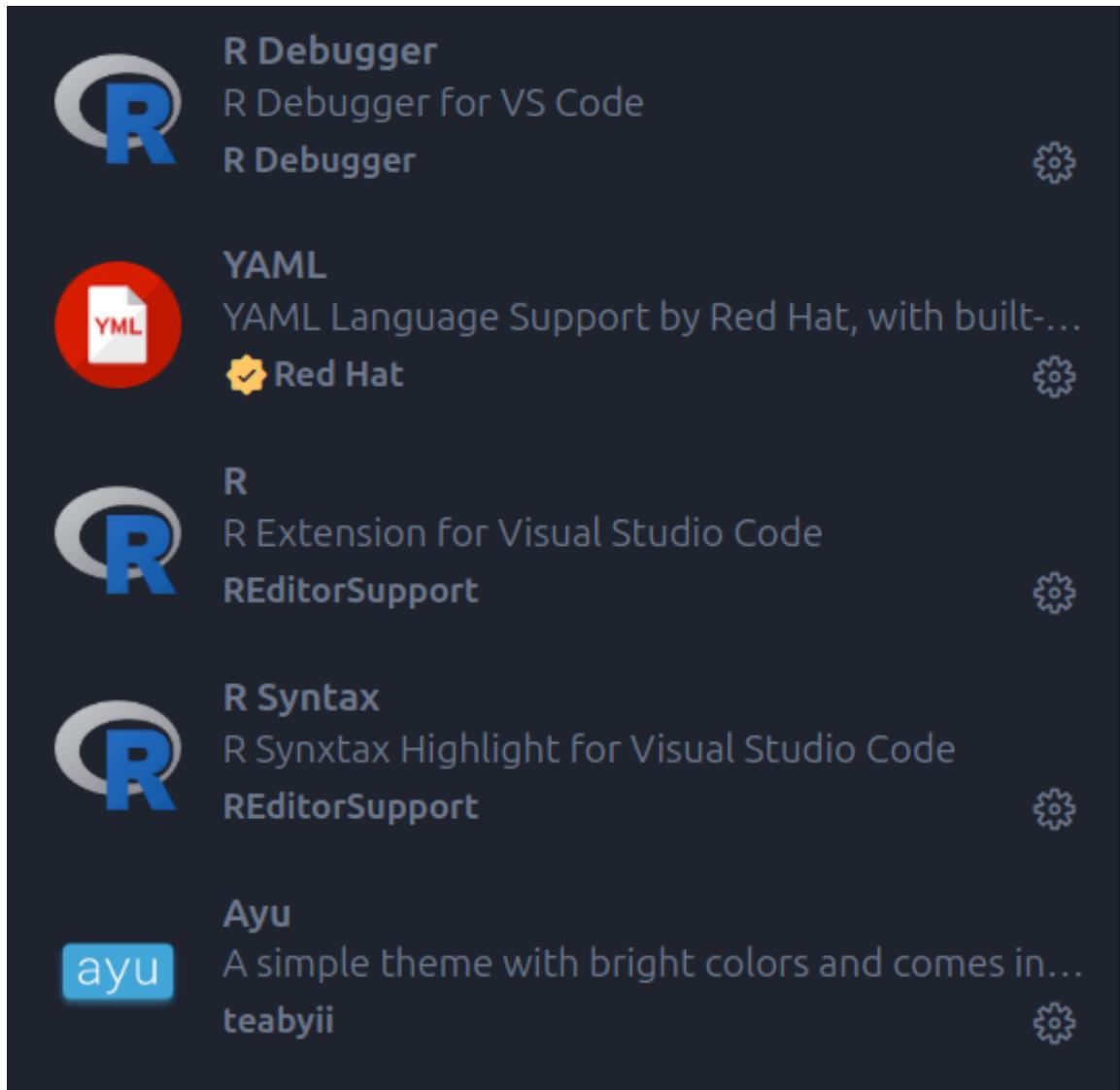
Depending on the version of Python you installed, there may be some issues in incorporating some of the packages. I will include alternatives to try and make sure everything works.

If you have not downloaded Python, then you will want to do that first. Once you have Python, then you can move forward with installing extensions for Python.





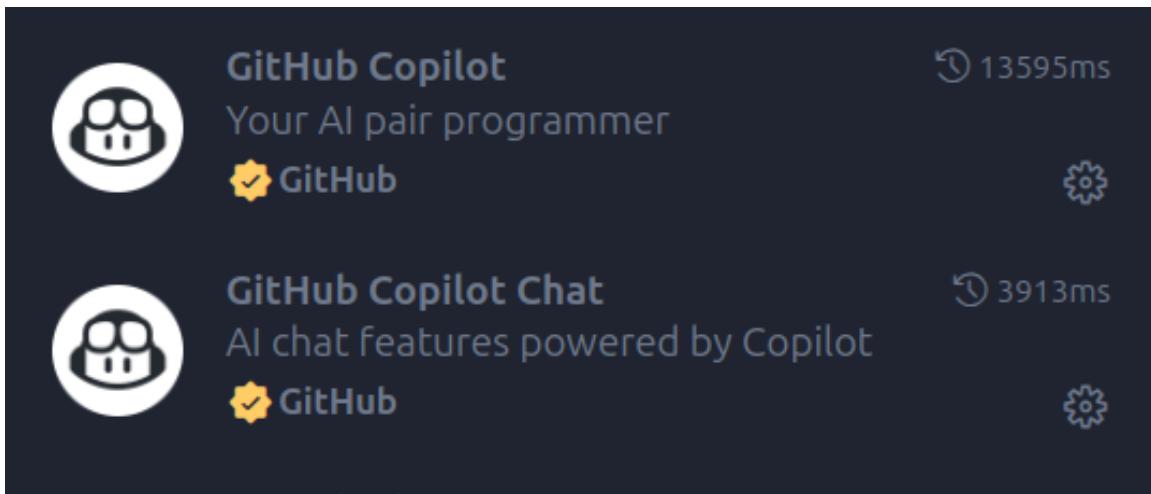
The easiest way of finding all the extensions you will need is to use the search bar. \*You can also use this to find a theme for your VScode. [See several examples of themes here](#) that you can then search for by name.



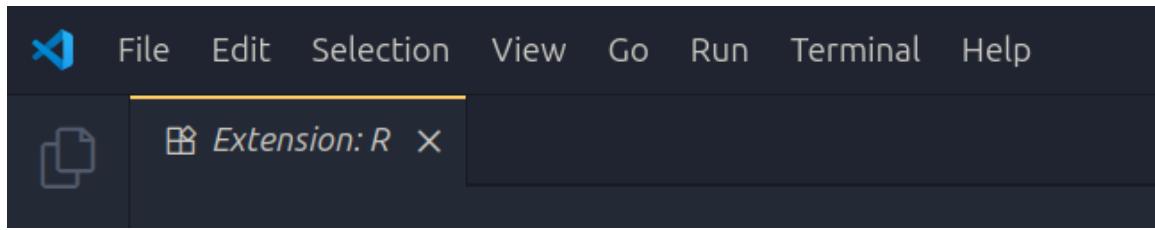
In addition to installing the Python extension, I also have installed the **R Debugger** and the **R Syntax** extensions. These add some additional tools when using Python.

The screenshot shows the R extension for Visual Studio Code in the marketplace. The extension has a rating of 4.5 stars from 43 reviews. It includes features like code analysis, interacting with R terminals, and managing packages. The installation section shows it was published 8 years ago and last released 5 months ago, with a size of 5.4MB. Categories include Programming Languages, Snippets, and Other. Resources link to Repository, Issues, License, and Marketplace.

You can also make sure that your GitHub Copilot is installed. There are two extensions that should be installed, `GitHub Copilot` and the `GitHub Copilot Chat` extension.



From these instructions, you will want to install the `languageServer` and the `httpgd` packages. You can also install the `radian` package if you would like; however, the trade-offs are not much better for the amount of effort to get it working in VSCode.



To install these packages, you can go to the top and click on Terminal followed up by New Terminal. That should open a terminal in bash, like in the screenshot below.

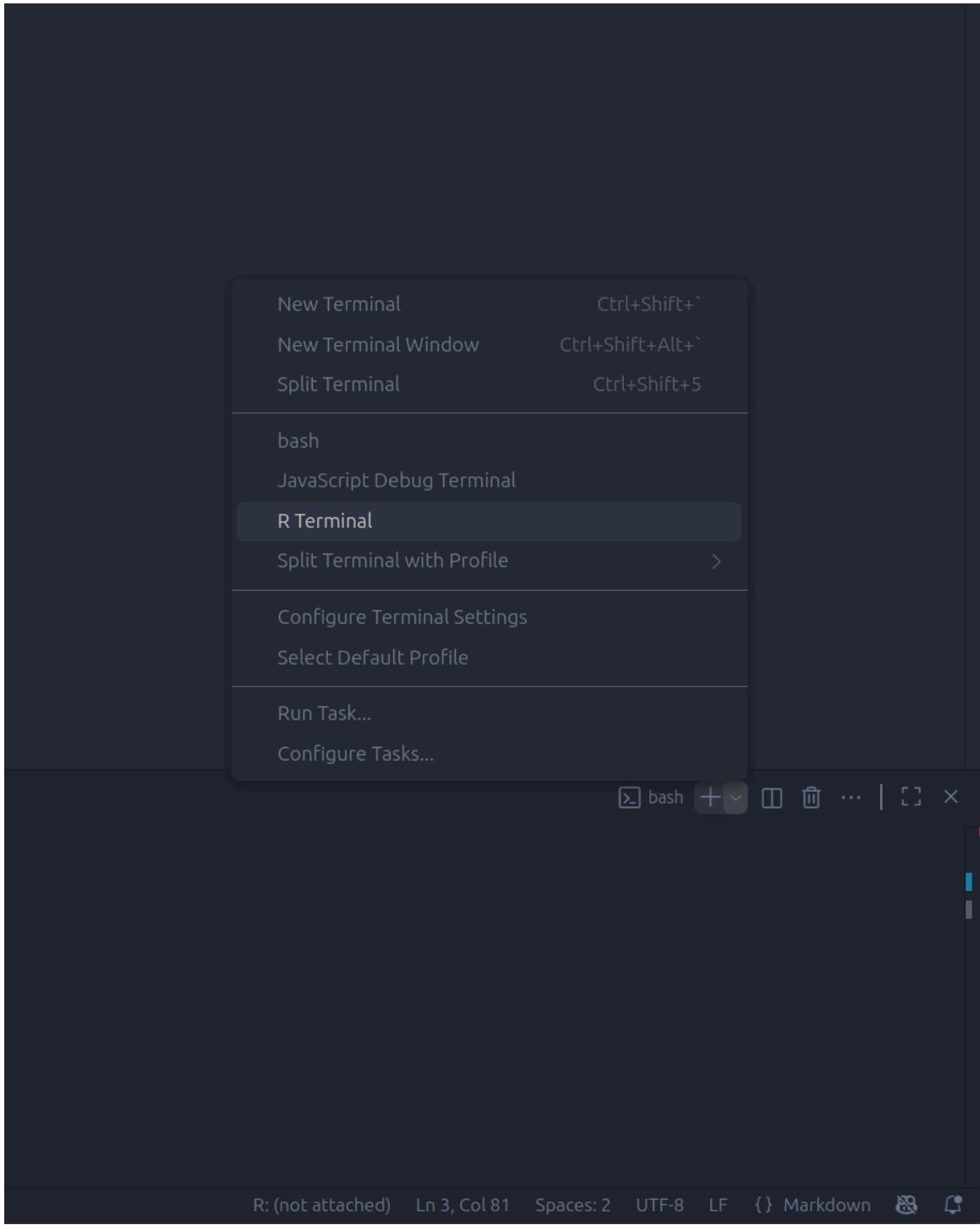
A screenshot of a terminal window. The title bar says "Terminal". The window contains the following text:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

● jp1228@jp1228-Swift-SF314-52:~$ R --version
R version 4.5.2 (2025-10-31) -- "[Not] Part in a Rumble"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under the terms of the
GNU General Public License versions 2 or 3.
For more information about these matters see
https://www.gnu.org/licenses/.

● jp1228@jp1228-Swift-SF314-52:~$ which R
/usr/bin/R
○ jp1228@jp1228-Swift-SF314-52:~$ █
```



PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```
Copyright (c) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
```

```
Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

```
> install.packages("languageserver")
```

```
> install.packages("httpgd")
Installing package into '/home/jp1228/R/x86_64-pc-linux-gnu-library/4.5'
(as 'lib' is unspecified)
Warning message:
package 'httpgd' is not available for this version of R

A version of this package for your version of R might be available elsewhere,
see the ideas at
https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
>
```

```
> install.packages("remotes")
Installing package into '/home/jp1228/R/x86_64-pc-linux-gnu-library/4.5'
(as 'lib' is unspecified)
trying URL 'https://p3m.dev/cran/__linux__/manylinux_2_28/latest/src/contrib/remotes_2.5.0.tar.gz'
Content type 'binary/octet-stream' length 165146 bytes (161 KB)
=====
downloaded 161 KB

* installing *source* package 'remotes' ...
** this is package 'remotes' version '2.5.0'
** package 'remotes' successfully unpacked and MD5 sums checked
** using staged installation
** R
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (remotes)

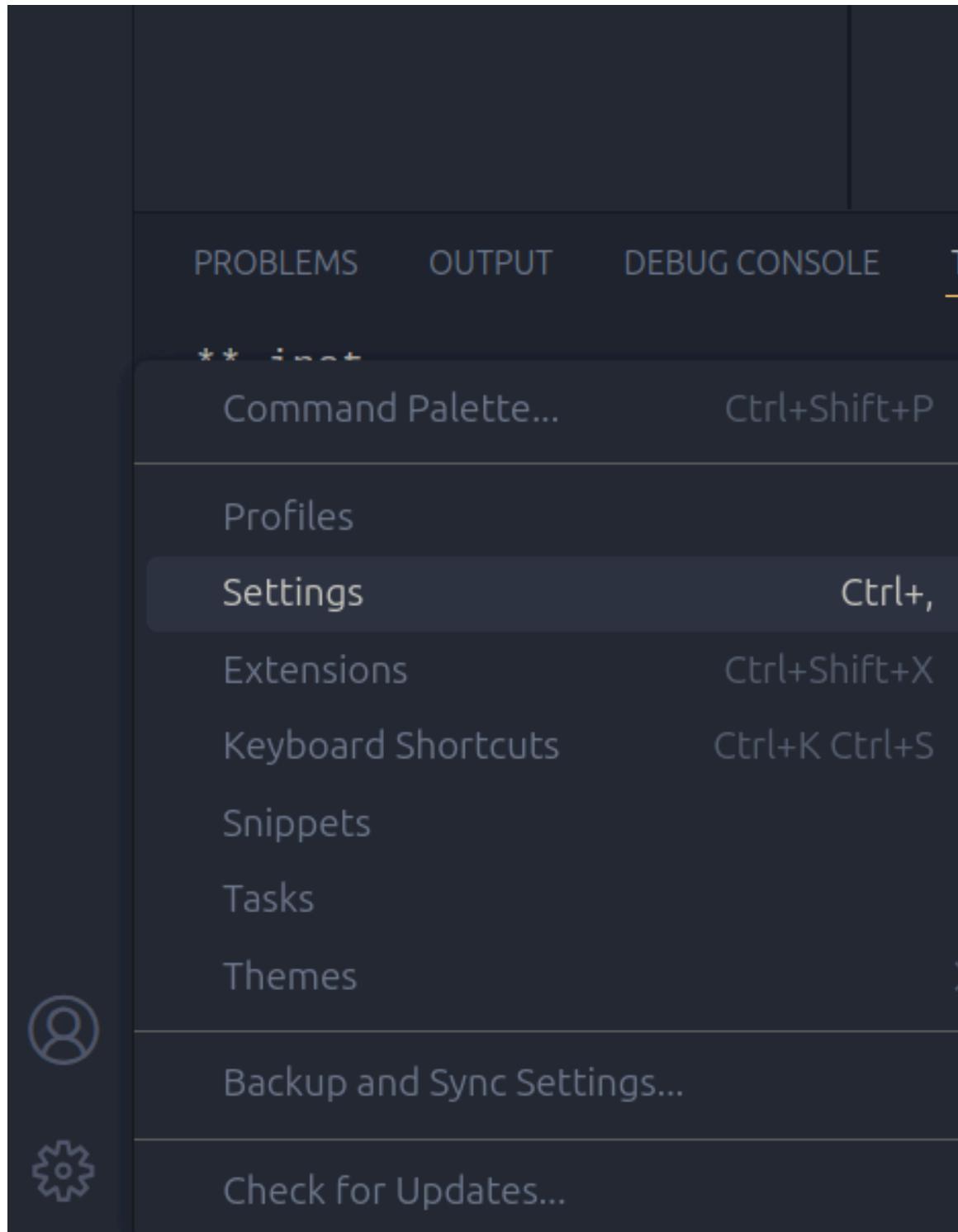
The downloaded source packages are in
  '/tmp/RtmpiuAIbT/downloaded_packages'
> remotes::install_github("nx10/httpgd")
```

You can get the `httpgd` package by first installing the `remotes` package. This will then allow you to install the development version from GitHub. If you want to follow along with the instructions from the developer(s) of the `httpgd` package, you can [follow the installation instructions here](#).

Lastly, I will show some extra customizable settings to make VSCode similar to using RStudio. Below are some settings that you can change in your `settings.json` file. You can also change the settings by going to the gear (see directions below) to make changes to your VSCode.

1. Making Changes by Settings Tab





R.

2217 Settings Found. AI Results Available

User

**Quarto > Render: R Package Output Directory**

Render output files in a temporary directory, when in an R package.

**R: Always Use Active Terminal**

Use active terminal for all commands, rather than creating a new R terminal.

**R: Bracketed Paste**

Use bracketed paste mode when sending code to terminal. Enable for `radian` console.

**R > Help Panel: Cache Index Files**

Whether/where to store parsed help indices between sessions.

None

**R > Help Panel: Click Code Examples**

What happens when clicking code examples on help pages. Might require restarting to take effect.

Item	Value
Click	Copy
Ctrl+Click	Run
Shift+Click	Ignore

**R > Help Panel: Enable Hover Links**

Show links to matching help pages in hover

**R > Help Panel: Enable Syntax Highlighting**

Enable syntax highlighting in the help panel.

**R > Help Panel: Preview Local Packages**

Which local directories to try for local help pages previewer. Set to `[]` to disable.

Add Item

**R: Lib Paths**

Additional library paths to launch R background processes (R languageserver, help server, etc.). These paths will be appended to `.libPaths()` on process startup. It could be useful for projects with `renv` enabled.

Add Item

**R > Live Share > Defaults: Command Forward**

Default boolean value for guest command forwarding.

**R > Live Share > Defaults: Share Browser**

Default boolean value for automatically sharing R browser ports with guests.

**R > Live Share > Defaults: Share Workspace**

Default boolean value for sharing the R workspace with guests.

**R > Plot: Use Httpgd**

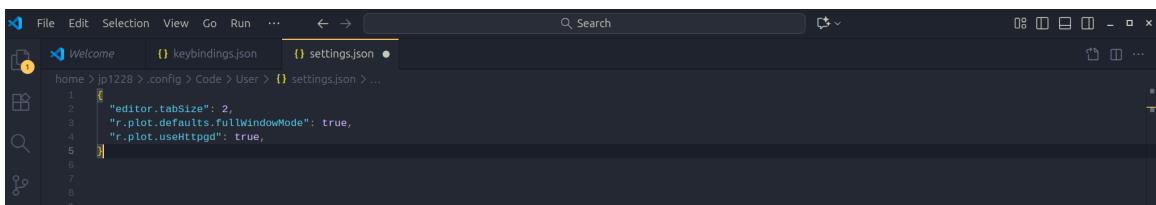
Use the httpgd-based plot viewer instead of the base VSCode-R plot viewer. Changes the option `vsc.use_httpgd` in R.

Requires the `httpgd` R package version 1.2.0 or later.

## 2. Making Changes Using the settings.json File

To get to the `settings.json` file, you will use the following keybinding shortcut (Windows/Linux: Ctrl + Shift + P, Mac: Cmd + Shift + P). This will open the command palette where you can then search for Preferences: Open User Settings (JSON). Here you can then copy and paste the code below to use the `httpgd` package and tab 2 spaces rather than the default 4. If you change your theme, this will also show up here as well as preferences made for other extensions. Once you have made these changes, you can save your settings and close out of the `settings.json` file.

```
{
  "editor.tabSize": 2,
  "r.plot.defaults.fullWindowMode": true,
  "r.plot.useHttpgd": true,
}
```



Another preference is to include shortcuts for some common RStudio shortcuts. The main two are being able to comment in/out code in your Python scripts. Similar to RStudio, you can add the shortcuts below using the `keybindings.json` file.

You will use the keybinding shortcut (Windows/Linux: Ctrl + Shift + P, Mac: Cmd + Shift + P) to get to the command palette again. Here you can search for the Preferences: Open Keyboard Shortcuts (JSON) and copy and paste the code below. **Note** Be aware that if you do not have \$ version 4.1.0 of greater, you will need to change the `{ "text": "|>" }` to `{ "text": "%>%"`. You can then save the file and close out and you should be able to use these shortcuts.

```
[
  {
    "key": "ctrl+shift+c",
    "command": "editor.action.commentLine",
    "when": "editorTextFocus && !editorReadonly"
  },
  {
    "key": "ctrl+shift+m",
    "command": "type",
    "args": { "text": "|>" },
    "when": "editorTextFocus && editorLangId == 'r'"
  }
]
```

To make changes to any other shortcuts, you can use the command palette to search Keyboard Shortcuts to change any other shortcuts. **Note** Be aware that you could possibly overwrite other

important VSCode functions so be cautious when making changes. You can type in the shortcut you want to create in the search bar at the top to see what current functions use that shortcut.

Command	Keybinding	When	Source
Accept Inline Completion	<code>Ctrl + /</code>	<code>accessibleViewIsShown &amp;&amp; accessibleViewCurrentProviderId == 'inl...</code>	System
Accept Inline Suggestion	<code>Tab</code>	<code>inlineEditIsVisible &amp;&amp; tabShouldAcceptInlineEdit &amp;&amp; !editorHover...</code>	System
Accept Inline Suggestion	<code>Tab</code>	<code>inInlineEditsPreviewEditor</code>	System
Accept Next Word Of Inline Suggestion	<code>Ctrl + RightArrow</code>	<code>cursorBeforeGhostText &amp;&amp; inlineSuggestionVisible &amp;&amp; !accessibl...</code>	System
Accessible Diff Viewer: Go to Next Difference	<code>F7</code>	<code>isInDiffEditor</code>	System
Accessible Diff Viewer: Go to Previous Difference	<code>Shift + F7</code>	<code>isInDiffEditor</code>	System
Add Cursor Above	<code>Ctrl + Shift + UpArrow</code>	<code>editorTextFocus</code>	System
Add Cursor Above	<code>Shift + Alt + UpArrow</code>	<code>editorTextFocus</code>	System
Add Cursor Below	<code>Ctrl + Shift + DownArrow</code>	<code>editorTextFocus</code>	System
Add Cursor Below	<code>Shift + Alt + DownArrow</code>	<code>editorTextFocus</code>	System
Add Cursors to Line Ends	<code>Shift + Alt + I</code>	<code>editorTextFocus</code>	System
Add Line Comment	<code>Ctrl + K Ctrl + C</code>	<code>editorTextFocus &amp;&amp; !editor_READONLY</code>	System
Add Selection to Next Find Match	<code>Ctrl + D</code>	<code>editorFocus</code>	System
Auto Fix...	<code>Shift + Alt + &lt;</code>	<code>textInputFocus &amp;&amp; !editor_READONLY &amp;&amp; supportedCodeAction =~ /(\s...</code>	System
Auto Fix...	<code>Shift + Alt + .</code>	<code>textInputFocus &amp;&amp; !editor_READONLY &amp;&amp; supportedCodeAction =~ /(\s...</code>	System
C/C++: Switch Header/Source	<code>Alt + O</code>	<code>editorTextFocus &amp;&amp; editorLangId =~ /^(c cuda-)?cpp\$/ &amp;&amp; !conf...</code>	C/C++
Calls: Show Call Hierarchy	<code>Shift + Alt + H</code>	<code>editorHasCallHierarchyProvider</code>	Reference Search View
Cancel Selection Anchor	<code>Escape</code>	<code>editorTextFocus &amp;&amp; selectionAnchorSet</code>	System
Change All Occurrences	<code>Ctrl + F2</code>	<code>editorTextFocus &amp;&amp; !editor_READONLY</code>	System
Change Language Mode	<code>Ctrl + K M</code>	<code>!notebookEditorFocused</code>	System
Chat: Apply in Editor	<code>Ctrl + Enter</code>	<code>accessibleViewInCodeBlock &amp;&amp; chatIsEnabled    chatIsEnabled &amp;&amp; i...</code>	System
Chat: Attach Instructions...	<code>Ctrl + Alt + /</code>	<code>chatIsEnabled &amp;&amp; config.chat.promptFiles</code>	System
Chat: Debug Last Terminal Command	<code>Ctrl + Alt + .</code>	<code>github.copilot-chat.activated &amp;&amp; terminalFocus &amp;&amp; terminalShellI...</code>	GitHub Copilot Chat
Chat: Focus Chat Confirmation	<code>Ctrl + Shift + A</code>	<code>accessibilityModeEnabled &amp;&amp; chatIsEnabled</code>	System
Chat: Go to Next Chat Edit	<code>Alt + F5</code>	<code>chatEdits.hasEditorModifications &amp;&amp; chatIsEnabled &amp;&amp; editorFocus...</code>	System
Chat: Go to Previous Chat Edit	<code>Shift + Alt + F5</code>	<code>chatEdits.hasEditorModifications &amp;&amp; chatIsEnabled &amp;&amp; editorFocus...</code>	System
Chat: Insert At Cursor	<code>Ctrl + Enter</code>	<code>accessibleViewInCodeBlock &amp;&amp; chatIsEnabled    chatIsEnabled &amp;&amp; i...</code>	System
Chat: Insert into Terminal	<code>Ctrl + Alt + Enter</code>	<code>accessibleViewInCodeBlock &amp;&amp; chatIsEnabled    chatIsEnabled &amp;&amp; i...</code>	System
Chat: Keep All Chat Edits	<code>Ctrl + Alt + Y</code>	<code>chatEdits.hasEditorModifications &amp;&amp; editorFocus &amp;&amp; !chatEdits.is...</code>	System
Chat: Keep Chat Edits	<code>Ctrl + Shift + Y</code>	<code>chatEdits.hasEditorModifications &amp;&amp; editorFocus &amp;&amp; !chatEdits.is...</code>	System
Chat: Keep this Change	<code>Ctrl + Y</code>	<code>chatEdits.hasEditorModifications &amp;&amp; editorFocus &amp;&amp; !chatEdits.is...</code>	System
Chat: New Chat	<code>Ctrl + L</code>	<code>chatIsEnabled &amp;&amp; inChat</code>	System
Chat: New Chat	<code>Ctrl + N</code>	<code>chatIsEnabled &amp;&amp; inChat</code>	System
Chat: New Chat Editor	<code>Ctrl + N</code>	<code>chatIsEnabled &amp;&amp; inChat &amp;&amp; inChatEditor</code>	System
Chat: Next Code Block	<code>Ctrl + Alt + PageDown</code>	<code>chatIsEnabled &amp;&amp; inChat</code>	System

- 
- [] Install GitHub
  - [] Sign up for GitHub Copilot
  - [] Install Python
  - [] Install Visual Studio (VS) Code
  - [] Adjust VSCode to work with Python
  - [] Download Zip file

## 7 Download Zip File

To get all the materials for the AI-Assisted-Coding-In-Python workshop, you will want to download the [Zip file here](#).

**Note** You may want to wait until the day of your workshop to make sure you download the most recent version of the contents.

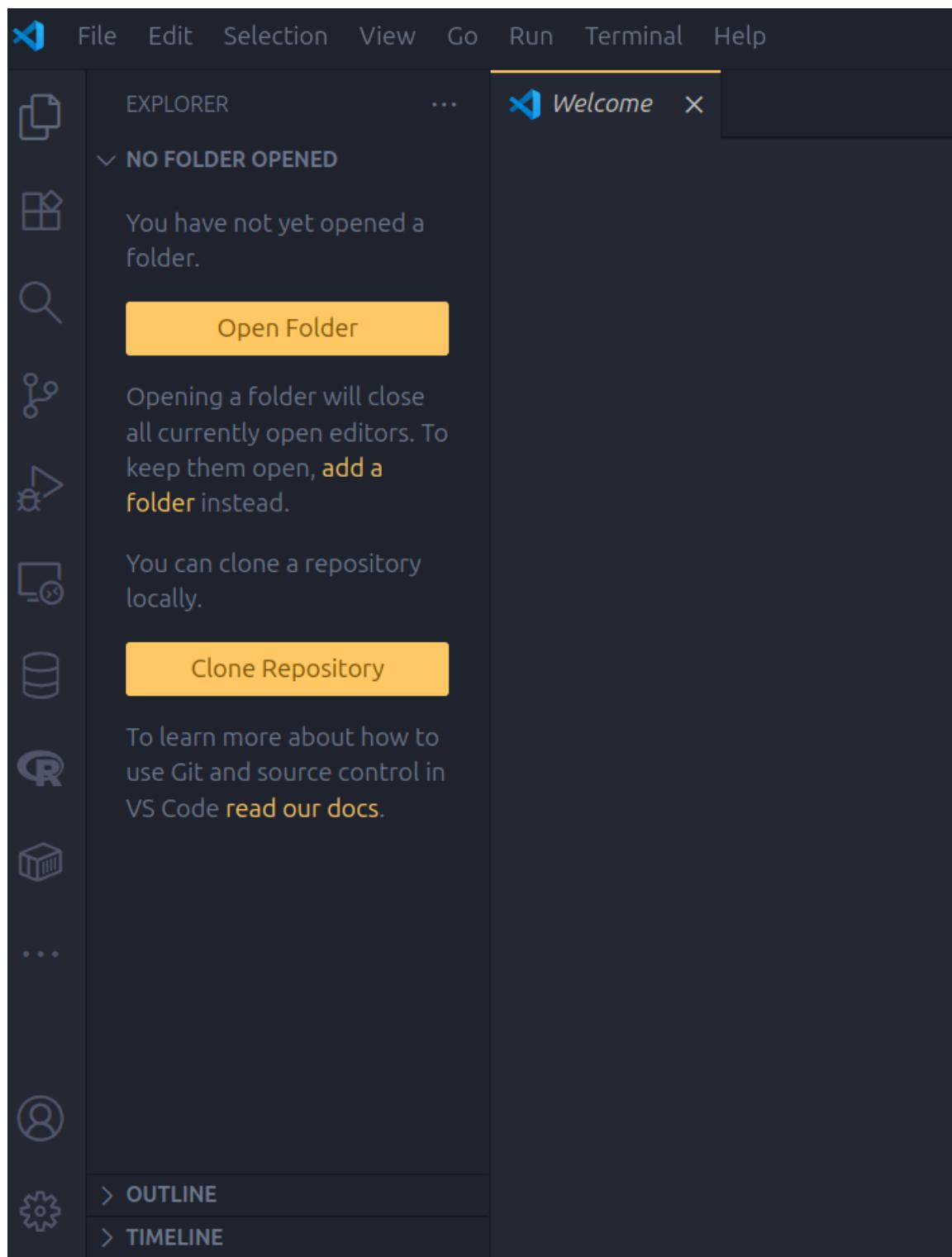
Once you are at the D-Lab GitHub repository for the AI-Assisted-Coding-In-Python workshop, you will go to the green button that says <Code> with a dropdown menu. Clicking on the button will give you options for how to put the repository's contents on your local computer. For now, you can Click on the Download ZIP to download a ZIP file. The ZIP file should then be in your Downloads folder. There you can extract the contents of the ZIP file.

Name	Size	Type
Desktop	8 items	Folder
Documents	13 items	Folder
Downloads	310 items	Folder
<b>example_vscode</b>	7 items	Folder
Music	0 items	Folder
Pictures	1 item	Folder

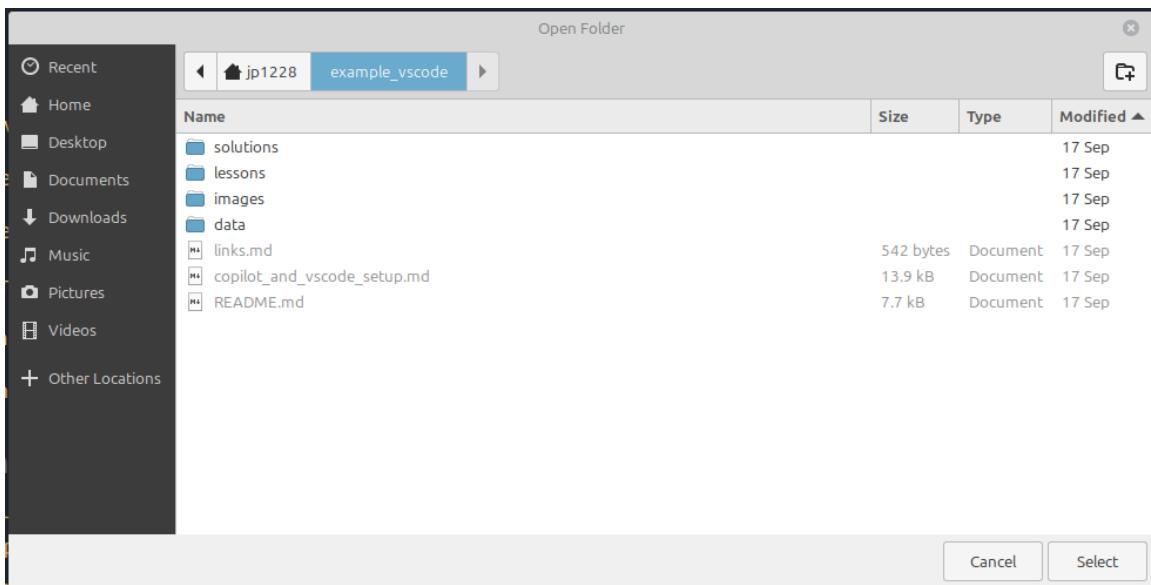
Once you have extracted the contents of your ZIP file. I put all of the contents in a new folder. You can put this folder wherever it makes sense to you. I have my folder (which I named example\_vscode) under my username folder. You can create the folder on your Desktop for easy access or within your Documents folder if you would like.

Name	Size	Type
data	2 items	Folder
images	6 items	Folder
lessons	1 item	Folder
solutions	1 item	Folder
<b>copilot_and_vscode_setup.md</b>	13.9 kB	Document
<b>links.md</b>	542 bytes	Document
<b>README.md</b>	7.7 kB	Document

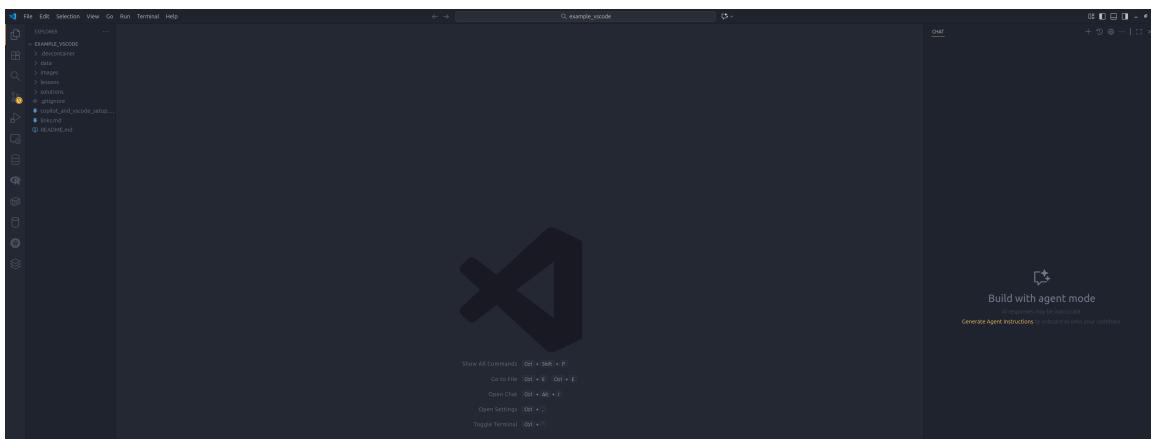
Within the folder, I copied and pasted everything from the ZIP file into this folder. The files should look similar to the screenshot above.



Now you can open VSCode and click on the folder tab on the left sidebar. There you can click on the option Open Folder to open the folder with the workshop contents.



In my case, I will look for my `example_vscode` folder and click select at the bottom to start VSCode from this folder.



Once you select your folder, your VSCode will populate with your workshop files on the left and a tab on the right for your prompts with GitHub Copilot.

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree with 'EXAMPLE\_VSCODE' expanded, showing '.devcontainer', 'data', 'images', 'lessons', and 'workshop.Rmd'. The 'workshop.Rmd' file is selected. The main editor area shows an R Markdown file named 'workshop.Rmd' with the following content:

```

1 #> title: "D-Lab GitHub Copilot Assisted Coding Workshop"
2 #> output: html_notebook
3 ...
4 ...
5 ...
6 ...
7 ...
8 ...
9 ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
23 ...
24 ...
25 ...
26 ...
27 ...
28 ...
29 ...
30 ...
31 ...
32 ...
33 ...
34 ...
35 ...
36 ...
37 ...
38 ...
39 ...
40 ...
41 ...
42 ...
43 ...
44 ...
45 ...
46 ...
47 ...
48 ...
49 ...
50 ...
51 ...
52 ...
53 ...
54 ...
55 ...

```

The right side of the interface has a 'CHAT' bar and a 'Build with agent mode' sidebar. The sidebar includes a note that AI responses may be inaccurate and a link to 'Generate Agent Instructions'.

Finally, you can click on the **lessons** folder on the left and click on your notebook.

Congrats! You are now ready for your AI-Assisted-Coding-In-Python workshop.

- 
- Install GitHub
  - Sign up for GitHub Copilot
  - Install Python
  - Install Visual Studio (VS) Code
  - Adjust VSCode to work with Python
  - Download Zip file