Web Based Dashboards

Module # 1 (Seting up IDE and Libraries)

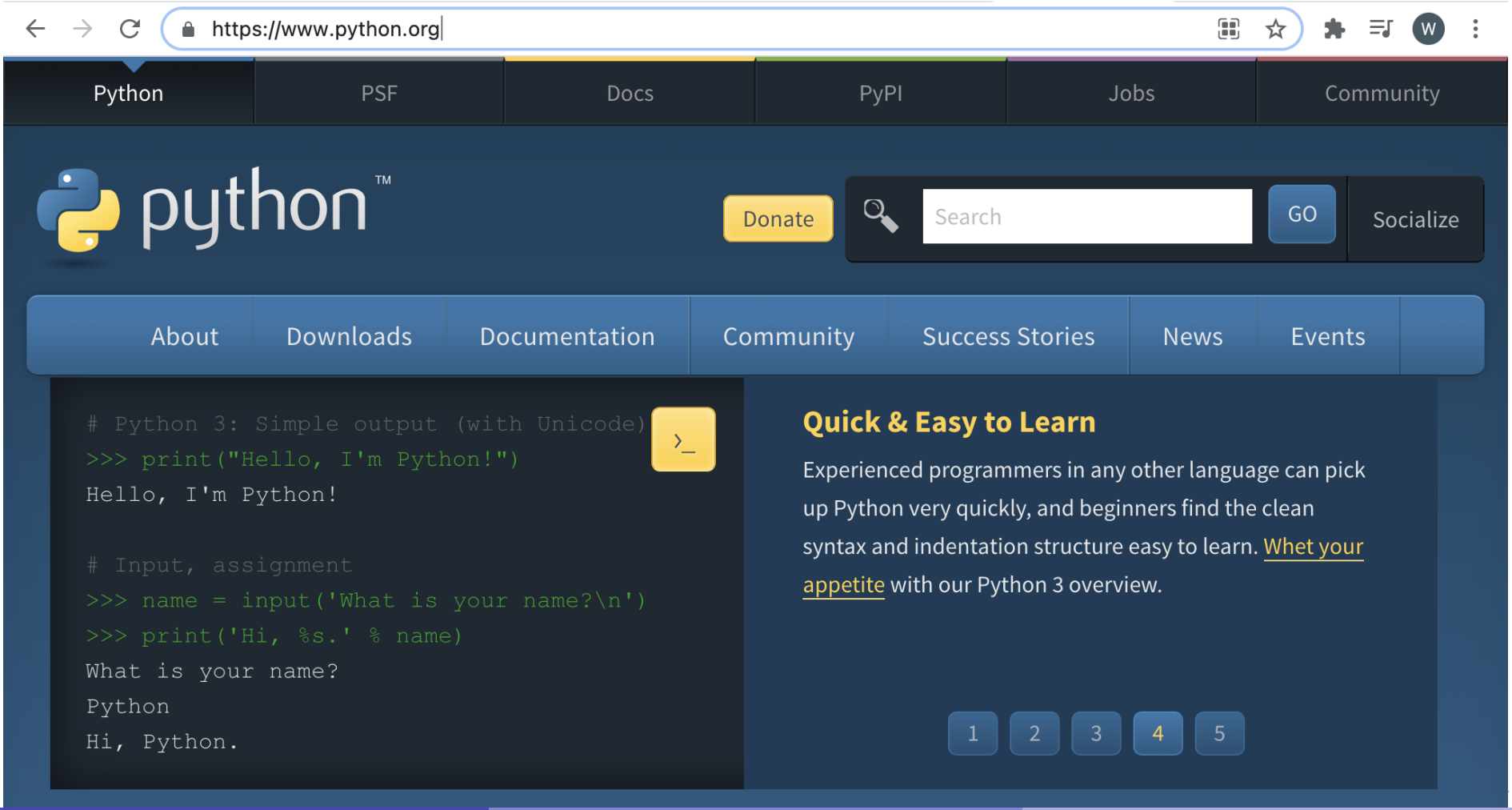
Dr. Wajahat Gilani

Rutgers Business School

June 8, 2021

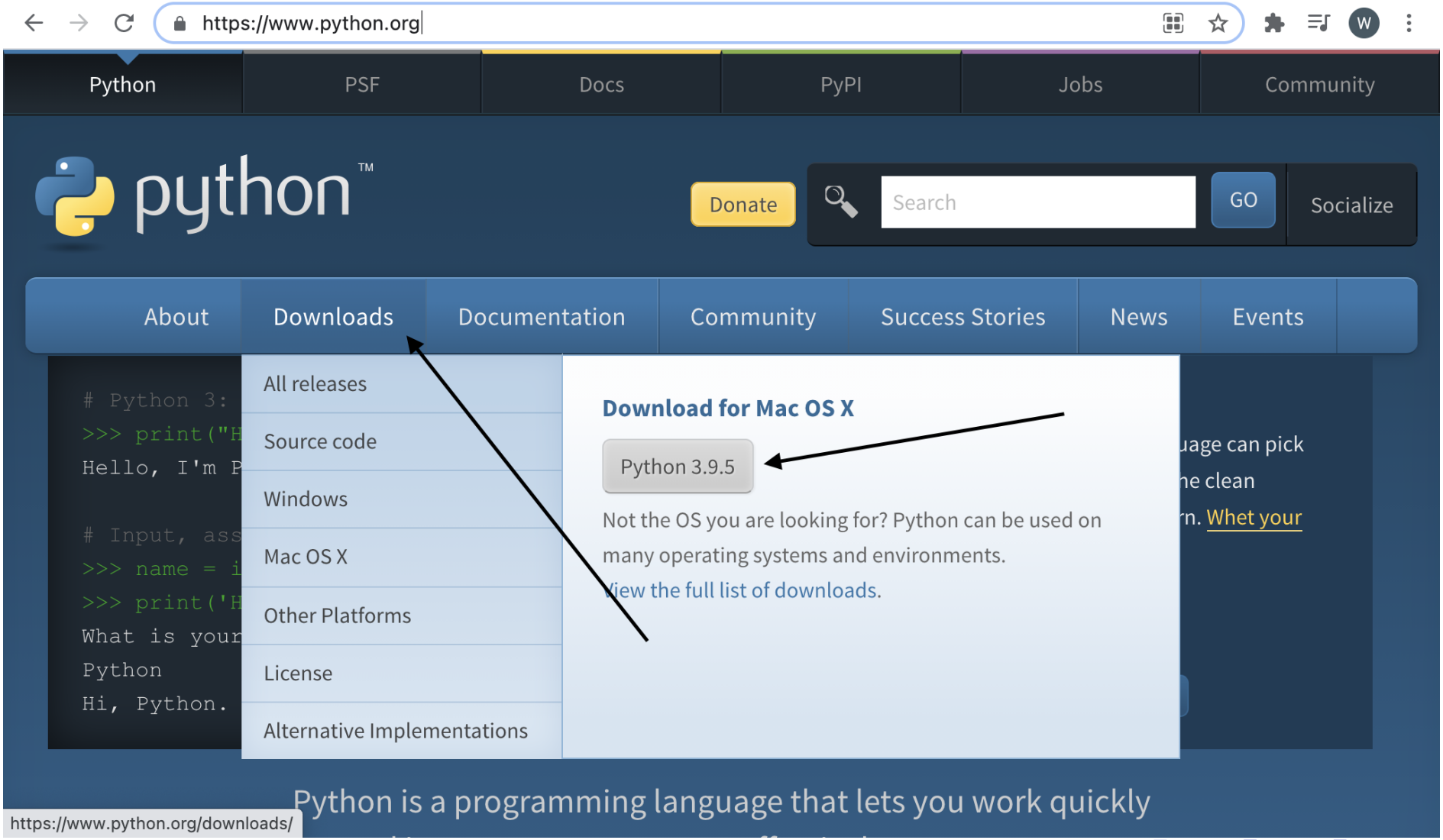
Step # 1: Download Latest Python

Even though we already have many IDE's in the Anaconda package, it is better to use lightweight editor atom to build our web applcation. For that reason it is also better to download the latest python package. Go to https://www.python.org/:



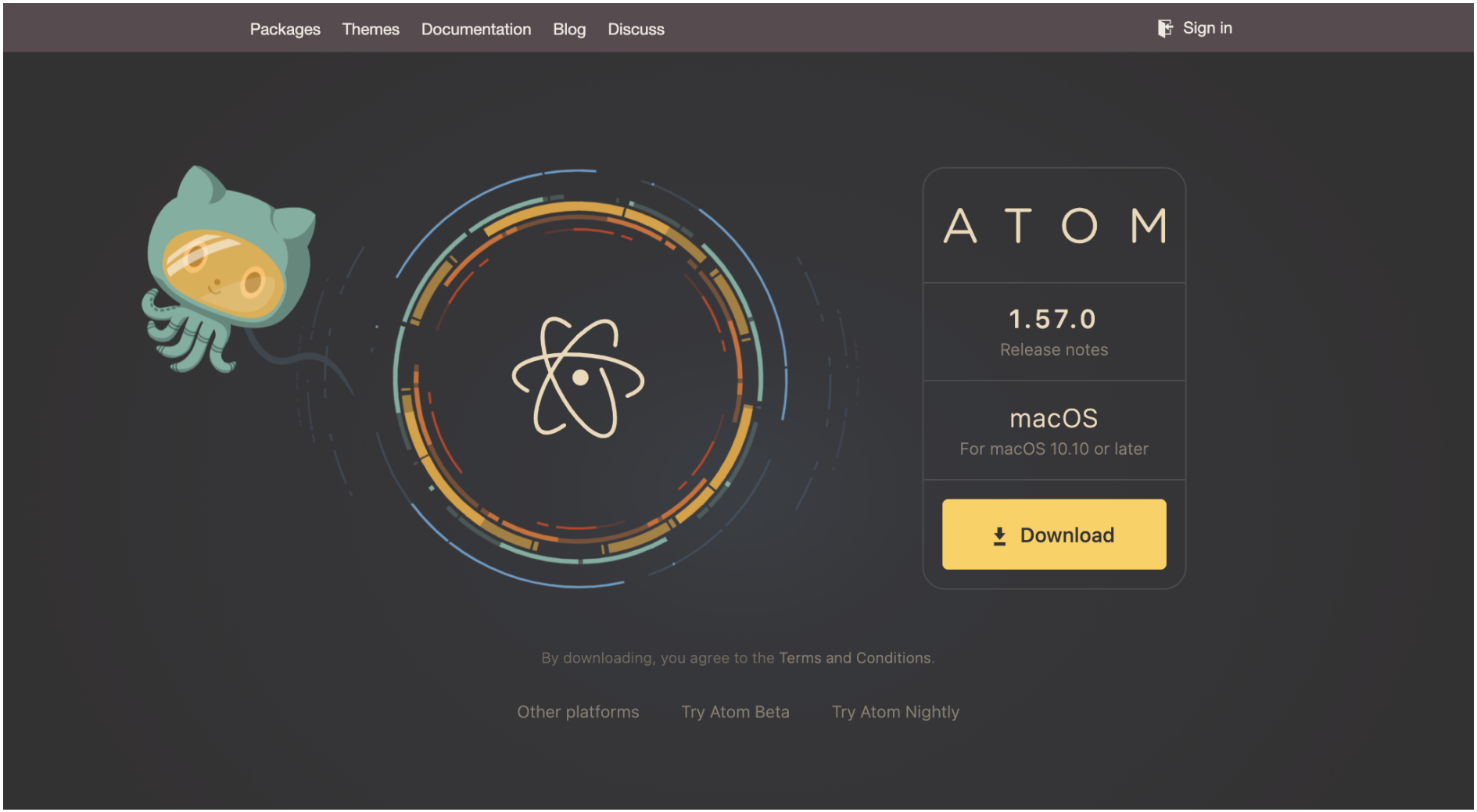
Step # 1: Download Latest Python

Click on downloads, and download the latest python version for your computer:



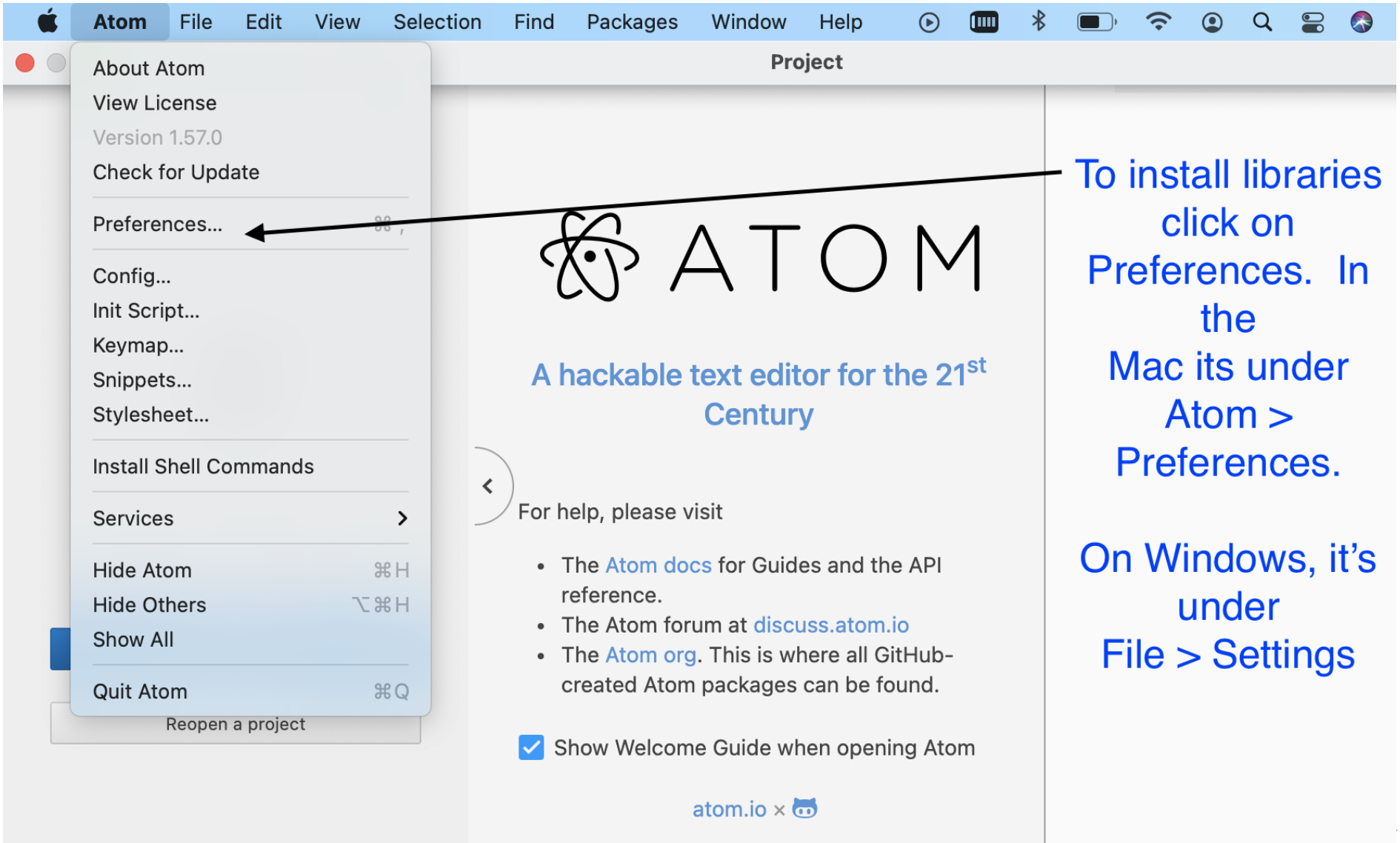
Step # 2: Download Atom IDE

Go to https://atom.io/ and download that atom IDE.



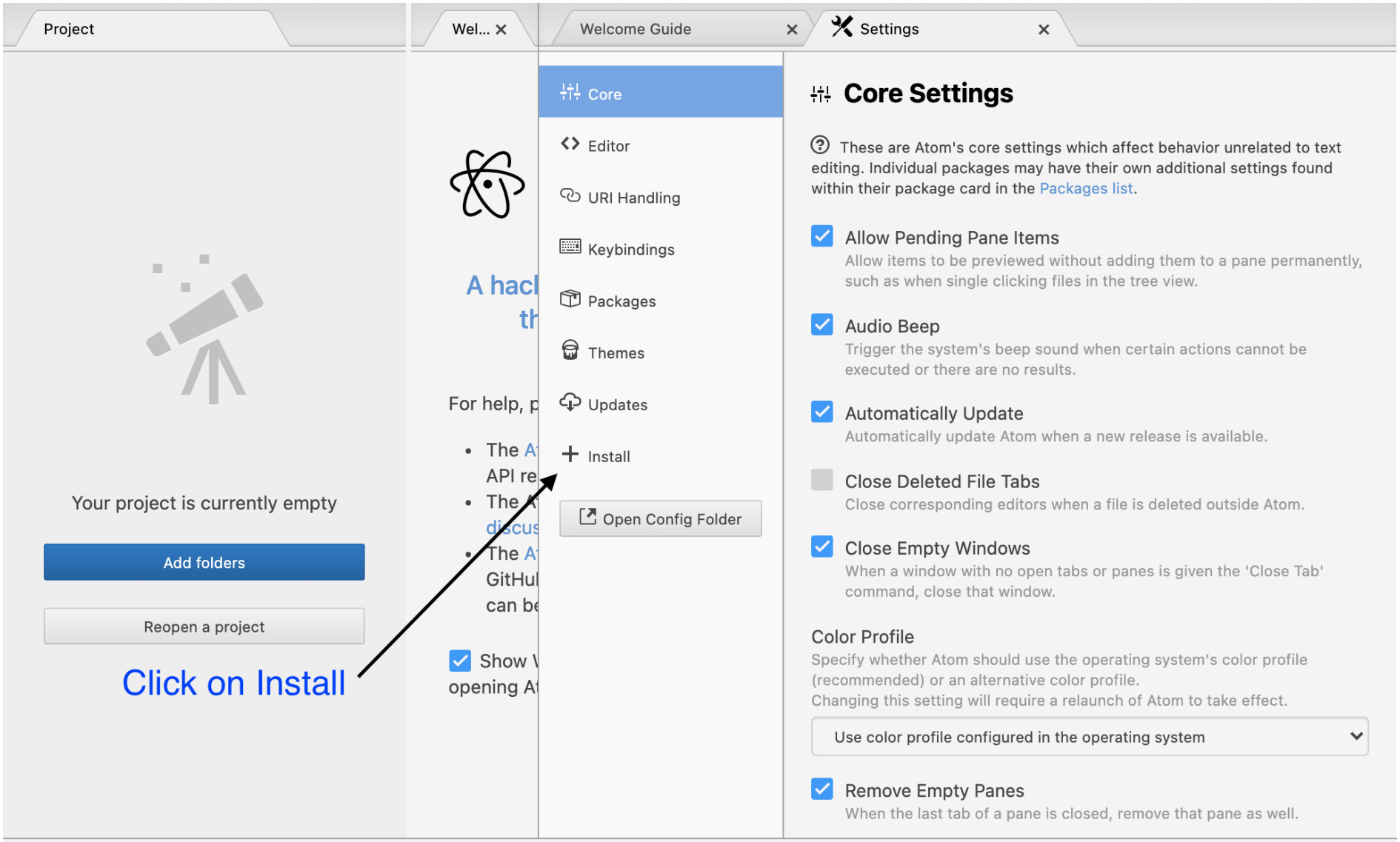
Step # 3: Download Atom Libraries from Preferences (Settings)

We will add packages to atom to enhance it:



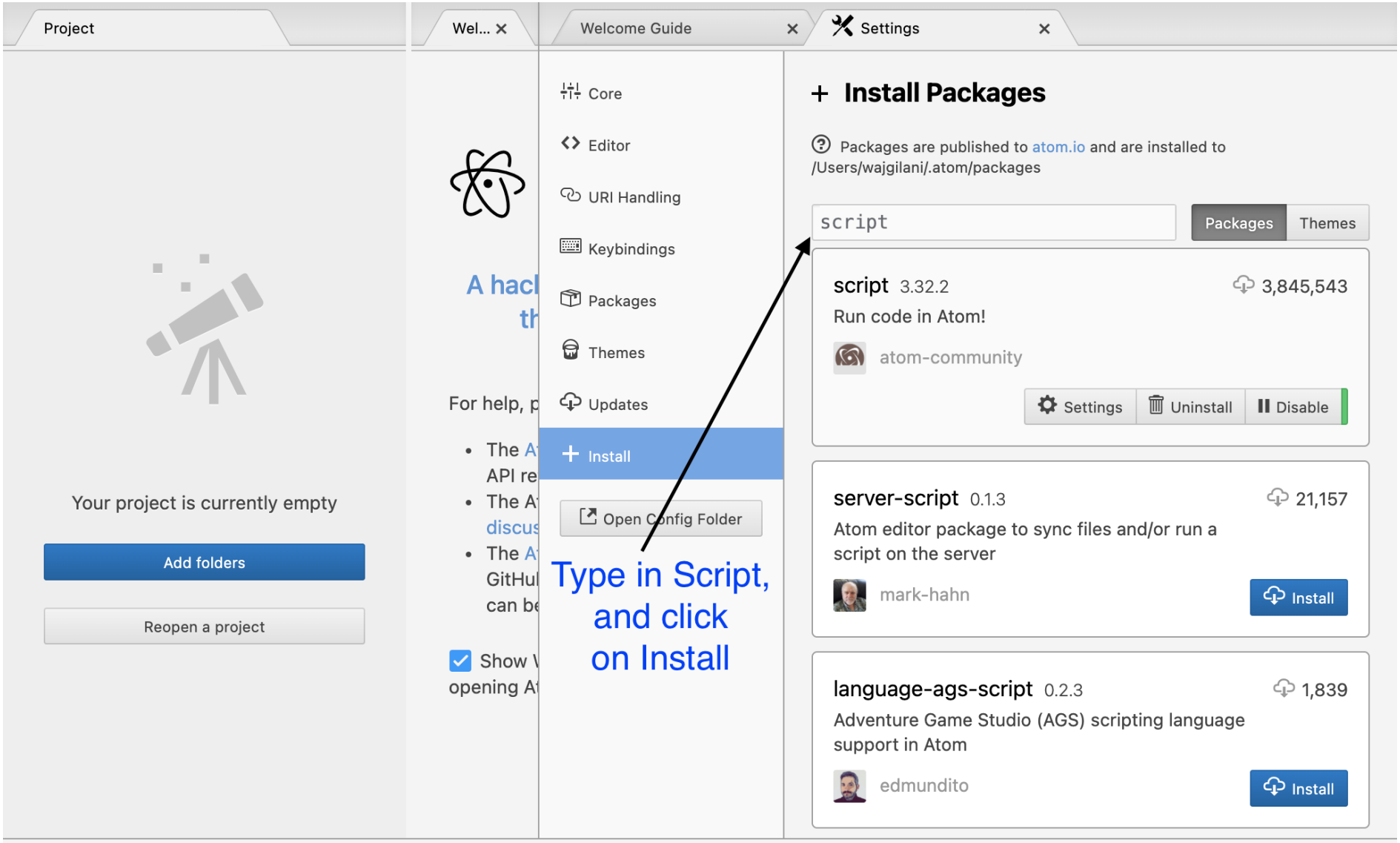
Step # 3: Download Atom Libraries from Preferences (Settings)

Click on Install:



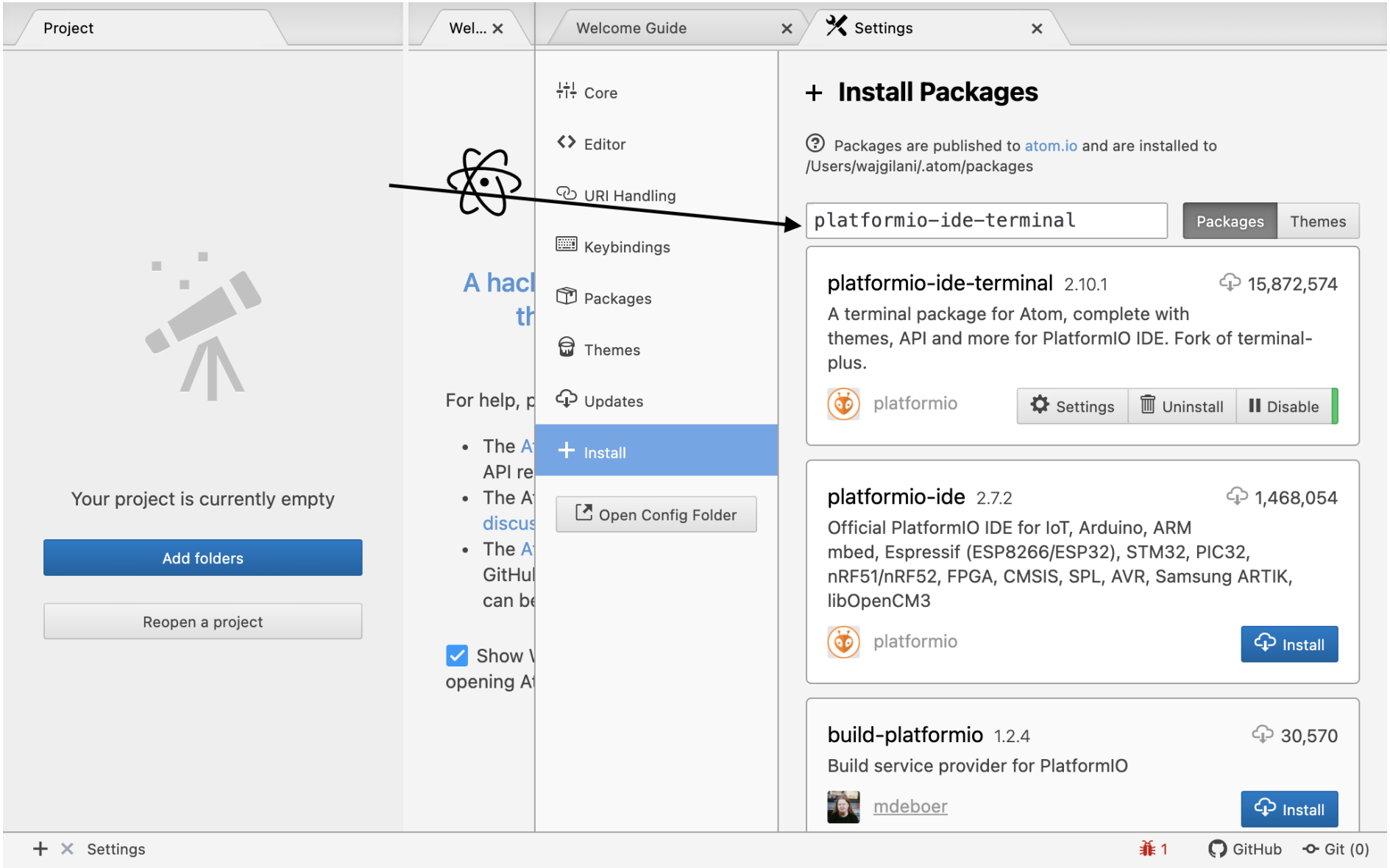
Step # 3: Install Script

My image doesn't have an Install option because it is already installed on my machine. (Notice how many downloads there were so far)



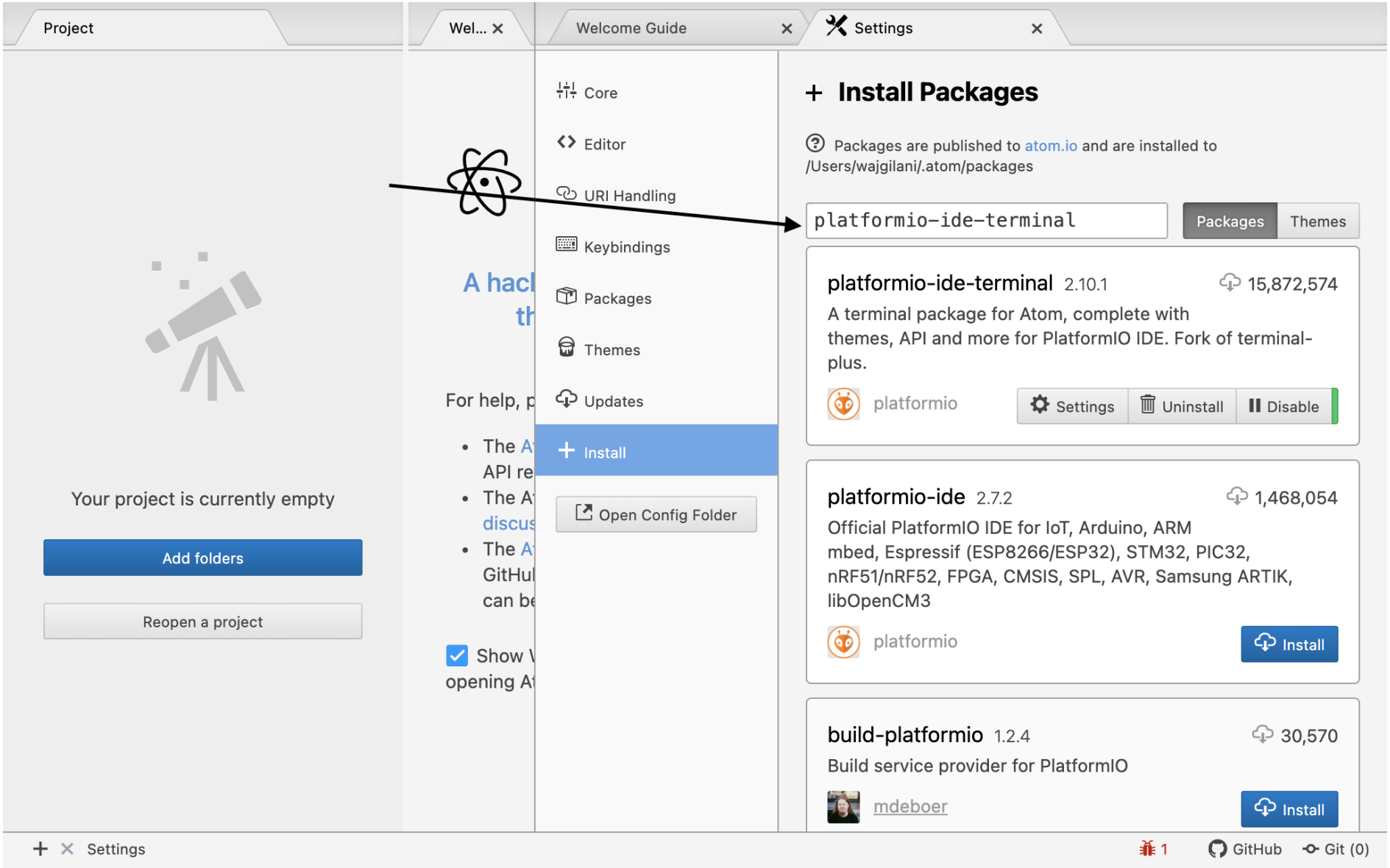
Step # 4: Install platformio-ide-terminal

This allows you to open a Terminal (command prompt) from atom.



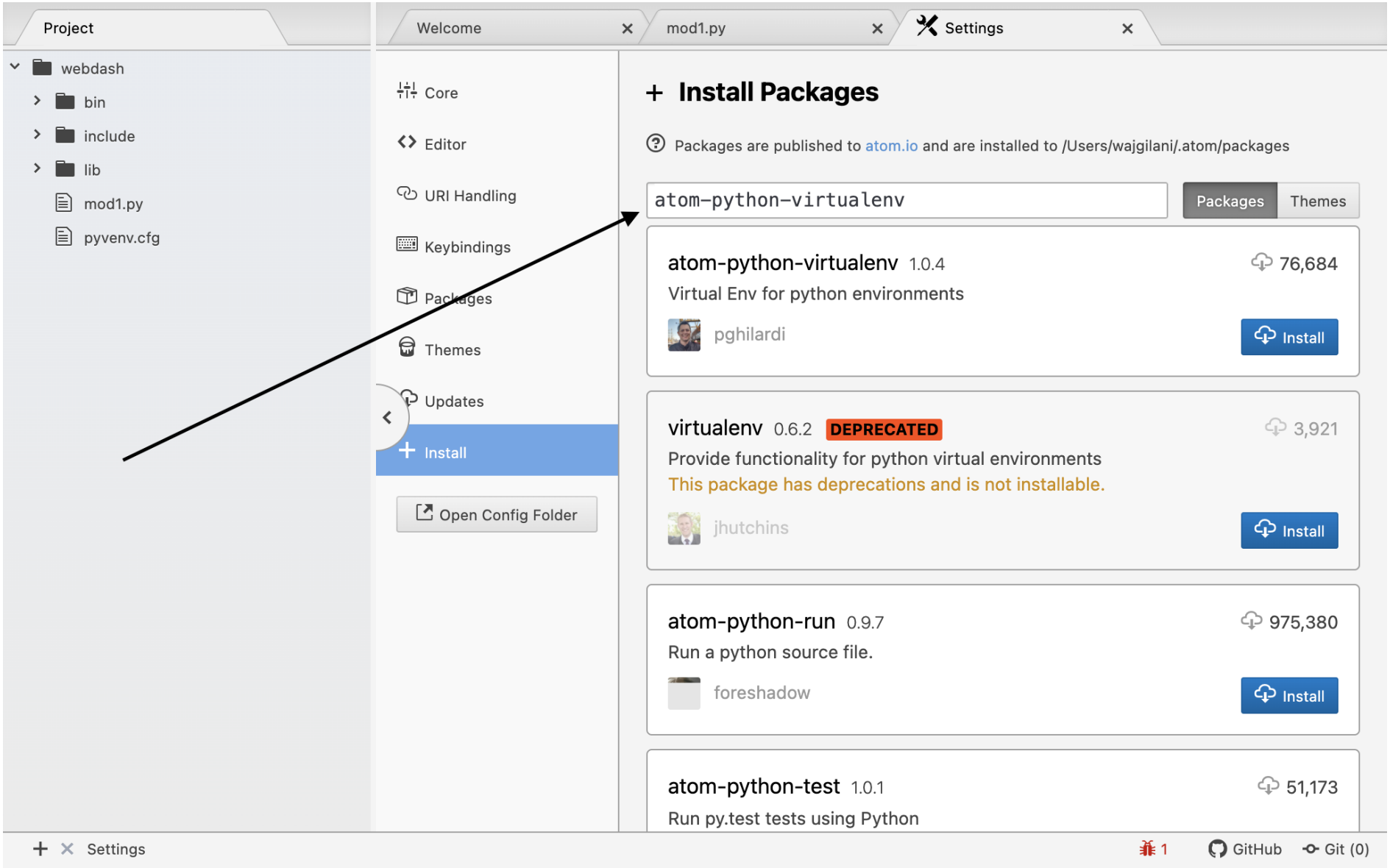
Step # 4: Install platformio-ide-terminal

This allows you to open a Terminal (command prompt) from atom.

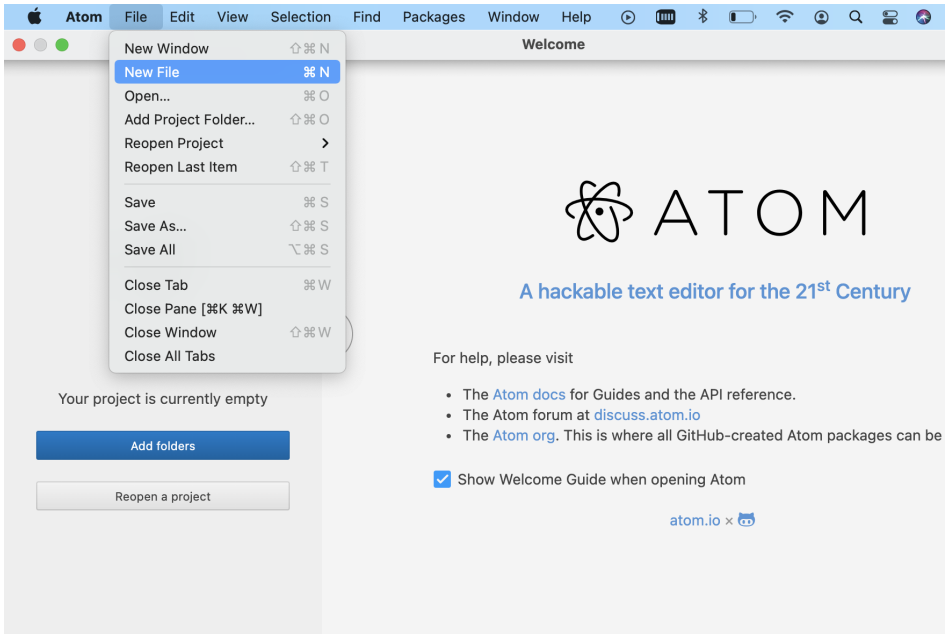


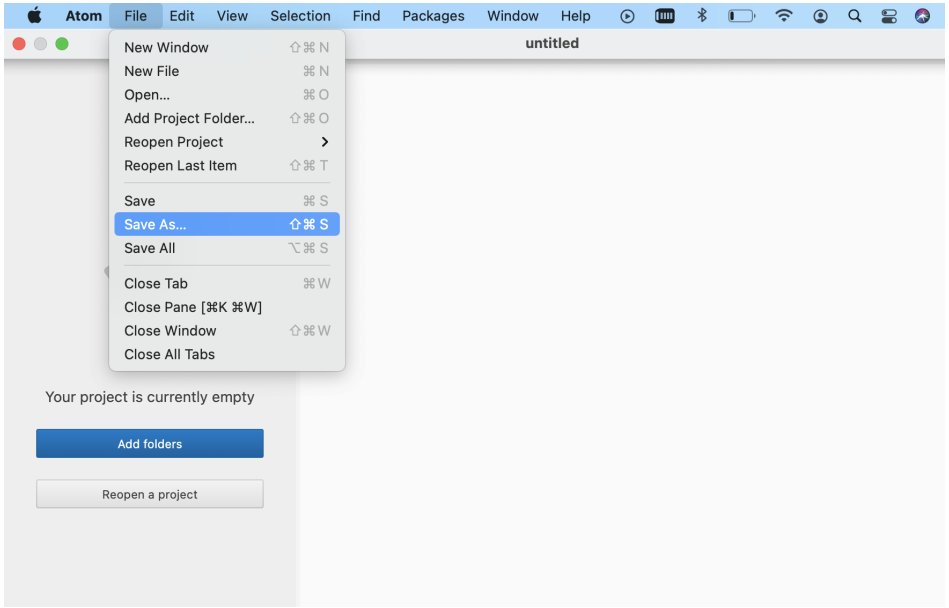
Step # 5: Install atom-python-virtualenv

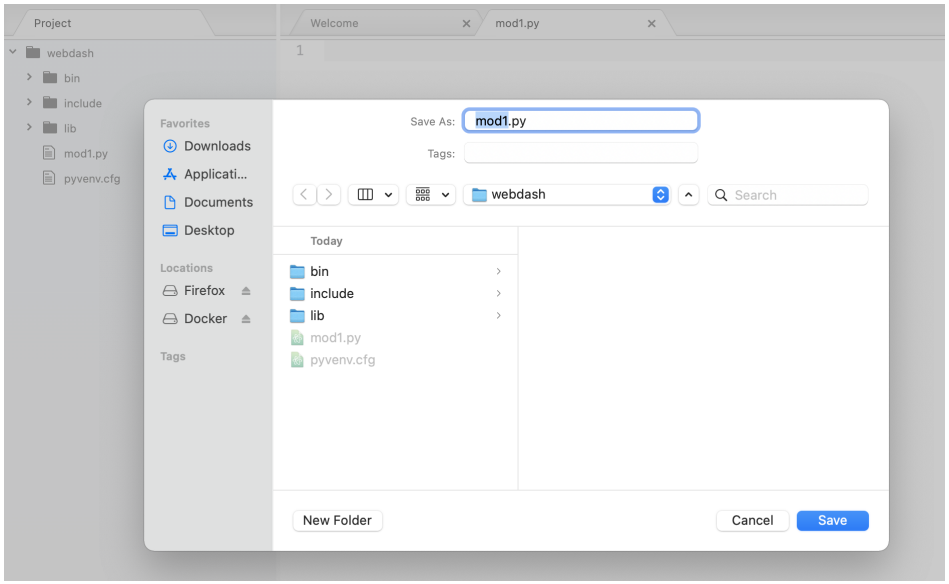
This will allow us to select our virtual environment.



Step # 6: Create a new File and Test Your Python



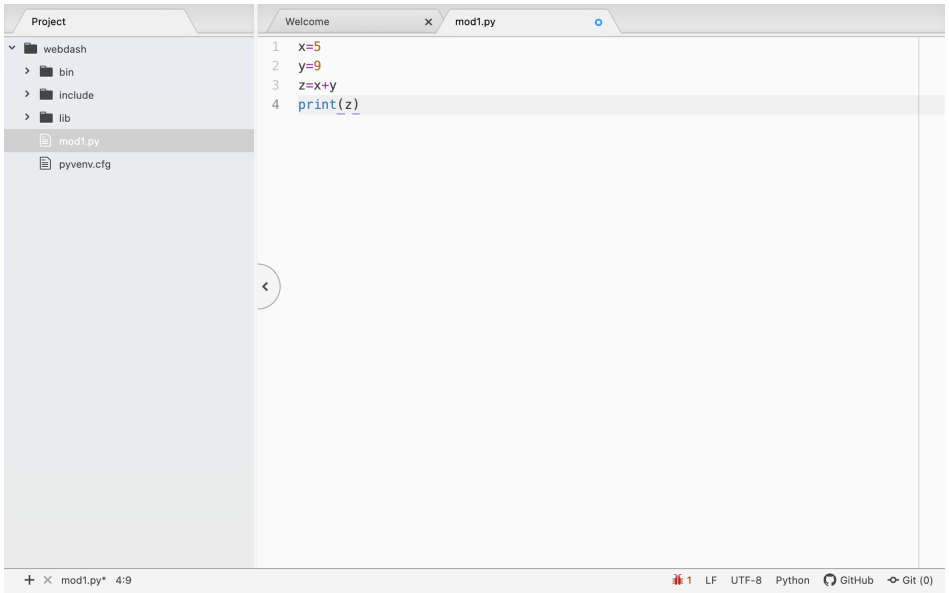


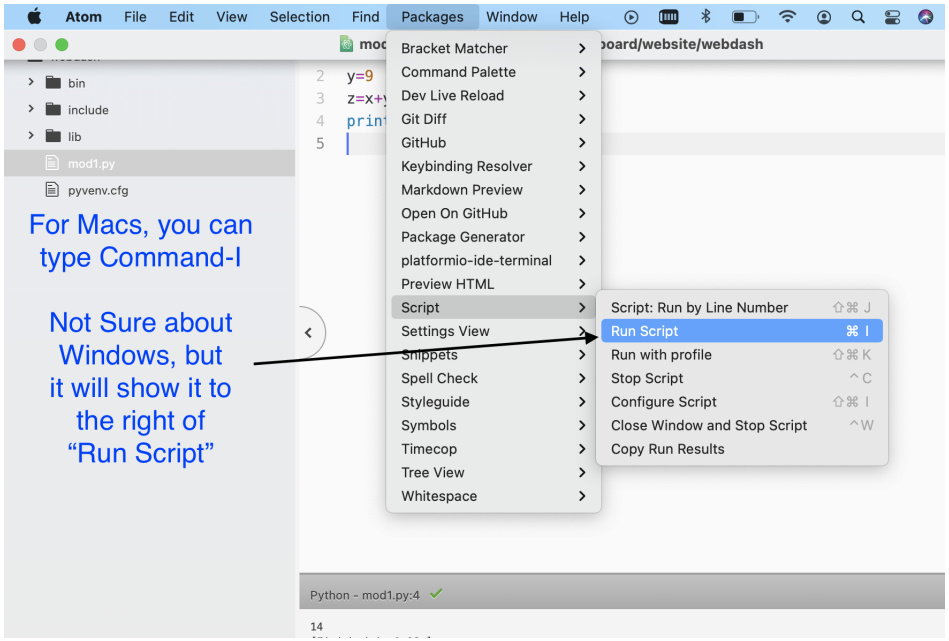


Open a new file and save the script file with a .py extentsion, in your class directory. Type in some python code, for example:

```
x=5
y=9
z=x+y
print(z)
```

Step # 6: Create a new File and Run Python Code





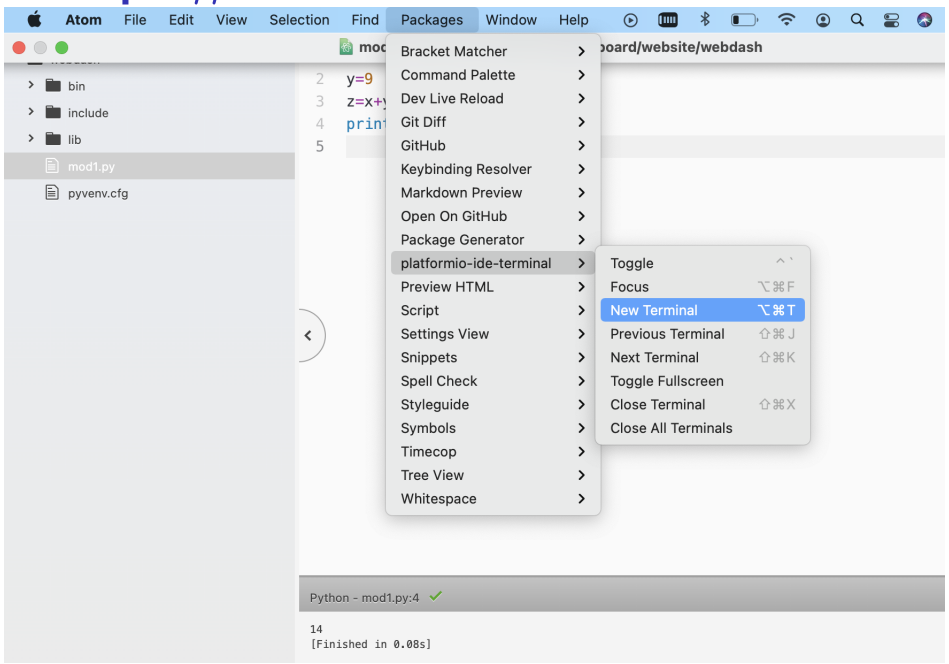
You can "Run Script" from the packages or use the short cut associated with it, on the right. Getting the answer 14, shows that your python code is working perfectly.

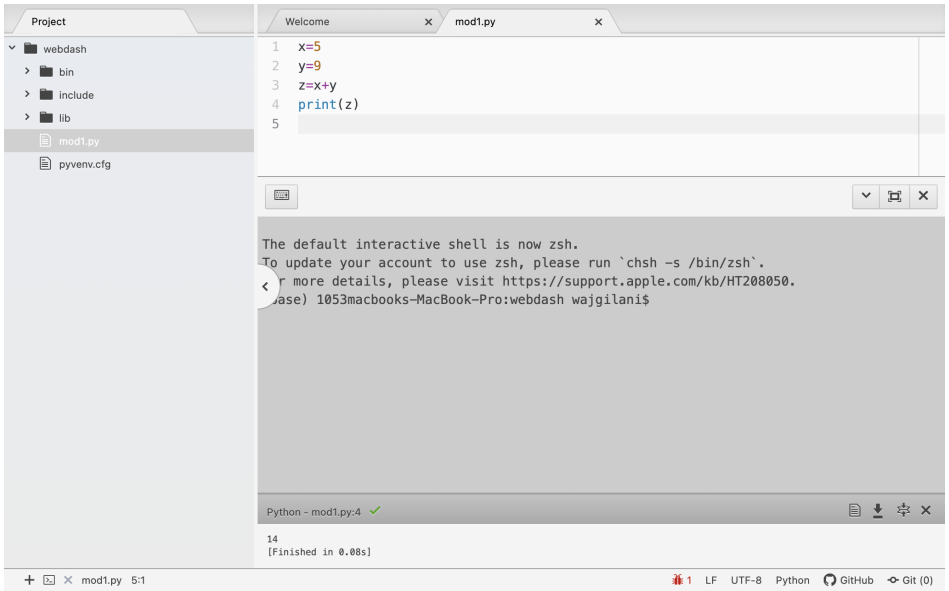
Step # 7: Create a Virtual Environment

For BAP, when we used PIP, to install libraries, we did it for the whole machine. Now we will create a virtual environment to separate our project's programming and running environment, separate from our general machine. **Why do we need to do this?**

- Python doesn't separate libraries by separate versions. Assume there are 2 programs, Program A and Program B, and they both use another program, Program C. If Program A needs a newer version of Program C, but Program B cannot use the newer version of Program C, this will cause a problem for our Program B. There is no way to specify one version for A and another for B. Python's solution Instead is to create a separate virtual environment for Program B, where the Program C in that environment will be the older version.
- We didn't need this for BAP, because Anaconda managed all the packages versions and interdependence between all the libraries. Now we are flying solo.
- We are going to run an asynchronous server on our laptop, we don't want it to cause problems for our normal operating system.

Step # 7: Create a Virtual Environment





We have to create a virtual environment using a terminal or command prompt. You can open it up directly on your computer, or you can open the one we downloaded earlier on atom, under Packages and platformio-ide-terminal.

Depending on your computer, the default python command might be using python 2 or 3, we want to use python 3, that's why we verify latest python download by typing in the command:

```
python3 –V
```

Which is Python 3.9.5 for me.

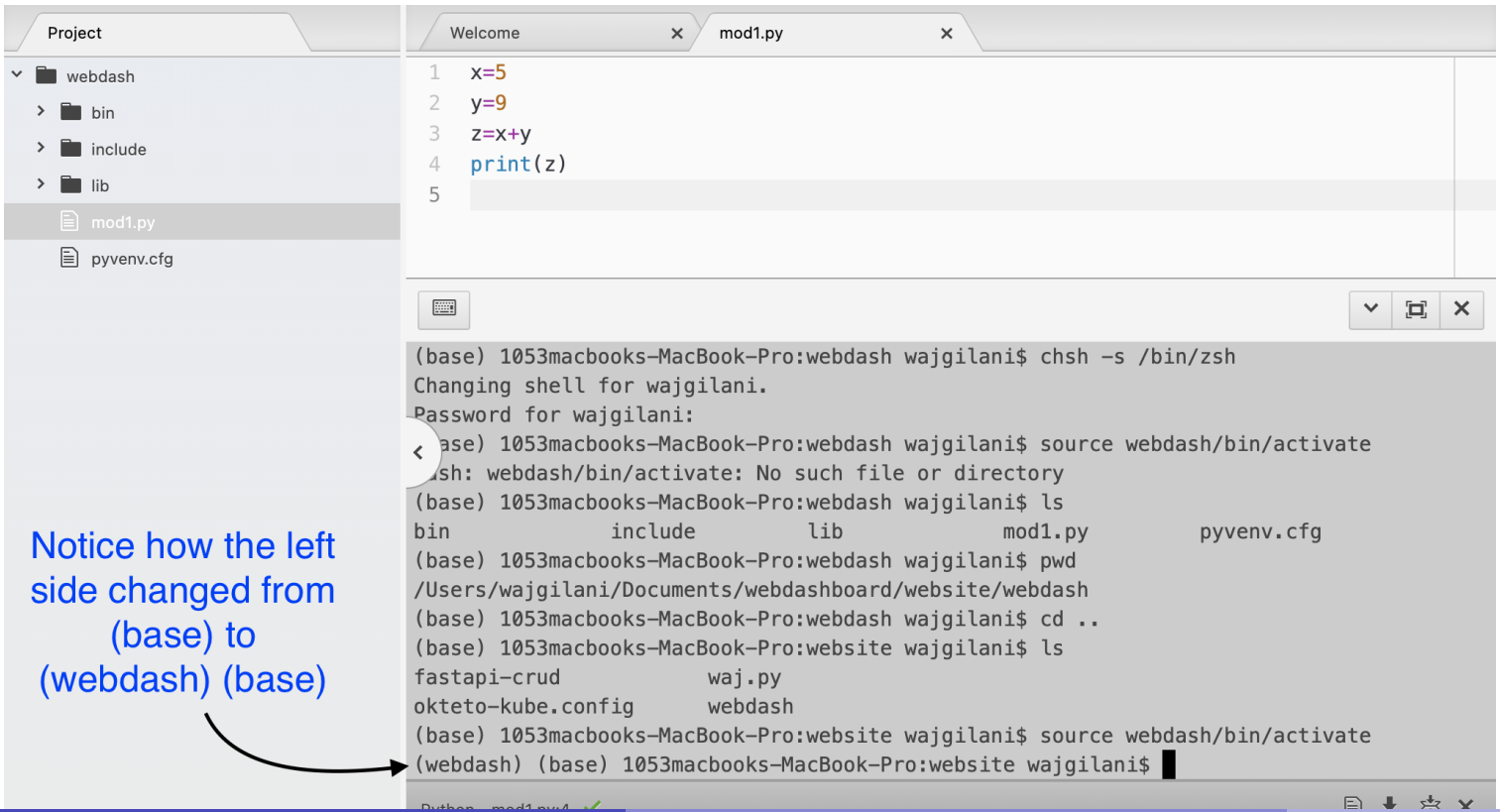
Step # 7: Create a Virtual Environment

To create the virtual environment (in this case my virtual environment will be named webdash), first type:

```
python3 -m venv webdash
```

Now to enter it, type in the following command:

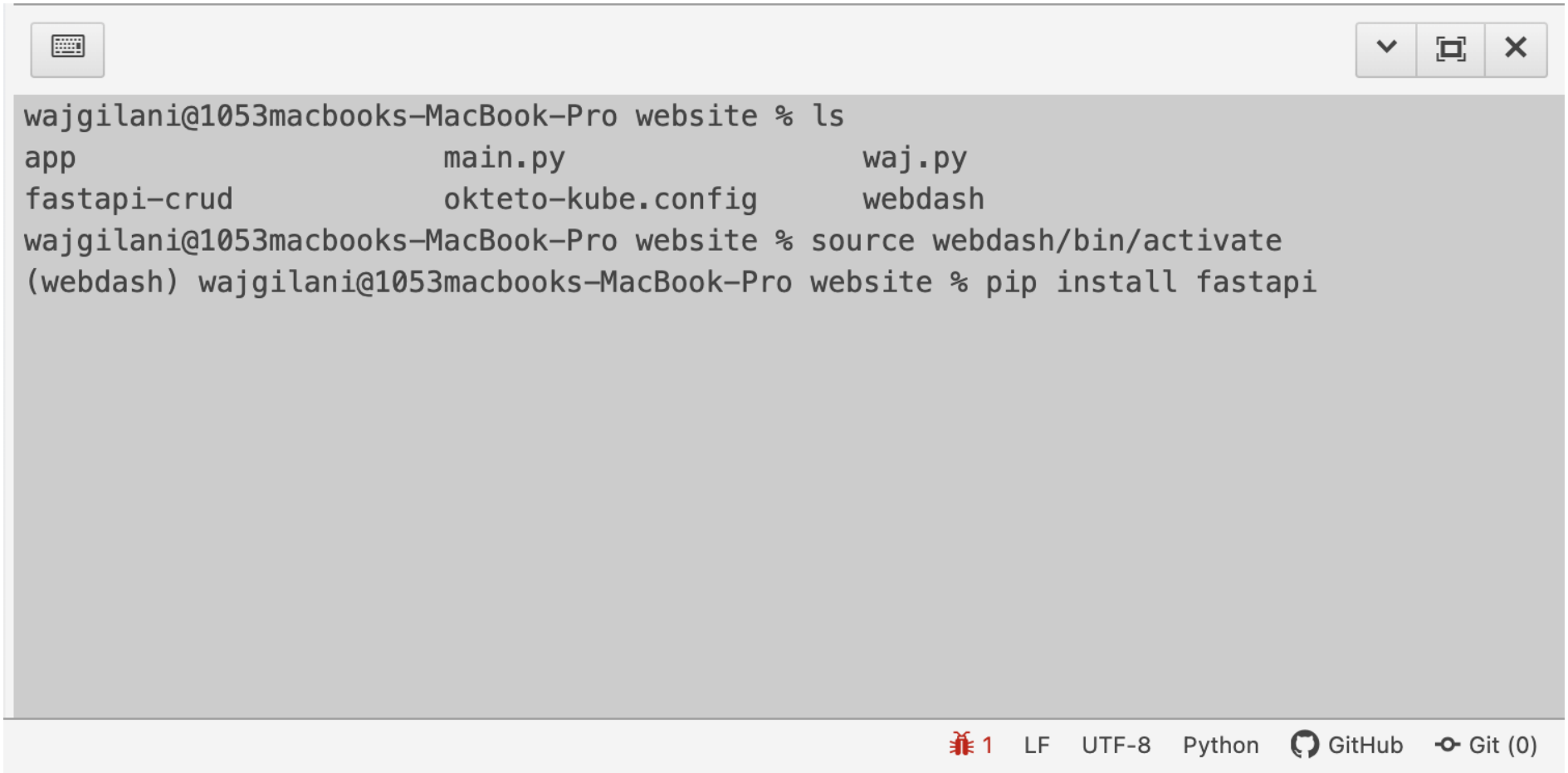
```
source webdash/bin/activate
```



To exit out of a virtual environment use the command, deactivate

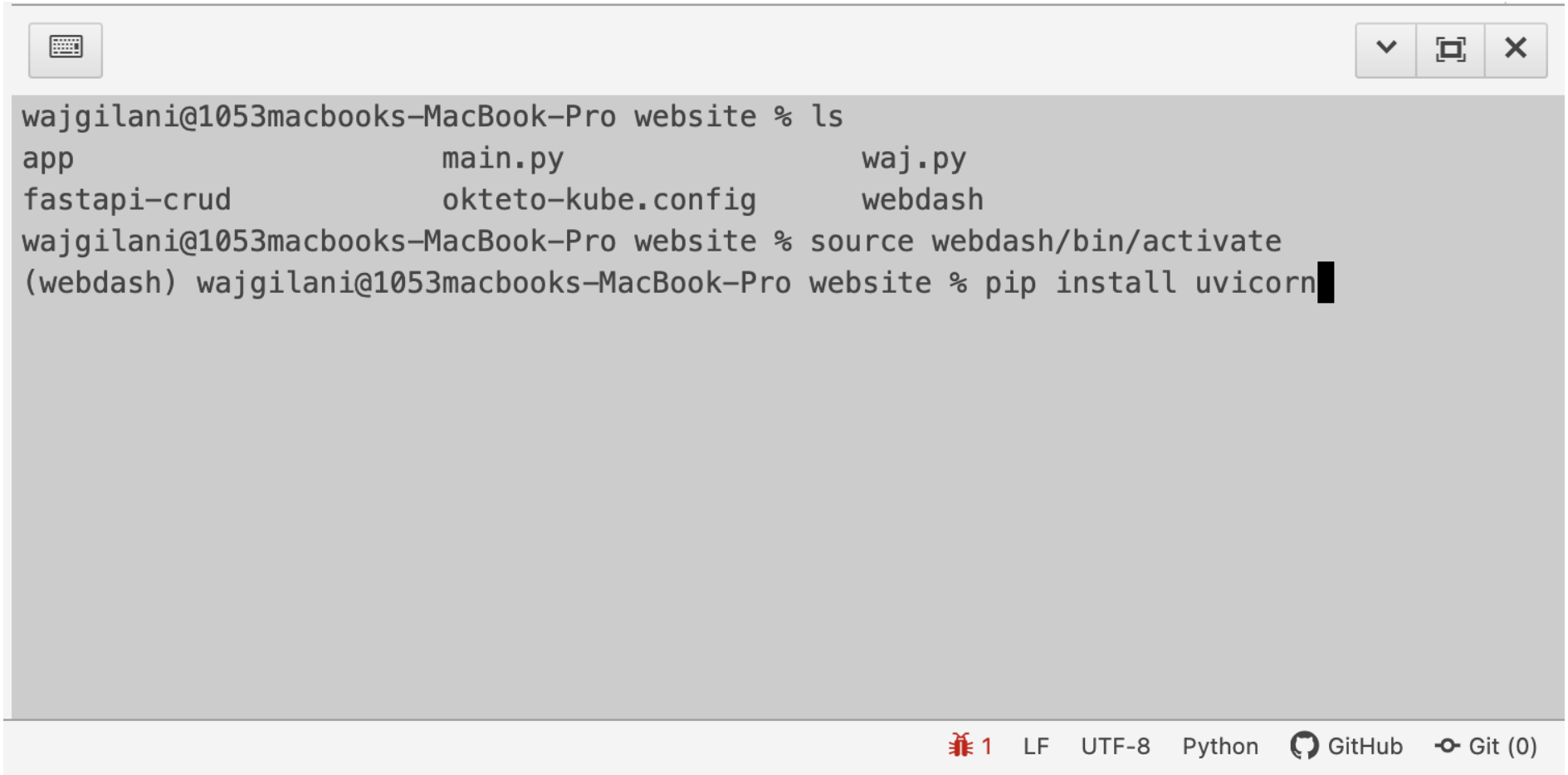
Step # 8: Download fastAPI Library

```
pip install fastapi
```

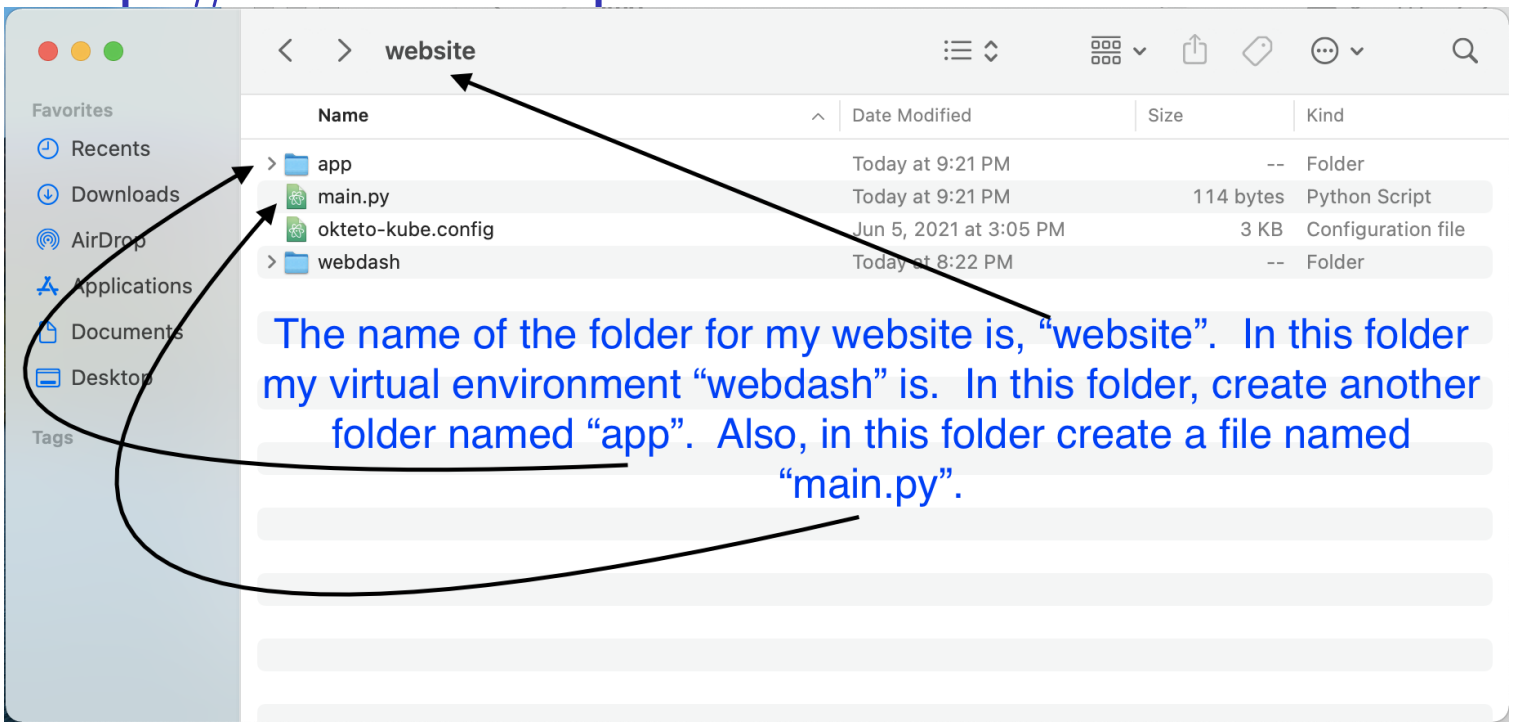


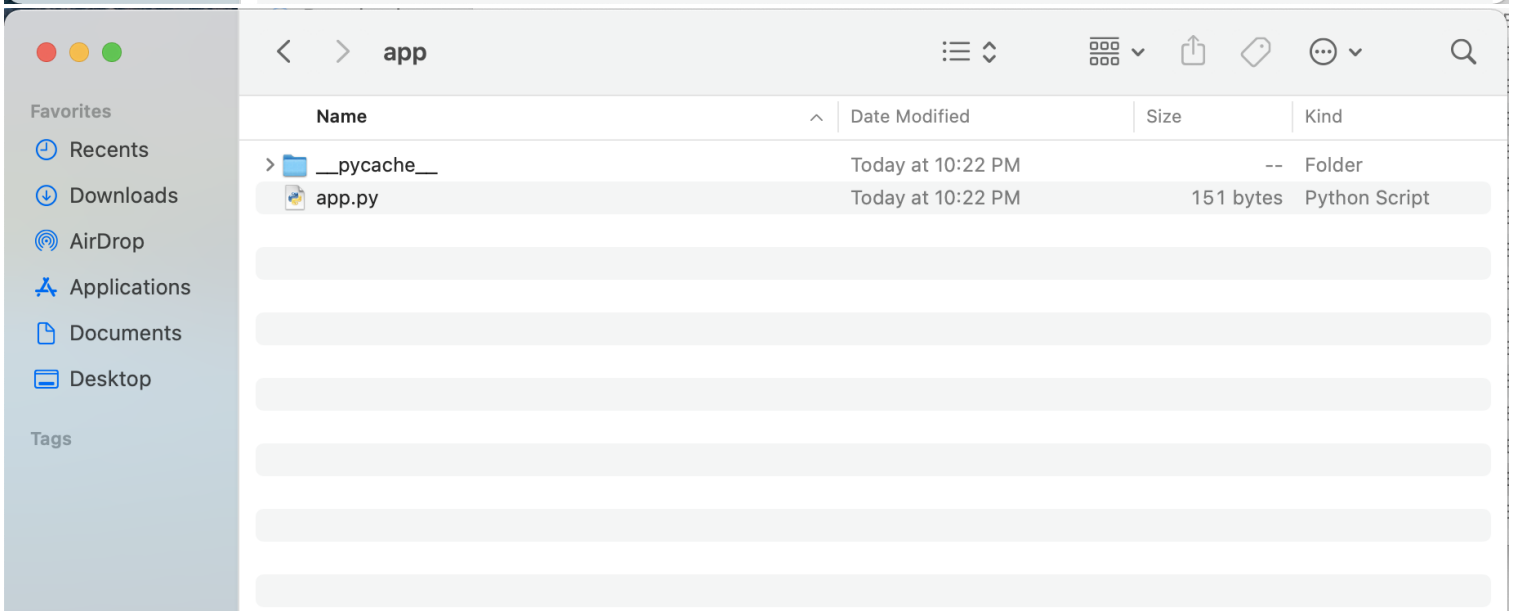
Step # 9: Download fastAPI Library

```
pip install uvicorn
```



Step # 10: Set Up Folders and Files





In the app folder, create another file named app.py

Step # 11: Write Code for app.py File

Now we will write our first web application, "Hello World"! Type in the following code for app.py:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def welcome():
  x = {"message": "HELLO WORLD!!! Welcome to fastAPI!!
      "}
  return x
```

Notice the variable x is a JSON object.

Step # 12: Write Code for main.py File

This is the code for main.py, main.py is the program that will run our web-server on our laptop.

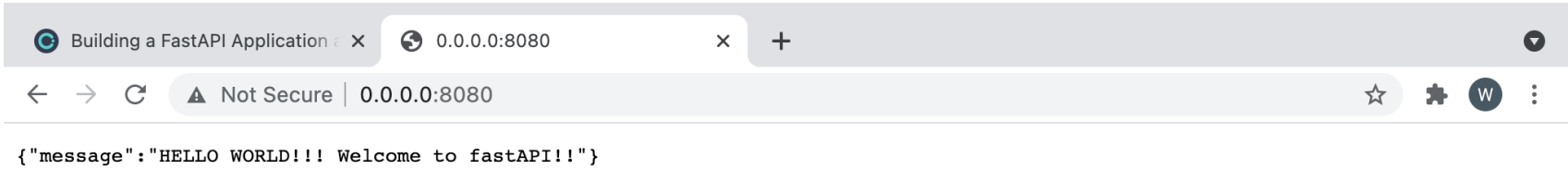
```
import uvicorn

if __name__ == "__main__":
  uvicorn.run("app.app:app", host="0.0.0.0", port
      =8080, reload=True)
```

Now run the command python main.py, to start the webserver:

```
(webdash) wajgilani@1053macbooks-MacBook-Pro website % python main.py
INFO:     Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
INFO:     Started reloader process [2790] using statreload
INFO:     Started server process [2793]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Notice the url, http://0.0.0.0:8080, take that url and put it in your browser.

Your First Web Application!



Notice that the webpage is "Not Secure", this will be addressed in module 2 when we load up our web application to the Okteto (and the public web).

Shutting Down the uvicorn webserver

To shut down the webserver, just type "control C":

```
^CINFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [2793]
INFO:     Stopping reloader process [2790]
(webdash) wajgilani@1053macbooks-MacBook-Pro website %
```