

Web Based Dashboards

Module # 1 (Setting 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 application. For that reason it is also better to download the latest python package. Go to <https://www.python.org/>:

The screenshot shows the official Python website at https://www.python.org/. The page features a dark blue header with the Python logo and the word "python" in white. A navigation bar below the header includes links for About, Downloads, Documentation, Community, Success Stories, News, and Events. On the left side, there is a code editor window displaying a Python script. The script outputs "Hello, I'm Python!" and asks for input, which is then printed back. To the right of the code, there is a section titled "Quick & Easy to Learn" with text explaining Python's readability and a link to a Python 3 overview. At the bottom right, there is a navigation bar with numbered buttons from 1 to 5.

```
# Python 3: Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!

# Input, assignment
>>> name = input('What is your name?\n')
>>> print('Hi, %s.' % name)
What is your name?
Python
Hi, Python.
```

Quick & Easy to Learn

Experienced programmers in any other language can pick up Python very quickly, and beginners find the clean syntax and indentation structure easy to learn. [Whet your appetite](#) with our Python 3 overview.

1 2 3 4 5

Step # 1: Download Latest Python

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

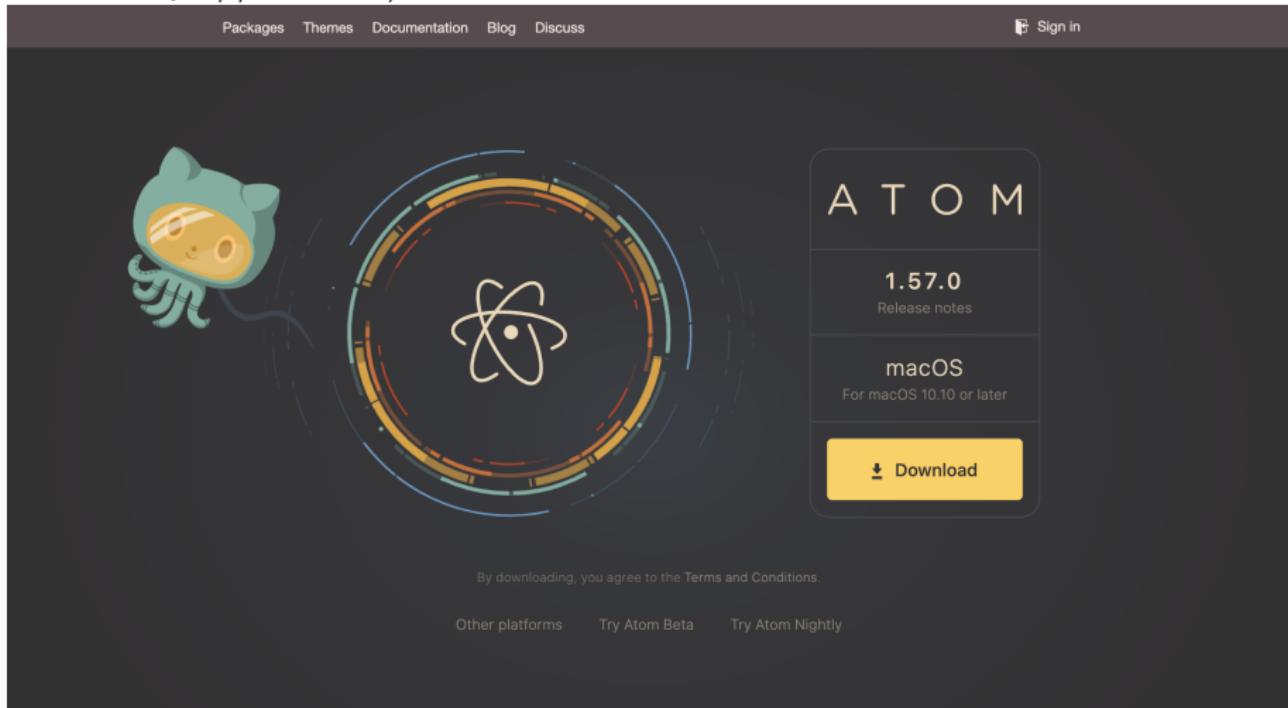
The screenshot shows the Python.org homepage with the URL https://www.python.org/ in the browser's address bar. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. The main content area features the Python logo and navigation tabs for About, Downloads, Documentation, Community, Success Stories, News, and Events. The Downloads tab is highlighted. A sidebar on the left contains code snippets and links for All releases, Source code, Windows, Mac OS X, Other Platforms, License, and Alternative Implementations. The Mac OS X section is expanded, showing a large button for "Download for Mac OS X" and a note that Python can be used on many operating systems. Arrows point from the "Downloads" tab to the "All releases" link and from the "All releases" link to the "Download for Mac OS X" button.

https://www.python.org/downloads/

Python is a programming language that lets you work quickly

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:

The screenshot shows the Atom text editor window. At the top is a blue menu bar with icons for Apple, Atom, File, Edit, View, Selection, Find, Packages, Window, and Help. A dropdown menu is open under the Atom icon, listing options: About Atom, View License, Version 1.57.0, Check for Update, Preferences..., Config..., Init Script..., Keymap..., Snippets..., Stylesheet..., Install Shell Commands, Services, Hide Atom, Hide Others, Show All, and Quit Atom. Below the menu bar is a toolbar with icons for Project, Home, Recent, Find, Replace, and others. The main area displays the Atom logo and the text "A hackable text editor for the 21st Century". It also includes a "For help, please visit" section with links to Atom docs, the forum at discuss.atom.io, and the Atom org, and a "Show Welcome Guide when opening Atom" checkbox. The bottom right corner features the atom.io logo.

To install libraries click on Preferences. In the Mac its under Atom > Preferences.

On Windows, it's under File > Settings

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

Click on Install:

The screenshot shows the Atom IDE interface with the 'Settings' tab selected. The left sidebar has a 'Project' tab and a 'Welcome Guide' tab. The main area displays the 'Core' settings. A large icon of a telescope is visible. The 'Updates' section contains a list of items, with the third item, 'The Atom GitHub API repository can be used to download packages', followed by an 'Install' button. A black arrow points from the text 'Click on Install' at the bottom left towards this 'Install' button. The right side of the screen lists various core settings with checkboxes, such as 'Allow Pending Pane Items', 'Audio Beep', 'Automatically Update', 'Close Deleted File Tabs', 'Close Empty Windows', 'Color Profile' (with a dropdown menu set to 'Use color profile configured in the operating system'), and 'Remove Empty Panes'. The 'Core Settings' section also includes a note about Atom's core settings affecting behavior unrelated to text editing.

Your project is currently empty

Add folders

Reopen a project

Click on Install

Core

Editor

URI Handling

Keybindings

Packages

Themes

Updates

Install

Show View opening At

Core Settings

These are Atom's core settings which affect behavior unrelated to text editing. Individual packages may have their own additional settings found within their package card in the [Packages list](#).

Allow Pending Pane Items
Allow items to be previewed without adding them to a pane permanently, such as when single clicking files in the tree view.

Audio Beep
Trigger the system's beep sound when certain actions cannot be executed or there are no results.

Automatically Update
Automatically update Atom when a new release is available.

Close Deleted File Tabs
Close corresponding editors when a file is deleted outside Atom.

Close Empty Windows
When a window with no open tabs or panes is given the 'Close Tab' command, close that window.

Color Profile
Specify whether Atom should use the operating system's color profile (recommended) or an alternative color profile.
Changing this setting will require a relaunch of Atom to take effect.

Use color profile configured in the operating system

Remove Empty Panes
When the last tab of a pane is closed, remove that pane as well.

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)

The screenshot shows the Atom IDE interface. On the left, there's a sidebar with a telescope icon and the text "Your project is currently empty". Below it are buttons for "Add folders" and "Reopen a project". A checkbox for "Show View" is checked. In the center, there's a "Welcome Guide" section with a "script" package listed. On the right, the main pane displays the "Install Packages" section with three packages listed:

- script** 3.32.2
Run code in Atom!
atom-community
Settings Uninstall Disable
- server-script** 0.1.3
Atom editor package to sync files and/or run a script on the server
mark-hahn
Install
- language-ags-script** 0.2.3
Adventure Game Studio (AGS) scripting language support in Atom
edmundito
Install

A large blue arrow points from the text "Type in Script, and click on Install" to the "Install" button for the "script" package.

Type in Script,
and click
on Install

Package	Version	Downloads
script	3.32.2	3,845,543
server-script	0.1.3	21,157
language-ags-script	0.2.3	1,839

Step # 4: Install platformio-ide-terminal

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

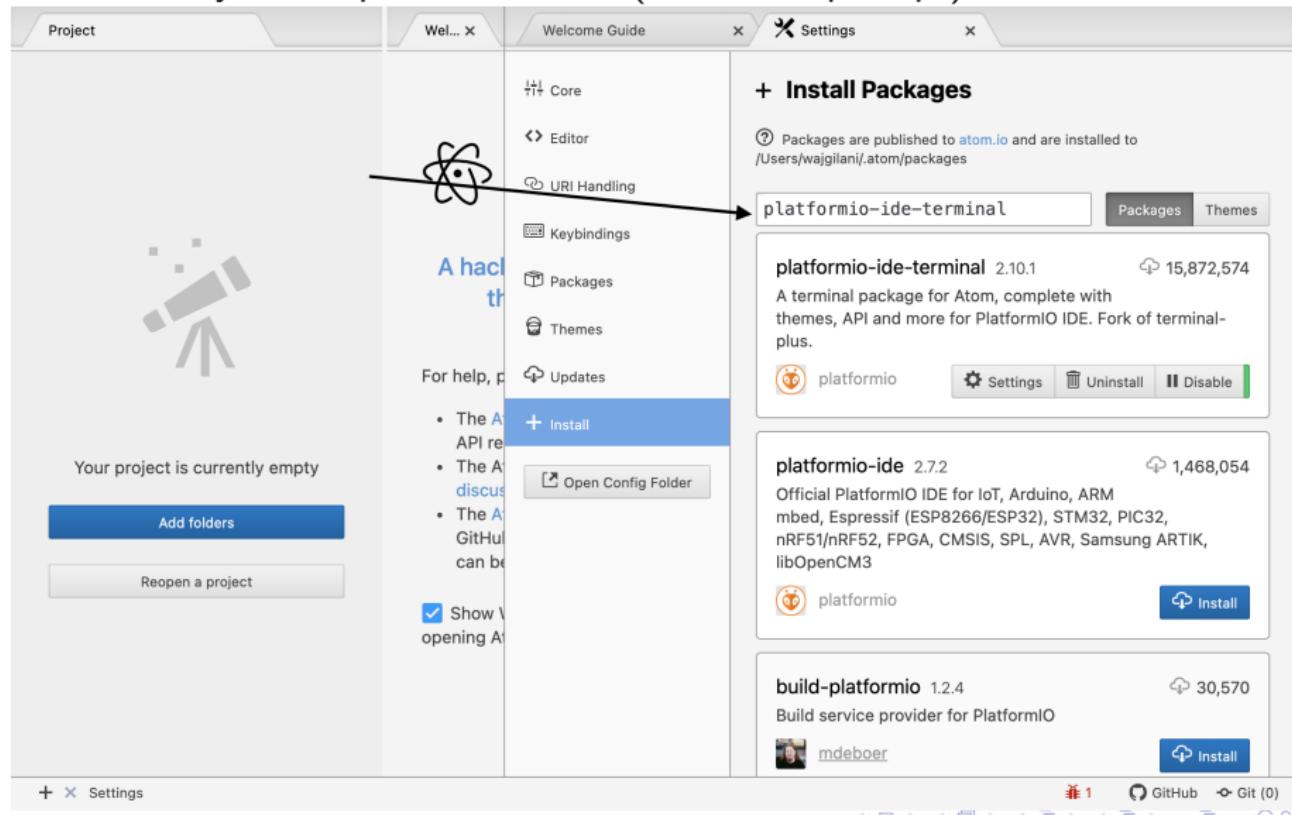
The screenshot shows the Atom IDE interface. On the left, there's a sidebar with a telescope icon and the text "Your project is currently empty". Below it are buttons for "Add folders" and "Reopen a project". A checkmark is next to the option "Show View opening Atom". The main area has tabs for "Project", "Welcome Guide", "Settings", and "Install Packages". The "Install Packages" tab is active, showing a search bar with "platformio-ide-terminal" and tabs for "Packages" and "Themes". Below the search bar, the package details are shown:

Package	Version	Downloads	Description
platformio-ide-terminal	2.10.1	15,872,574	A terminal package for Atom, complete with themes, API and more for PlatformIO IDE. Fork of terminal-plus.
platformio-ide	2.7.2	1,468,054	Official PlatformIO IDE for IoT, Arduino, ARM mbed, Espressif (ESP8266/ESP32), STM32, PIC32, nRF51/nRF52, FPGA, CMSIS, SPL, AVR, Samsung ARTIK, libOpenCM3
build-platformio	1.2.4	30,570	Build service provider for PlatformIO

At the bottom, there are GitHub and Git links, and standard browser navigation icons.

Step # 4: Install platformio-ide-terminal

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



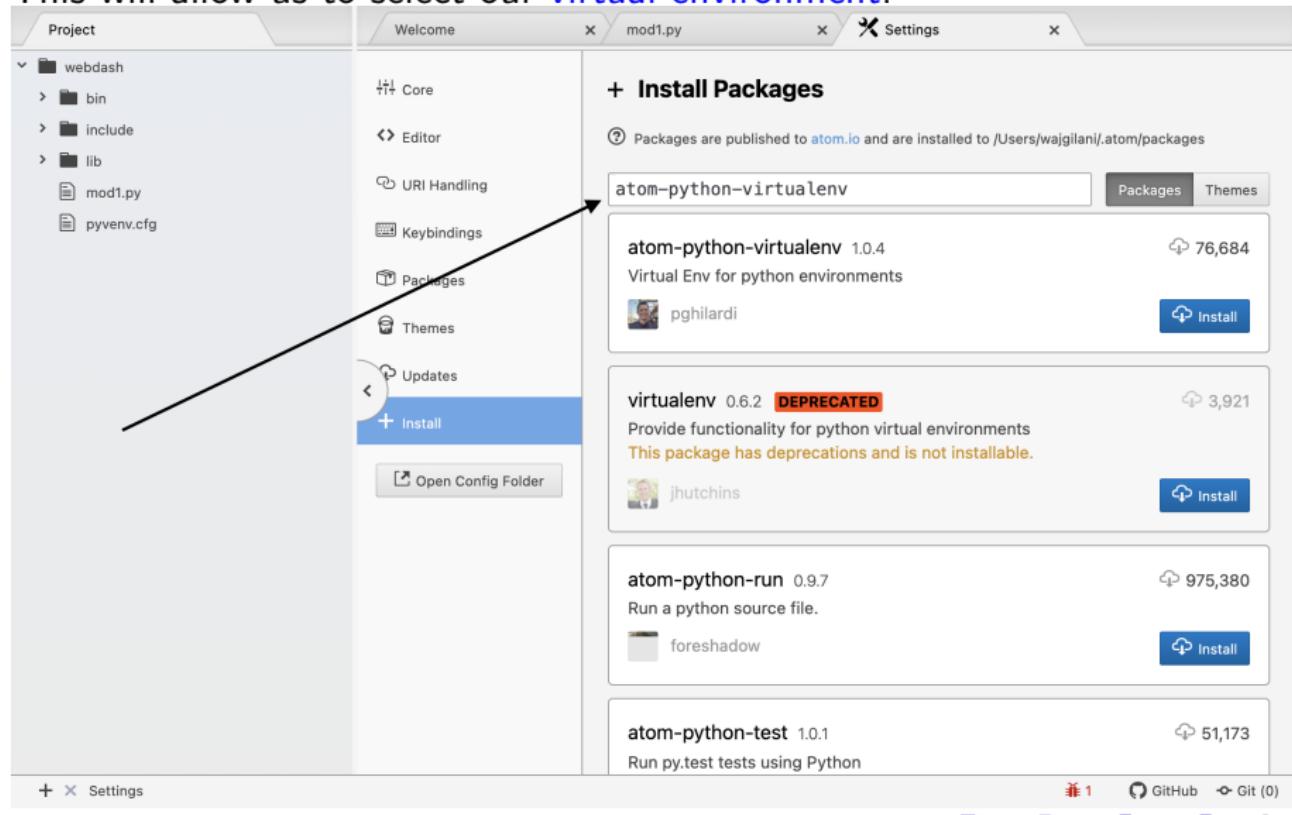
The screenshot shows the Atom IDE interface. On the left, there's a sidebar with a telescope icon and the text "Your project is currently empty". Below it are buttons for "Add folders" and "Reopen a project". In the center, there's a "Welcome Guide" section with a hand-drawn style illustration of a brain and the text "A hack". On the right, the main area has tabs for "Welcome Guide", "Settings", and "Install Packages". The "Install Packages" tab is active, showing a search bar with "platformio-ide-terminal" and buttons for "Packages" and "Themes". Below the search bar, the text says "Packages are published to atom.io and are installed to /Users/wajahgilani/.atom/packages". The search results list three packages:

- platformio-ide-terminal 2.10.1** by platformio (15,872,574) - A terminal package for Atom, complete with themes, API and more for PlatformIO IDE. Fork of terminal-plus.
- platformio-ide 2.7.2** by platformio (1,468,054) - Official PlatformIO IDE for IoT, Arduino, ARM mbed, Espressif (ESP8266/ESP32), STM32, PIC32, nRF51/nRF52, FPGA, CMSIS, SPL, AVR, Samsung ARTIK, libOpenCM3
- build-platformio 1.2.4** by mdeboer (30,570) - Build service provider for PlatformIO

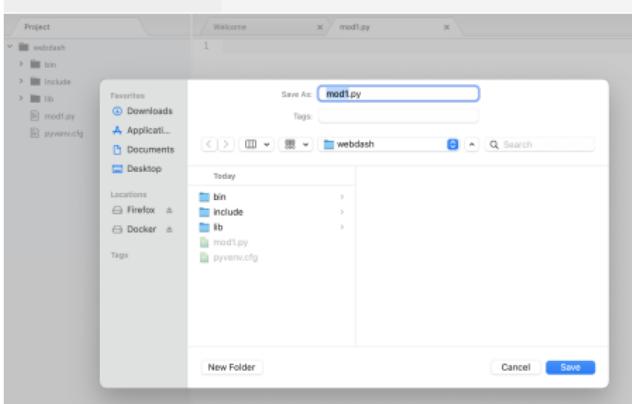
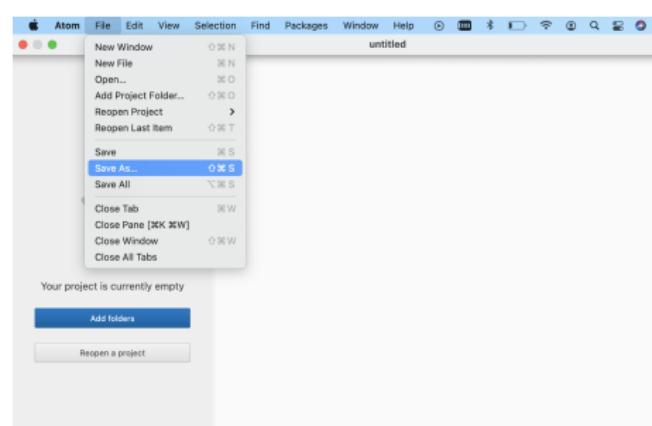
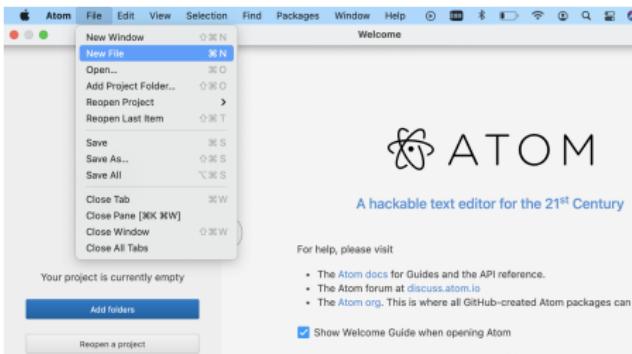
At the bottom, there are GitHub and Git links, and a status bar showing "+ 1 GitHub" and "Git (0)".

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 extension, 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

The screenshot shows the Atom code editor interface. On the left, the Project sidebar lists a 'mod1.py' file. The main editor window displays the following Python code:

```
1 x=5
2 y=9
3 z=x+y
4 print(z)
```

The status bar at the bottom indicates the file is saved and has 49 lines. On the right, the Packages menu is open, showing various options like Bracket Matcher, Command Palette, and GitHub. A callout arrow points from the text "Not Sure about Windows, but it will show it to the right of 'Run Script'" to the 'Run Script' option in the 'Script' submenu of the Packages menu.

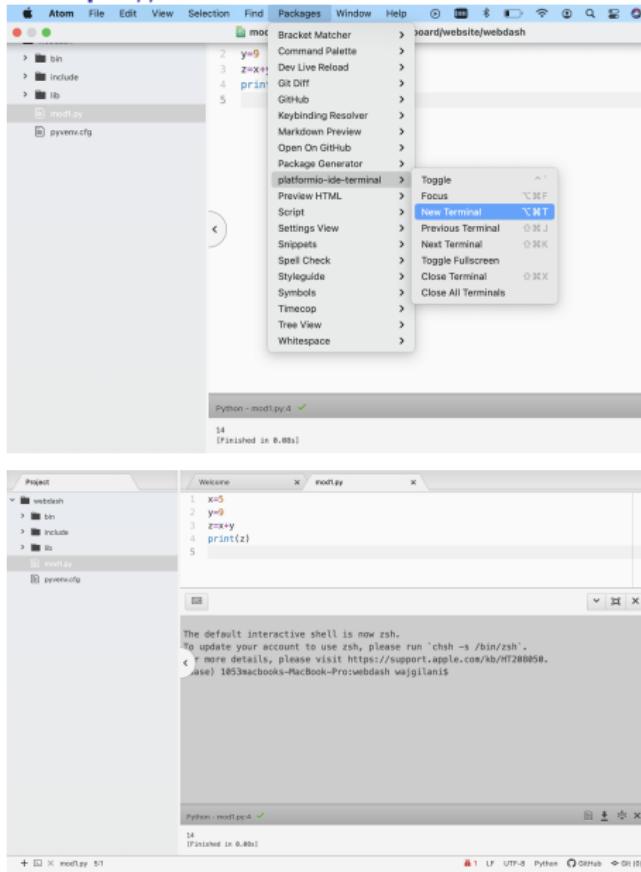
You can "Run Script" from the packages or use the short cut associated with it, on the right. Getting the answer 14, shows that your python code is working perfectly.

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
```

Project

- webdash
- bin
- include
- lib
- mod1.py
- pyvenv.cfg

Welcome mod1.py

```
1 x=5
2 y=9
3 z=x+y
4 print(z)
5
```

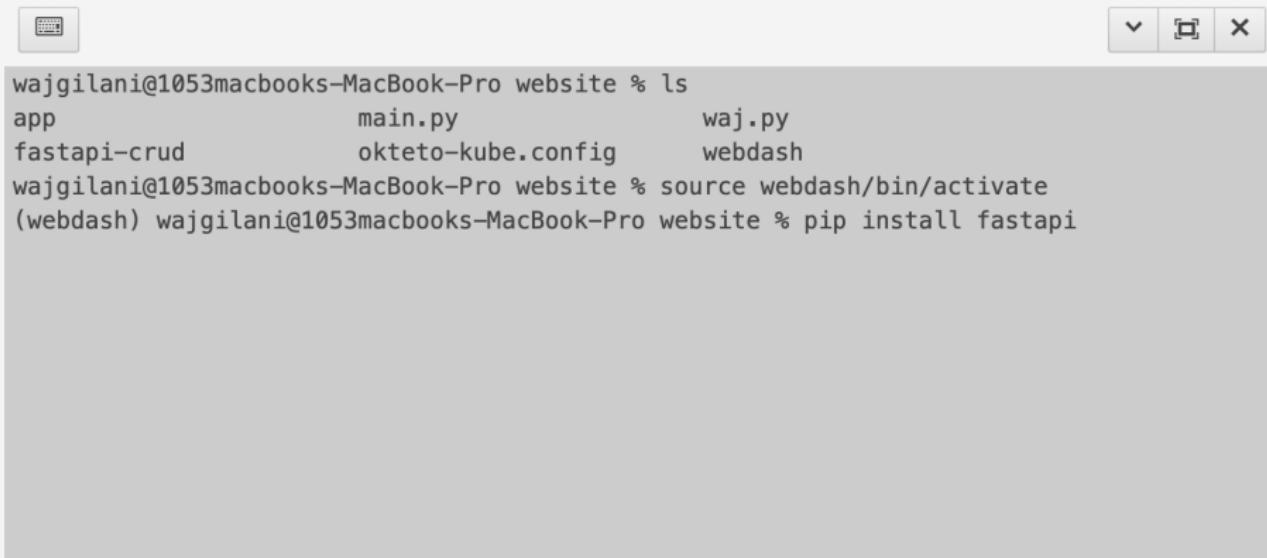
(base) 1053macbooks-MacBook-Pro:webdash wajgilani\$ chsh -s /bin/zsh
Changing shell for wajgilani.
Password for wajgilani:
< base> 1053macbooks-MacBook-Pro:webdash wajgilani\$ source webdash/bin/activate
zsh: webdash/bin/activate: No such file or directory
(base) 1053macbooks-MacBook-Pro:webdash wajgilani\$ ls
bin include lib mod1.py pyvenv.cfg
(base) 1053macbooks-MacBook-Pro:webdash wajgilani\$ pwd
/Users/wajgilani/Documents/webdashboard/website/webdash
(base) 1053macbooks-MacBook-Pro:webdash wajgilani\$ cd ..
(base) 1053macbooks-MacBook-Pro:website wajgilani\$ ls
fastapi-crud waj.py
okteto-kube.config webdash
(base) 1053macbooks-MacBook-Pro:website wajgilani\$ source webdash/bin/activate
(webdash) (base) 1053macbooks-MacBook-Pro:website wajgilani\$

Notice how the left side changed from (base) to (webdash) (base)

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

Step # 8: Download fastAPI Library

```
pip install fastapi
```



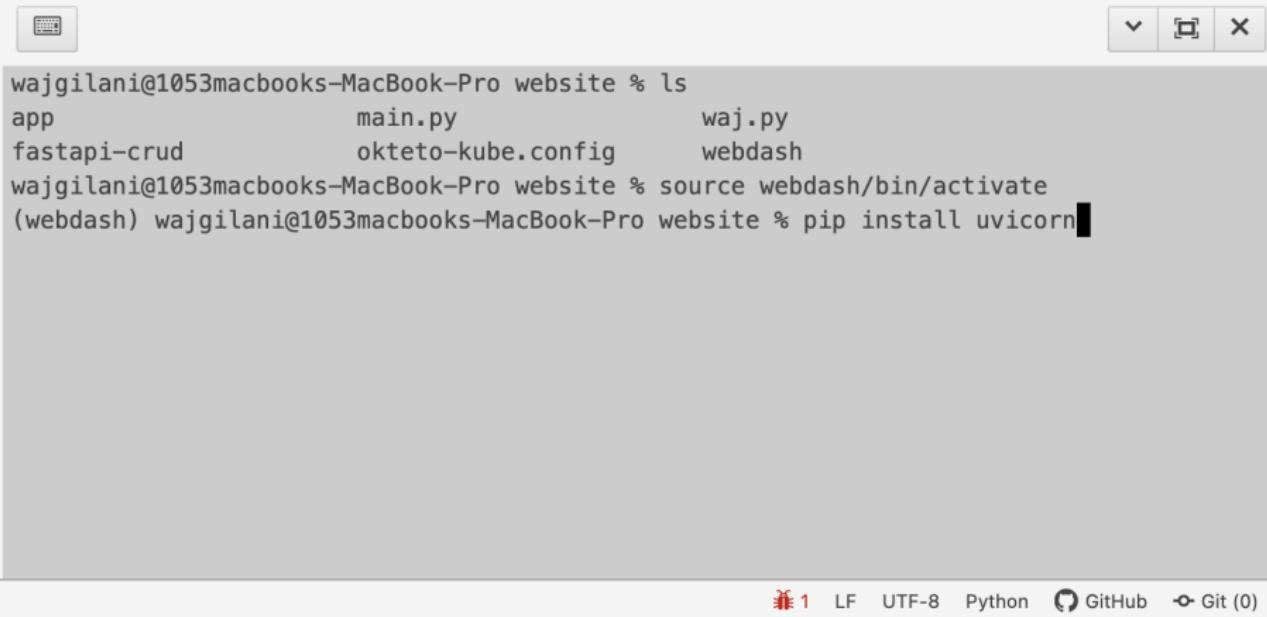
```
wajgilani@1053macbooks-MacBook-Pro website % ls
app                      main.py                  waj.py
fastapi-crud             okteto-kube.config    webdash
wajgilani@1053macbooks-MacBook-Pro website % source webdash/bin/activate
(webdash) wajgilani@1053macbooks-MacBook-Pro website % pip install fastapi
```

1 LF UTF-8 Python GitHub Git (0)



Step # 9: Download fastAPI Library

```
pip install uvicorn
```



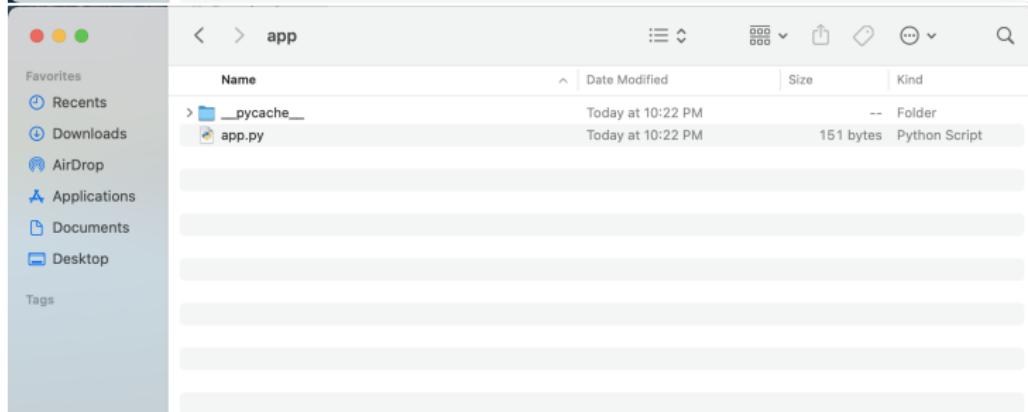
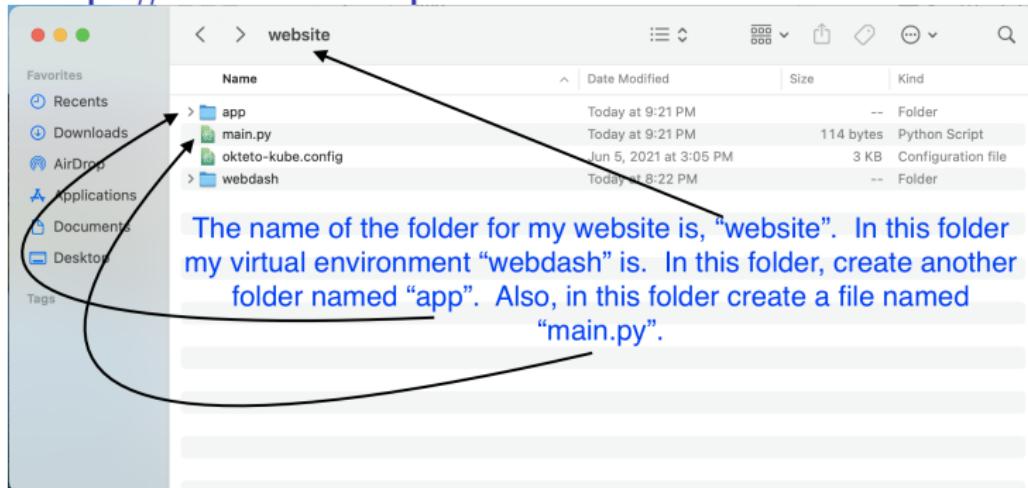
```
wajgilani@1053macbooks-MacBook-Pro website % ls
app                      main.py                  waj.py
fastapi-crud             okteto-kube.config    webdash
wajgilani@1053macbooks-MacBook-Pro website % source webdash/bin/activate
(webdash) wajgilani@1053macbooks-MacBook-Pro website % pip install uvicorn
```

The screenshot shows a terminal window with a light gray background. At the top, there are standard window controls (minimize, maximize, close). Below the title bar, the terminal prompt is "wajgilani@1053macbooks-MacBook-Pro website %". The user has run the command "ls" which lists several files: app, main.py, waj.py, fastapi-crud, okteto-kube.config, and webdash. Then, the user runs "source webdash/bin/activate" to enter a virtual environment. Finally, the user types "pip install uvicorn" and presses the Enter key. The status bar at the bottom of the terminal shows file statistics (1 file, LF line endings), the Python version (Python), and links to GitHub and Git.

1 LF UTF-8 Python GitHub Git (0)



Step # 10: Set Up Folders and Files



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

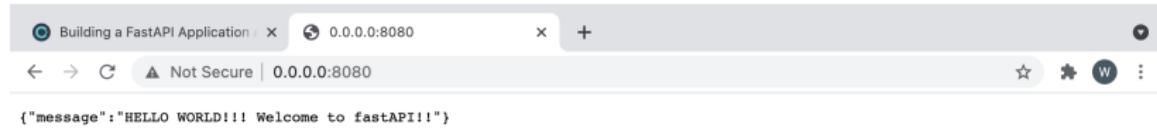
if __name__ == "__main__":
    uvicorn.run("app.app:app", host="0.0.0.0", port=8080, reload=True)
```

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!



Building a FastAPI Application X 0.0.0:8080 X +

Not Secure | 0.0.0.0:8080

{"message": "HELLO WORLD!!! Welcome to fastAPI!!!"}

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 unicorn 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 [2798]
(webdash) wajgilani@1053macbooks-MacBook-Pro website %
```