Introduction to Programming with Python: Day 1

Jackie Champagne (University of Arizona)

May 30, 2023

**1. Introduction**

Welcome to the NSF REU Python workshop! The goal of this workshop is to get you acquainted with basic navigation of the macOS command line terminal as well as some basic Python syntax. The terminal will soon become your home for navigating directories, editing script files, and starting up programs. This document will be a tutorial for some command line basics, and in the last section I will link you over to the beginning of the Python tutorial.

**1. Open a Terminal.** If you’re working on a Mac, this is under Applications>Utilities. You can also search for applications using Spotlight (command+space), and  
it’s a good idea to add it to your dock.

2. **Find out where you are.** Whenever you want to know what directory you’re working in, type pwd. This will print out the path to where you’re located; since you just opened a terminal, it should print your home directory, i.e., /Users/yourusername. To see what files and folders are in that directory, type ls.

3. **Create a directory**. It’s a good idea to stay organized with dedicated folders. To create a directory the same way you’d use “New Folder,” use the command mkdir followed by the name of a directory. For example, type the following:

mkdir TEST

4. **Navigate to a new directory**. To change directories, use the command cd followed by the name of the directory. Try navigating to your TEST directory. Then type pwd to confirm you made it TEST. To go back up one directory, type cd ..

To go back to your home directory at any point, simply type cd into the terminal and it will get you there. Navigate back to TEST.

**2 Using VI**

1. **Get started with a text editor**. There are a few command line text editors out there, but I personally stick with vi. Let’s practice a few basic vi commands. To create a file, type vi followed by a filename, including the extension. For example, vi test.py

2. **Edit your file**. This should have opened a window for you. To begin inserting text, type i. The window should say --INSERT-- at the bottom. Begin typing something: print(‘Hello world!’) To get out of insert mode, press ESC.

3. **Save your file**. Always make sure you exit insert mode before saving. To save without quitting, type :w and press enter. To quit without saving, type :q. To save and quit, type :wq. Save and quit this file.

There is a full dictionary of vi functionality here: *http://linuxfocus.org/~guido/vi/vilearn/1basics.txt*

**3 Moving/Copying Files**

By now you should have one file in your TEST directory called test.py. You can copy files directly from the command line, which is highly recommended if you are editing code or modifying data but you want to keep the original.

1. **Copy your file**. To make a copy of your file, type cp followed by the name of the file to copy and then the name of the copy. Type cp test.py test2.py to copy your test.py to test2.py.

2. **Rename your copied file**. The command mv can be used either to rename a file or move it to a new location. To rename it, type mv followed by the current name of the file followed by the new name of the file. If the last argument contains the path to a directory that exists, it will move it there, but if it is not a directory, it will rename the file. Type mv test2.py blah.py to rename test2.py to blah.py.

3. **Move your copied file.** To move a file to a new location, type mv followed by the name of the file followed by the directory you’d like to move it to. The name of the directory should be the full path, like /Users/jackie/Downloads, but there are a few helpful shortcuts. “~/” is the shortcut for your home directory, so “~/Downloads” would work. “..” is the shortcut for “one directory above here” like we learned before, and “.” is the shortcut for your current working directory. Move blah.py one directory above you. (Note, always use pwd and ls to navigate through directories)

4. **Delete your files.** To delete a file, type rm followed by the filename. Navigate to your home directory and delete blah.py. ***Be very careful with rm as it cannot be undone!!***

5. **Copy a directory.** To copy or move a full directory rather than a single file, add a space and then -r to add a recursive option.

cp -r TEST/ TEST2/

rm -r TEST2/

**Let us use what we just learned into practice.**

**For this exercise we will be making your research folder where you will house all the data and scripts you will be working on this summer.**

**Please navigate to your Desktop directory. Once you are there we ask that you follow the steps below:**

• Please create a directory named astrolab and this will be the top directory that will house your research for the summer

• Next go into the directory that you have just made and make the following 2 directories

⁃ data

⁃ scripts

• Next go into the data directory and make the following 2 directories

⁃ raw

⁃ processed

• Next we will create what is called a README.txt file for the processed and raw directory

⁃ This README.txt will keep track of what data is in the folder and gives useful context of where and when the data was taken

⁃ This is especially true in the processed folder as something has been done to the raw data to get it to where it is at, so writing down in the README.txt file what happened to the data is useful information to have

⁃ Since we have no data as of now we will simply make the file and write out our name, affiliations and Today’s Date

⁃ Your Name, University of Texas at Austin, Date

⁃ NOTE: it is a good idea that as you update the folder you also update the README.txt file and the date to reflect when it was last updated

* Copy the README.txt file that you have just made in your directory and copy it to the other directory in the data folder (i.e: either raw or processed)

• The last step we will do is populate the script folder with the test.py file we created, please move the test.py to the scripts/ directory

The data/ folder which contains the two subdirectories, raw and processed, will house all of your data.

As the subfolders imply the raw folder will have the original unaltered raw data. It is good practice to have the original data in a separate folder so that you never accidentally overwrite it.

At some point in your research you will have to perform some analysis on the raw data. This can be sub-selecting a smaller catalog with only sources you are interested in working with.

The processed/ folder will contain all the processed data you will create and it would be here where you store a subsample of your data, for example.

The scripts/ folder is a catch all folder where you will put all coding items into. This includes python scripts (.py files) or Jupyter notebooks (.ipynb) or anything coding related should be stored in this folder

These commands are only a small amount of linux/terminal commands that you can use. There is a file in this directory titled Week\_1\_Terminal\_Controls.pdf.

This document outlines some other useful terminal commands that you may find useful as you carry out your research project. There is also some blank spots that you can then fill in with new commands that you have found that were useful for you.

We encourage you to use this page as a reference to remind yourself what the commands covered today do and how you can use them.

**4 Loading up Python**

To use Python from the command line, you can always type python to begin a session. When you want to quit, type quit().

To run a script, simply type python scriptname.py

Let us try this out on the script we have just made, go to your script/ directory that has the test.py file and type python test.py on the command line

**5 Loading up iPython**

To use iPython from the command line, you can type ipython to begin a session. When you want to quit, type exit

You can use this terminal to run code and test things out if you do not want to fire up a Jupyter Notebook and it is more user-friendly that just a regular python interpreter.

**6 Loading up Jupyter Notebook**

To open a notebook, first you’ll need to know what your current local directory is. Use pwd to make sure you know where you are. Note: when you are in a jupyter notebook or jupyter lab, you can not go to a higher directory than your current directory. Eg. if you’re in the ‘~/Downloads/research’ directory, you can’t open the notebook ‘~/Downloads/notebook\_1.ipynb’ b/c you can’t access things in ‘~/Downloads’.

Then, you can use a notebook by typing jupyter notebook or jupyter lab . When you’re done with the notebook, press ctrl+C to exit out of the terminal session.

One last useful Terminal thing: the jupyter notebook, and some other programs, will effectively ’kill’ a terminal session, meaning you can no longer use this window until the program is closed. You can open new Terminal windows, or keep it available for use by typing & after the name of the program, i.e. jupyter notebook &. When using the ampersand, you’ll see a number next to the job. When you want to quit the notebook session, you can type kill [jobnumber]. Open up the Workshop Day 1 notebook from the NSF\_REU\_python\_bootcamp\_2024 directory and selecting Day1 Folder.