Introduction to Programming

Welcome to Yeah Education’s Data Science course! Before each lecture I will have a notes handout which will contain the main points of the lecture. Feel free to look over it during class as a guide but it will not contain everything I talk about.

[Computer Setup]

Refer to your computer setup handouts for instructions on how to install Python. We will be using Python 3.6 in this class that comes bundled in Anaconda. Anaconda is a package management software that pre-installs many important Python libraries for you. Python libraries allow you to run functions like **max** or **read\_csv** without manually having to code it yourself. You will be relying on libraries constantly when you program.

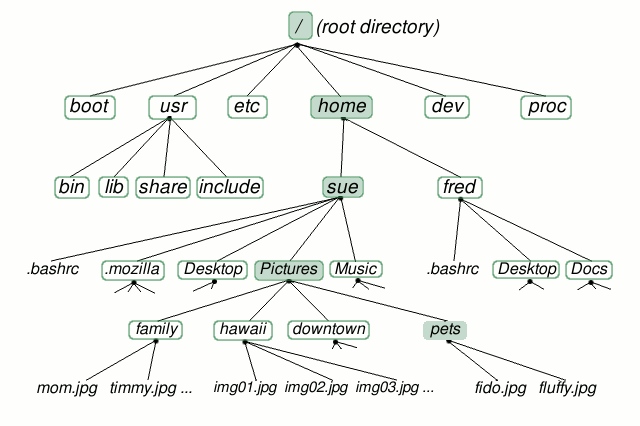
You also installed Sublime Text. This is a text editor that allows you to edit your code easily.

[Navigating Your Computer]

For Windows users, I had you install Git Bash. This is to mimic the terminal that Mac computers have. It behaves very similarly.

Git Bash and the terminal allow you to look around your computer (change folders, launch applications, write documents, etc.) without a mouse. This was how computers were traditionally navigated. It is also very helpful to know as a data scientist because you can do things MUCH faster.

Your computer files are organized in a tree-structure like so:



To view your files in your current directory: **ls**

To move into another folder in your current directory: **cd <folder\_name>**

To move “up” into the parent folder: **cd ..**

To edit a text file with Sublime Text Editor (you probably need to figure out how to setup this shortcut manually): **subl <file\_name>**

And that’s all you need to know for now!

[Python Basics]

--- Here I went on Kaggle to show what Python syntax looks like ---

Much of the basic concepts are better explained while typing on the computer. Here I will only leave some ideas I cover in lecture:

Data types: float, integer, string, boolean

Storing data in variables

Lists, list splicing, item replacing: EXAMPLE: **x = [0, 5, 6, 2, 4]**

**x[2]** prints 6

**x[2] = 3** replaces 6 with 3

Generators: EXAMPLE: **range(10)**

List comprehensions: EXAMPLE: **[x for x in range(10)]**

For loops: EXAMPLE (first line of syntax): **for i in range(10):**

While loops

If-elif-else conditions and logic

Functions: EXAMPLE (first line of syntax): **def calculate\_fib(n):**

Global and Local Scope

A little about global vs. local scope. In Python, when you declare your variables, your variables exist in Python’s global environment. It exists everywhere. When you make a function, however, all variables you declare exist in the function’s local environment. Variables created within a function only exist within that function and cannot be used outside of the function.

--- I give in-class exercises here ---

[NumPy – Your First Library]

NumPy is one of the most popular libraries for data scientists. It is a library that has functions and syntactic sugar that allow you to find the maximum value of a list, sort a list, store data inside matrices, multiply matrices, and much more. Your readings and homework next week will focus on how to use this library.

To actually get this library inside Python just type: **import numpy as np**

The “**as <nickname>**” renames numpy to just np to make it easier to type later. You can give libraries any nickname you want.

To use the **max** function inside numpy just type:

**np.max( <list> or <array>)**

Notice that we have to reference the library first followed by a “**.**”. Think of the dot as telling numpy that we want to use one of its functions. The syntax used by all libraries is: **<library\_name>.<function\_name>**

[NumPy – Arrays]

Arrays can be considered the same thing as matrices. When you work with real data, your data will commonly be stored in either arrays or, covered later in the course, something called DataFrames. The reason is that it is convenient to analyze data when they are in tables.

To create a 1 dimensional array just type: **np.array(<list>)**

To create a 2 dimensional array (matrix) just type: **np.array([<list>,<list>,…,<list>])**

Note: Each **<list>** represents a row in your array.

To multiply element-wise between arrays: **array1 \* array2**

To matrix multiply between arrays: **array1 @ array2** [See board for traditional matrix notation]

To take the transpose of an array: **array1.T**

To get the dimensions of your array: **array1.shape**

Subsetting an array is simple

One-value: **array1[row\_index, column\_index]**

A row: **array1[row\_index, :]**

A column: **array1[:, column\_index]**

Specific rows and columns: **array1[r\_id1:r\_id2, c\_id1:c\_id2]**

The colon “**<left index>:<right index>**” notation means “go from the left index to the right index MINUS ONE”. In math terms, we take the left index inclusive and go to the right index exclusive.

--- I give in class exercises here ---

[Quiz Instructions]

Quizzes and homework require you to download the appropriate files from my github repository. A repository is a storage space that you can dump all your files and data in, much like your own computer’s file system. In your case, you are going to download my files. To do so, in a separate terminal window (launch another Git Bash terminal in Windows or Command+t on Mac) type:

**git clone** [**https://github.com/jc003/datascience\_yeah.git**](https://github.com/jc003/datascience_yeah.git)

This will copy my repository/folder named “datascience\_yeah” into your computer. Explore the repository. You will find folders for homework, quizzes, and my notes. Your homework that is due next week is already there.

~~Right now navigate to the quizzes folder and open up the quiz template in a text editor. I will be writing the questions on the board so go ahead and open the file. You will have roughly 20 minutes to work on the quiz. Quizzes are to be done individually with no collaboration. When you are finished, email your python code file to~~ [~~jerryc@berkeley.edu~~](mailto:jerryc@berkeley.edu) ~~and wait for others to finish.~~

Since I didn’t have time to give the quiz to you during class, the quiz is now part of your homework. Please submit your quiz in the same way you submit your homework. You may also collaborate with other students on how to solve the quiz problems.

In the future if you want to update your repository (because I made some changes to the quiz or homework file) or to grab new homework assignments type in your terminal:

**git pull**

\*If you get a merge error, that means git does not know how to update your files. This happens for example when you start writing solutions for hw1.py, but then I make changes to the blank hw1.py template to the repository. When you do “git pull”, git (the program) will look at my hw1.py file and your hw1.py file and be confused because it won’t know which lines to add or delete in the correct order. To fix, make a copy of your hw1.py (e.g. hw1\_copy.py), delete your hw1.py, type git pull, then copy your solutions from hw1\_copy.py to the fresh copy of hw1.py.

[Homework Instructions]

You have already downloaded the homework for this week. Navigate to the appropriate folder and open up **hw1.py**. Having done the readings might help for this particular assignment. In general, if you don’t know what a function does, go look it up! Doing google searches on your own should be a natural habit. Homework 1 is due before start of class next week 4/15. Likewise for quizzes, please email your python code file to [jerryc@berkeley.edu](mailto:jerryc@berkeley.edu) once you are done.

Note: Any time you see “FILL ME” that is a problem worth 2 points. I give 2 points for a correct answer. 1 point for a wrong answer but good progress was made, and 0 points for leaving it blank.

Note: The homework problems are intended to be more challenging than the in-class examples or the quiz problems. I highly recommend working with other students together when doing these problems.