MR2020 Worksheet/Homework 1: Data Types, Assignment, Control Flows

Complete by August 6. Test out what you've learned on these examples in class or at home. You don't have to submit this for a grade, but you should be able to work through this worksheet by the due date.

Assigning New Variables

1. Assign a new variable called "A" the number 3

2. Create a new list called "B1" that contains the numbers 4.5, 2.3, and 3.6 in that order.

3. Create a tuple containing the same numbers but call it "B2".

4. Create a set from the items in a list (called whatever you want it to be) that contains 'apple', 'orange', 'banana', and 'orange' in that order. (You can cast the list as a set, but I'd also like to see that you know what would be returned by doing so.)

5. Create a dictionary named "dict". In it, define the following keys and paired values:

Keys	Values
Temperature	25
Dewpoint	18
Wind Speed	10

Manipulating or Reassigning Existing Variables

6. Suppose you have a variable called “A” that currently holds 2. Create a variable called “B” that is assigned whatever “A” is plus 1.

7. Reassign “A” to whatever it is times 3.

8. Suppose you see code like this:

   mylist = (2.4,8.3,10.3,4.5)

   Recast “mylist” as a list.

Control Flows (Pay attention to indentation!)

9. Assume you have used the random integer number generator with the following line of code

   num = np.random.randint(2),

   to assign either a 0 or a 1 to variable num. Then write code that prints “Heads” if num is 0 or “Tails” if the num is 1.

10. Suppose you have the following list:

    mylist = [10.0, 11.4, 12.3, 5.6, 4.8, 10.2, 8.8, 7.6, 12.4]

    a. Use a for-loop to iterate through “mylist” and keep track of the number of times that an element is greater than 10. Assign this number to a variable named “count”. *Hint: You may need to initialize “count” before the for-loop then reassign it inside an if-else statement that is nested inside the for-loop.*

    b. Alternatively, use np.sum() and a list comprehension to do the same thing.

11. Suppose you have a variable “fnames” that contains the names of a bunch of files you want to open. However, you don’t know if some of the files are corrupt and might throw an error. In this case, you might want to use a try-except block to attempt opening the files. Pretend that a function called “readfile” already exists to read these files. In this case, you would open the file with the following line assuming “I” is one of the elements in “fnames”:

    readfile(i)

    Write a try-except block inside a for-loop that iterates through each file, attempts to open it using the arbitrary function “readfile” with the file name as input, and prints an error message of your choice if the readfile function fails on that file.