h03: Input/Output, Data Mutation and Related Topics

Assigned: Tuesday, April 16th, 2019
Due: Tuesday, April 23rd, 2019
Points: 100

- You may collaborate on this homework with AT MOST one person, an optional "homework buddy". MAY ONLY BE TURNED IN THE LECTURE LISTED ABOVE AS THE DUE DATE. There is NO MAKEUP for missed assignments; in place of that, we drop the single lowest score (if you a zero, that is the lowest score.)
- IMPORTANT: When submitting this homework:
  - **DO NOT USE STAPLES**
  - WRITE YOUR **NAME ON EACH PAGE** IN THE SPACE PROVIDED
  - USE DARK INK PENS – PLEASE **DO NOT USE PENCIL**
  - PRINT THIS HOMEWORK **DOUBLE-SIDED** PLEASE!
- **REMEMBER**: If you use code/techniques we have not learned in class, you will **NOT** get credit!

READING ASSIGNMENT: Read Chapter 3.1, 3.4, and 3.5 in Perkovic, review your lecture slides/notes. Then complete these problems.

1. (25 pts) Answer these questions about the built-in `print()` function:

   a) (3 pts) How can I print a newline inside of a `print()` function?

   Add " \n " in order to print a new line inside a print () function.

   b) (3 pts) How can I print the double-quotations (") character inside of a `print()` function?

   Use a backslash with double quotes after the backslash to print double quotes in a print function.

   c) (4 pts) What would this print out in IDLE (*exactly*)? Hint: try it out! ☺
   ```
   >>> num = 42
   >>> print("Welcome, number \"", num, "\"!")
   ```

   Welcome, number "42"!

   d) (5 pts) What would change in the output if we changed the **print()** statement above to the following? Again, show the *exact* output.
   ```
   >>> print("Welcome, number \"", num, "\"!", sep="...")
   ```

   Welcome, number "42"!

   e) (10 pts) Explain what happened in part(d) (i.e. why did you see a difference? What was it due to?)

   An additional ellipses was added within the quotation marks surrounding 42 because of the adding of "sep ="..." Because of then, the value before will he added within the quotation marks to include "..."

2. (16 pts) IF they are needed, show where all indent(s) in the code below need to happen (_put arrows at the start of the lines that need indenting_). Hint: start by running this code as it is written.

```
def printName(name):
--> name = "The name I will print out is: " + name
--> print(name)
   def goHome():
--> print ("Ok, we're done here!")
--> n = input("What is your name? ")
--> printName(n)
   goHome()
```

3. (4 pts) What do each of these print out?
   a) (2 pts) print(list(range(9, 14)))

   [9, 10, 11, 12, 13]

   b) (2 pts) print(list(range(1, 17, 5)))

   [1, 6, 11, 16]

4. (15 pts) Write a function in Python, called **PrintThem(ls)** that takes a *list* variable **ls** as argument. This list could have any number of items in it (the function does not know ahead of time). The function then prints out the sum, the maximum, the minimum, and the average of all elements in the list, each on separate lines and each with an announcement, exactly like this sample run below. The function prints these things without returning any value. You can safely assume that the variable **ls** will always just include integer numbers (you don't have to check for that).

```
>>> x = [5, 2, 10, 8]
>>> PrintThem(x)
The sum is: 25
The max is: 10
The min is: 2
The average is: 6.25
```

```
>>> def PrintThem(ls):
        print (sum(ls))
        print (max(ls))
        print (min (ls))
        print (sum(x) / len(x))
```

5. (12 pts) We talked about a function called **swap()** in lecture (see code below). Explain why it does not work the way it is intended to?

```
def swap(a,b):
    temp = a
    a = b
    b = temp
```

This does not work because integers are non-mutable. This CAN be done with a list, however.

~~or If the swap~~

"Immutable are the objects whose state can't be changed once the object is created"

— Math?

6. (8 pts) Given the following values, what is the output of these **print()** statements? 2 pts for each part.

```
x = 5
y = 3
z = False
```

a) print((x > y or y == 1) and (x <= 5))

True

b) print(z or (x <= y))

False

c) print((y + 3) < x or not(z))

True

d) print(not(x != 5) and y != x or z)

True

7. (20 pts) Write a function definition for myFunc(L, target), which takes 2 input arguments: a list L, which is assumed to be made up of all integers (the function does not need to check for that), and an integer target (again, there's no need to check to see if target is actually an integer).

If the number target is found in the list L <u>and</u> the largest number in L is larger than target, then the function must return a Boolean value of True, otherwise it has to return a Boolean value of False.

Plan out this code carefully! It is highly recommended to try it out on Python IDLE too! ☺

```
>>> def myFunc (L, target):
        y = L
        x = target
        if x in y and max(y) > x:
            print ("True")
        else:
            print ("False")
```

4