Prelab 2

We encourage you to work together on the Pre Lab. The Pre Lab is not graded but will help you prepare for your lab session. In the Pre Lab, you may find questions to answer. We do not require you to provide us the answers, but we do recommend you to try to answer these questions. If you have any questions on the material in the pre lab, first check the book and recitation slides, if you do not find your answer please email your recitation TA or the course instructors.

Section 1: Importing Libraries

We can import python libraries in 2 ways.

• **import <library name here>**. This statements imports a specific module or package from the library that we need to use. When using this import method the library name will always be appear with the package.

For the purpose of this lab we need to use the math library. An example of that can be.

```python
import math ← imports the math library
x = 25
print(math.sqrt(x))
```

The following code produces the result as

```bash
>>>5.0
```

In this we see that **math** is the library name and **sqrt** is the package we are want to use. The . connects the two.

• **from <library name> import ***

* in programming generally means everything. So the statement can be read as, from <library name> import everything. Using this method produces the same result but with a little difference in the code.

```python
from math import * ← imports the math library
x = 25
print(sqrt(x))
```

The following code produces the result as

```bash
>>>5.0
```

As you can see in the code above, we have the same result but we did not use the word **math** in front of **sqrt**. That is because we imported all the packages in the math library and can use them directly.
Section 2: Nested if-else conditions

While you program, there may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, you can use the nested if construct.

Simple If-else statement

```python
def main():
    x = int(input("Enter an integer: "))
    if(x == 10):
        print("The value is 10")
    else:
        print("The value is not 10")
main()
```

Nested if-else

In the code above, if we input the value as 10, the program prints “The value is 10”. Otherwise it prints “The value is not 10”. Now suppose we want to make another decision if the value is 10. If the value is 10, the program should take another input from the user. If the value of second input is 5, it prints “The value is 5”. Otherwise it prints “The value is not 5”. This can be done in the following way.

```python
def main():
    x = int(input("Enter an integer: "))
    if(x == 10):
        y = int(input("Enter second input: "))
        if(y == 5):
            print("The value is 5.")
        else:
            print(\"the value is not 5.\")
    else:
        print("The value is not 10 and did not enter the nested if.\")
main()
```

Output if we enter value as 10 the first time.

Enter an integer: 10
Enter second input: 5
The value is 5.

Output if we enter any other value than 10 the first time.

Enter an integer: 8
The value is not 10 and did not enter the nested if.

The above code uses nested if-else. So the format for using nested if condition looks like.

```python
if expression1:
    statement(s)
```
if expression2:
    statement(s)
else:
    statement(s)
else:
    statement(s)

**Section 3: Loop structure: for**

This exercise shows the basics of the **for** loop structure. In this structure a variable takes values in an specified range. For this, the **for** loop structure uses the function `range()`. The function `range()` is complex and allow you to specify several series of numbers. For example, it allows you to specify continuous series such as 0,1,2,3... or 5,6,7,8...; series of odd numbers such as 1,3,5,... or 9,11,13,15...; series of even numbers such as 2,4,6,8... or 19,12,14,16...; and so on. Here we will focus on the first case. We will learn how write codes where the variable of the **for** loop takes values from 0 to n-1, for a given n.

**Example 1**

```python
def main():
    n = 5
    for i in range(n):
        print(i)

main()
```

**Output of example 1**

```bash
>>> ============= RESTART =============
>>> 0
1
2
3
4
>>> 
```

There are a few things to highlight from this example:

1. The line `for i in range(n):` is equivalent to `for i in range(5):`
2. The `range(5)` function generates values from 0 to 4 (not from 1 to 5). VERY IMPORTANT!
3. You can do any operation (+,-,* and more) over the variable `i` to alter the value to be printed (see example below).

**Example 2**

```python
def main():
    n = 5
```
for i in range(n):
    print(i + 2)

main()

Output of example 2

>>> ========================= RESTART
================================
>>> 2
3
4
5
6
>>>

Section 4: Printing more complex lines

It is very important to master how to combine strings with other types of values in the print statement. There are several ways to achieve this in Python. One of the easiest strategies is to cast the non-string variables to string for then concatenate all the variables. See examples below.

Example 1: Concatenating two strings

def main():
    str1 = "Hello"
    str2 = "World!"
    str3 = str1 + " " + str2
    print(str3)

main()

Output of example 1

>>> Hello World!
>>>

Note: Please notice the addition of the string " " (blank space) between str1 and str2.

Example 2: Concatenating one string to an integer

def main():
    str1 = "Value = " # This is a string. Notice it includes a blank space at the end.
    value = 5 # This is an integer.
    str2 = str1 + str(value) # This is a string. The statement "str(value)" convert the integer 5 to string '5'.
```python
print(str2)
main()
```

Output of example 2

```python
>>> ==========================
RESTART

>>> Value = 5
```