Recitation 12 - Python (Loops, Conditions, Functions)

CSCI 1300 Fall 2017

Instructor: Dr. David Knox

Due: Sunday 6pm

Python

From this point on in the semester we will be working with Python. A great resource for learning Python is, "Learning Python the hardway": **https://learnpythonthehardway.org/book** You will have to save your files as **.py** instead of .cpp when working with python.

You can use Geany as your IDE for writing python programs or just use a text editor and a terminal window to compile your program.

Indentation

In Python the delimiter is a indentation of the code itself. No curly braces will be used to mark where the code starts and stops and for any loops, if-else statements, or functions there is not an explicit begin or end without the indentation. There are no explicit braces, brackets, or keywords. This means that whitespace is significant, and must be consistent.

"Code blocks" or sections of code are defined by their indentation. Code blocks means sections of code for functions, if statements, for loops, while loops, and so forth. Indenting starts a block and is ended when the indentation moves back a tab.

Pound/Hash (#) is used for comments in Python.

For example:

```
Block 1             #statements in Block 1 are Executed
    Block 2         #statements in Block 2 are Executed
        Block 3     #statements in Block 3 are Executed
    Block 2         #Remaining statements in Block 2 are Executed
Block 1             #Remaining statements in Block 1 are Executed
```

Functions in Python

A function has a few key components: the name, the parameters, the body and the return value. Let's take a look at a simple example of a Python function.

Name **Parameters**

↓ ↓

def adder (x, y):

z = x+y ← **Body**

return z ← **Return**

Function Example: This example defines a function called adderTwo that takes two parameters and returns their sum. The adderTwo function is called from another function called main.

```
def adderTwo(numOne, numTwo):
     print(numOne, numTwo)
     numOne = numOne + numTwo
     print(numOne)
     return numOne
def main():
     X = 3
     Y = 4
     print(X,Y)
     Z = adderTwo(X,Y)
     print(X, Y, Z)
```

Loops in Python: For and While

Loops allow us to run a chunk of code multiple times. If we know how many times we'll need to run the code, we typically use a for loop. When we don't know how many times we need to run the code, we usually use a while loop.

For Loops

The basic structure of a for loop in Python looks like this:

```
for i in range(startNum, endNum):
        #do something using i
```

Here, i is the dummy variable and range(startNum, endNum) is the iterable. The dummy variable takes on the values of the iterable, beginning with startNum and up to, **but not including**, endNum. Each loop, i will increment by 1. For example, if we enter the following code:

```
for i in range(0, 5):
        print(i)
```

We should see the following output:

```
0
1
2
3
4
```

Alternatively, we can define range with a single parameter: range(endNum) By default, the dummy variable will begin at 0 and iterate up to, but not including, endNum.

If we want to have i increase by some number other than 1, we can specify this number as the 3rd parameter in range:

```
for i in range(0, 5, 2):
        print(i)
```

The output:

```
0
2
4
```

Of course we can do things far more complex than just printing a single variable. Try writing a loop that outputs the even numbers between 0 and 10.

While loops

A while loop in python uses the following syntax:

```
while(condition):
        #do something
```

As long as condition is true, the code within the while loop will run. Consider the following code:

```
userChoice = 1
while(userChoice):
     userChoice = int(input("Do you want to see this question
     again? Press 0 for no, any other number for yes"))
```

Entering '0' will terminate the loop, but any other number will cause the loop to run again.

Note how we have to initialize the condition before the loop starts. Setting userChoice equal to 1 ensures that the while loop will run at least once.

Also note that we must "cast" the input to type int. By default, input() returns a string value. A string value will always evaluate as true. Consequently, we must turn it into an integer to have its "truth" evaluated within the while condition.

Converting For into While

It's possible to use a while loop instead of a for loop. Let's rewrite the second for loop example from above as a while loop:

```
i = 0
while i < 5:
      print(i)
      i = i + 2
```

This accomplishes the same as output as above, but with more lines of code. Notice how you must manually initialize i to equal 0 and then manually increment i by 2.

Debugging Tip: Inserting print statements into your loops is a quick way to debug your code if something isn't working.

Recitation Activity (due Sunday 6PM)

There are three questions on Moodle to be completed by **Sunday at 6 PM.**

The first question asks you to write a function that prints out whatever is passed in as arguments.

The second question asks you to write a function that takes in an integer as a parameter and prints out whether the number is even or odd.

The third question asks you to write a function that loops from 0 to the number 10, printing the number or a word at each iteration. If the current number is divisible by 3, print the word "Fizz", if it is divisible by 5, print "Buzz", if it is 0 or otherwise, print the number itself.