cs002 Lab 8
Python Part II

Assigned: November 28, 2016 **Due: December 7, 2016**

Introduction

Today you will be learning about some of the most powerful components of programming - functions, loops, and variable scope. Nearly every electronic device on the planet will use these ideas, and even the most complicated bits of code rely on these building blocks to work.

Lab Goals

- Understand and write loops in Python
- Understand and write functions in Python
- Understand and implement different variable scopes
- Understand how to use and import libraries in Python

Lab Assignment

Comments

Recall from the Javascript assignment and lab that **comments** were used to explain how parts of our code worked. The text written in the comments were not part of the runnable code, but were just for readability purposes.

In Javascript, to leave a comment we started the line of code with two slashes:

```
// This is a Javascript comment
```

In **Python**, to leave a comment, we start the line of code with a hash:

```
# This is what a comment looks like in Python
```

While Loops

A **while loop** will execute a piece of code until a certain condition is met. If you want a piece of code to repeat several times, you can use a while loop.

Here is an example: the following code will print the words “Where is my Super SUIT?!" 10 times:

cs002 Lab 8
Python Part II

```
count = 0
while count < 10:
    print "Where is my Super SUIT?!"
    count = count + 1
```

Let's dissect this piece of code:

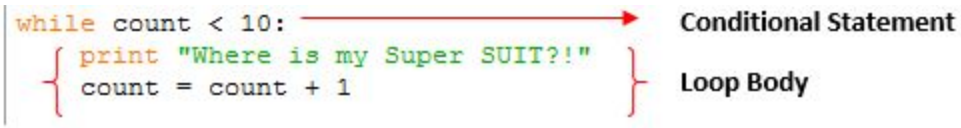
- The first line is just a simple assignment as you've seen before, giving `count` the value `0`.
- The next line is the *conditional* part of the while loop. Every while loop starts with the word while followed by a conditional statement and then a colon.

  ```
  while {conditional_statement} :
  ```

  The conditional statement will work exactly as the `if` statement conditionals from last lab- they must have a mathematical condition that can be evaluated as `true` or `false`. The loop will repeat while the condition remains true, and continue beyond the loop body when it finally becomes false.

Your conditional statement MUST eventually evaluates to false,or else you will be trapped within an infinite loop. (for example, in the above code if the conditional statement had been `count < -1`, the code would never end)

In Python, it is essential to indent each line that is part of the loop body (the lines beneath the conditional statement which will be executed during each loop). Our loop body consists of two lines of code: a print line and an increment line.



- The print line prints out the string "Where is my Super SUIT?!" during every iteration of the loop.
- The increment line modifies our variable `count` during each iteration of the loop, in this case increasing it by one. It is essential to have some line in the loop body that modifies the conditional statement, so as to avoid an infinite loop.

While the code is running, the loop will:

- start with `count = 0`
- check that `count` is less than 10 (it is so the loop continues)
- print the string Where is my Super Suit?!
- increment the variable `count` by 1

During the second iteration of the loop, `count` is now equal to 1 which is still less than 10, so the string is printed again, and `count` is again incremented by 1. This process will

continue until `count` has been incremented to 10. When `count = 10` it is no longer less than 10, so the condition will evaluate to false and the loop will exit without printing.

For Loops

Another way of looping is by using a for loop. Unlike while loops which go on while a certain condition is true, for loops go for a set number of times. **For loops have a definite, exact number of times they will loop that is known to the programmer**. In Python, for loops iterate over the items in a sequence in the order that the items appear in the sequence.

Here is an example:

```
hero_names = ["Frozone", "Elastigirl", "Dash"]
for name in hero_names:
    print name, len(name)
```

This code will output:

```
Frozone 7
Elastigirl 10
Dash 4
```

Looking at the code line by line the first line creates a list named `hero_names` that contains the strings Frozone, Elastigirl, Mr. Incredible in that order. The next line contains the actual FOR loop. Every FOR loop has the same basic syntax:

```
for {some placeholder variable} in {collection to be iterated over}:
```

In our case, we have a placeholder variable (`name`) and a list (`hero_names`) that was created before the loop. The body of the loop simply compounds two functions, `print` and `len()`, together by separating the two functions with a comma.

- The `print` function simply prints the current element of the collection represented by the placeholder variable `name`, during that current loop iteration
- The `len()` function returns the length of the element placed in between the parentheses, so in the case of the string 'Frozone', `len` would return 7, because Frozone has 7 letters.

After printing out "Frozone 7", the code will continue running, this time, `name` will represent the second element in the list, "Elastigirl". After that runs, `name` will represent

"Dash". After the loop iterates over every element in the list `hero_names`, it stops running.

Functions

Functions are used to divide the tasks of a program into small manageable pieces. Each function should do something unique. In order to figure out which tasks of a program should be defined as their own functions, you should think about the number of times that task needs to be done. If a certain specific task needs to be done multiple times then it should be defined as a function. As a rule of thumb, you should define functions for tasks that have to be done repeatedly. If, on the other hand, there is a task that will only be done once, then it should be written directly in the main program.

We can create a function that adds three numbers together.

```
def add3(x, y, z):
    print x + y + z
```

Now we can "call" the function we defined by writing the code below **outside** your function (on the same indentation level as def):

```
add(3, 5, 7)
```

Many times it is useful to have a function **return** some data instead of simply printing it. For example, perhaps you want to store the sum of three numbers in order to use that value later.

```
def add3(x, y, z):
    return x + y + z


def average(a, b, c):
    total = add3(a, b, c)
    avg = total/3
    return avg
```

This function `average` takes in three parameters (inputs) a, b, and c (the numbers you want to average), and calculates the average of those numbers.

cs002 Lab 8
Python Part II

Notice that when if we call `average(30,6, 10)` , nothing happens. The result is still computed, it just doesn't print out.

If we do:

```
print average(30, 6, 10)
```

15 should be printed out.

Local vs. Global Variables

Inside the function average, there are variables a, b, and c. These are **local variables,** so they are not known outside of the function in your main program. Within the scope of a given function, variable names can be whatever you want them to be and are not related to the variables outside the function. **Global variables** are defined outside the functions.

Importing Libraries

In Python you can preface your program with **import** statements which allow you to use predefined **functions** already written by someone else. For example, one library frequently used is the **random** library. To use this library, add the following line above all of your code in your Python file:

```
import random
```

Now, you can use any of the functions found in the library `random`. Documentation for the `random` library can be found here.

For example, if I wanted to print a random float in the range [0.0, 1.0), I could use the random function:

```
import random

print(random.random())
```

Tasks:

To show understanding of this lab, we wish for you to create music playlist for the Incredible family to listen to while saving the world, and write two functions that use and manipulate the collection.

Included for use in the task is a simple list of ten songs. Please use this list!

cs002 Lab 8
Python Part II

- "Love in this Club"
- "Right Round"
- "Heartbreaker"
- "Three Little Birds"
- "Run This Town"
- "Dance Dance"
- "Smooth"
- "Party in the USA"
- "Hero"
- "Hotline Bling"

Make sure that you write the songs as they appear. For example "Hero" should be written as "Hero" not "Hero " or "Hero \n" as this may affect what the len() function returns for that string

Task 1

Create a list of the above 10 songs. Please put them in the order they appear above.

Task 2

Write a function that takes in a list as a parameter and prints each item in the list. Call this function and use your playlist as the parameter. It is not necessary to modify the list in anyway, so there is no need to copy the list. It is **mandatory** that you simply print the current string element being looked at in the loop, rather than printing the list object itself. This function does not need to return anything.

Task 3

Write another function that will cycle through the your list and put all of the songs that have a length less than five into a new list. In order for this to work, your for loop condition should be :

```
for y in L:
```

First print the original ten songs, then call your second function that puts songs with titles that are shorter than five characters into a new list. Have this function return the new list that you created, then call your first function again to print the songs in your new list. Be sure that you are passing the list to the functions as necessary.

The output after the new list is created should simply print a list with the song "Hero".

cs002 Lab 8
Python Part II

Task 4

Explain how the randint function works from the random library. More specifically, **print** out the function's output and then in a **comment** explain what the function is doing.

Check-off Requirements

- Function that prints the list of songs
- Function that removes songs with length greater than 5
- Call the functions so that the original list is printed, then the altered list is printed after songs are removed
- Show and explain how the randint function works.

Submission

To submit this lab, please raise your hand so a TA can come check your work. Make sure you tell the TA to check you off their list, or else you will not receive credit. If you do not finish your lab in the allotted time, please either come to office hours or another lab section to finish your work. If you are unable to complete this lab due to sickness or injury, please contact BOTH Don and the cs002 TAs via email. Please contact the cs002 TA's if you have any further questions.

cs002 TA Email: cs002tas@cs.brown.edu
Don Stanford's Email: don.stanford@gmail.com