Programming Bootcamp 2013: Lab 3

Sarah Middleton

Last Updated: June 7, 2015

Name:
If you would like to receive points for your work, copy and paste your code into this document, save it, and send it to **sarahmid@mail.med.upenn.edu** with "Bootcamp Lab3" as the subject line **before 9am on 5/12**.

1 Guess the output: loop practice (1pt)

For each of the following statements, first guess what the output will be, and then run the code yourself. You can use the interpreter if you want. If you get a result you didn't expect, make sure you understand why. **Watch out for endless loops, and remember you can terminate any terminal program with Ctrl+C.**

Code	Predicted output	Actual output
`for i in range(1,10,2):` `  print i`		
`for i in range(5,1,-1):` `  print i`		

`count = 0`
`while (count < 5):`
`  print count`		

`count = 0`
`while (count < 5):`
`  print count`
`  count = count + 1`		

`total = 0`
`for i in range(4):`
`  total = total + i`
`print total`		

`name = "Mits"`
`for letter in name:`
`  print letter`		

`name = "Wilfred"`
`newName = ""`
`for letter in name:`
`  newName = newName + letter`
`print newName`		

`name = "Wilfred"`
`newName = ""`
`for letter in name:`
`  newName = letter + newName`
`print newName`		

`x = "C"`
`if x == "A" or "B":`
`  print "yes"`
`else:`
`  print "no"`		

`x = "C"`
`if (x == "A") or (x == "B"):`
`  print "yes"`
`else:`
`  print "no"`		

Surprised by the last two? I didn't stress this very much last time, but it's important to note that when you compare a variable against multiple things, you only compare it to one thing at a time. Although it makes sense in English to say, is x equal to A or B?, in Python you must write: `((x == "A") or (x == "B"))` to accomplish this. The same goes for e.g. `((x > 5) and (x < 10))` and anything along those lines.

2 Simple loop practice (4pts)

Write code to accomplish each of the following tasks using a for loop or a while loop. Choose whichever type of loop you want for each problem (you can try both, if you want extra practice).

(a) **(1 pt)** Print the integers between 3 and 35, inclusive.

(b) **(1 pt)** Print the positive integers less than 100 that are multiples of 7.

(c) **(1 pt)** Starting with x = 1, double x until it's greater than 1000. Print each value of x as you go along.

(d) **(1 pt)** Print each character of the string `"supercalifragilisticexpialidocious"` on a separate line.

3 File reading & processing (6pts)

For this problem, use the file `sequences.txt` provided on Piazza. This file contains several DNA sequences of different lengths. You can assume each sequence is on a separate line.

Note: For this and for all following problems, try to find ways to verify your answers are correct. For example, spot-check some of the output against the original file to see if it's what

you expect. Pretend that you are writing this code for your actual research! I have provide some of the expected output where reasonable.

(a) **(1 pt)** Using a loop, read in each sequence from the file and print it to the terminal screen. Make sure to remove any newline characters (`\n`) while reading in the data.

(b) **(1 pt)** Instead of printing the sequences themselves, output the length of each sequence to the terminal screen. At the end, skip a line and then print the average length of the sequences. (You should get 77.56 as the average.) *(Hint: use the concept of an "accumulator" variable to help with computing the average)*

(c) **(2 pt)** Instead of printing lengths, print the GC content of each sequence (GC content is the number of G's and C's in a DNA sequence divided by the total sequence length). At the end skip a line and print the average GC content. (You should get $\sim$0.4877 as the average.)

(d) **(2 pt)** Convert each sequence to its reverse complement. This means changing each nucleotide to its complement (A$\rightarrow$T, T$\rightarrow$A, G$\rightarrow$C, C$\rightarrow$G) and reversing the entire sequence. *(Hint: we've already touched on everything you need to know to do this – see problem 1 for some clues)*

4 A guessing game (2pts)

Write a script that plays a number guessing game that loops until the user gets the number right. First, have your program generate a random integer between 1 and 20 (the "secret number"). Then prompt the user "Guess a number between 1 and 20 (enter 0 to quit): ". Read in their answer (recall that you'll need to convert the user input to an int). If they entered 0, print "I win!" and end the program. If the guess is correct, print "You got it!" and end the program. If the guess is larger than the secret number, print "Lower..." and allow the user to keep guessing. If the guess is lower than the secret number, print "Higher..." and allow the user to keep guessing.

Tip: It will be easier for you to test your program if you initially set the secret number to be a number of your choosing instead of something random. Then when you run the program, you will know if the responses are correct. Make sure to test all possibilities (higher, lower, equal, zero).

5 Family simulation (5pts)

Use a simulation to determine the average number of children in a family if all families had children until they had a girl (and then stopped). Assume equal probability of girls and boys and use a random generator to simulate each birth. Simulate 10,000 families and output the average number of children they had. Your answer should be close to 2.

Note: I'm purposely not giving any step by step instructions here because I'd like you to practice translating a word problem into code. It may help if you try break down the problem into smaller parts first. For example, see if you can simulate just one family first, then once that's working, add code to make it run through 10000 families. Alternatively, you might feel more comfortable writing code to print "hi" 10000 times, and then replace "print hi" with code for simulating each family. Do whatever makes the problem feel more manageable to you. Learning to break down big problems into smaller parts like this is an essential skill for programming, so I encourage you to work through this. If you get it, then I think it's safe to say you've mastered the material in this lesson and are well on your way to becoming a true programmer!