Programming Bootcamp 2015: Lab 2

Sarah Middleton

Last Updated: June 4, 2015

Name: ANSWER KEY

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 Lab2" as the subject line **before 9am on 5/9**.

1 Guess the output: conditionals 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.

The following variables will be used **(add these to your program before running)**:

```
a = True
b = False
x = 2
y = -2
cat = "Mittens"
```

Code	Predicted output	Actual output
`print a`		True
`print (not a)`		False
`print (a == b)`		False
`print (a != b)`		True
`print (x == y)`		False
`print (x > y)`		True
`print (x = 2)`		SyntaxError (should be ==)

`print (a and b)`		False
`print (a and not b)`		True
`print (a or b)`		True
`print (not b or a)`		True
`print not (b or a)`		False
`print (not b) or a`		True
`print (not b and a)`		True
`print not (b and a)`		True
`print (not b) and a`		True
`print (x == abs(y))`		True
`print len(cat)`		7
`print cat + x`		TypeError
`print cat + str(x)`		Mittens2
`print float(x)`		2.0
`print ("i" in cat)`		True
`print ("g" in cat)`		False
`print ("Mit" in cat)`		True

What is `in`? This is just another Python operator that you can use in your conditionals. As you may have guessed, `(x in y)` evaluates to `True` if `x` is in `y`. This can be used to check if a string is contained in another string (e.g. `"Mit" in "Mittens"`).

2 Guess the output: control flow practice (1pt)

For each of the following code snippets, 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.

*Use the pre-defined variables from question 1.

Code	Predicted output	Actual output

`if (x % 2) == 0:`
`  print "x is even"`
`else:`
`  print "x is odd"`
		x is even

`if (x - 4*y) < 0:`
`  print "Invalid!"`
`else:`
`  print "Valid"`
		Valid

`if cat == "Mittens":`
`  print "Awwww"`
`else:`
`  print "Get lost, cat"`
		Awwww

`if "Mit" in cat:`
`  print "Hey Mits!"`
`else:`
`  print "Where's Mits?"`
		Hey Mits!

3 Quadratic formula: checking for negative roots (1pt)

Recall that when calculating the quadratic formula, you will get an error if $b^2 - 4ac$ is negative, since you can't take the square root of a negative number. Edit your quadratic formula program from Lab1 so that it checks for this potential error before it occurs. If the error is going to occur, print an informative message saying "non-real answer" or something like that, and do not calculate the values of x. If the error is not going to occur, continue on to calculating and printing the values of x.

```
a = float(raw_input("a = "))
b = float(raw_input("b = "))
c = float(raw_input("c = "))

underRoot = b**2 - 4*a*c
if underRoot < 0:
```

4 Motif checker (2pts)

Prompt the user for a DNA sequence and a motif to search for (these can both can be any string of A/T/G/C's; see below for examples).

(a) **(1 pt)** Find the length of the motif and length of the DNA sequence and make sure the motif is shorter than the sequence. If it is not, print an informative error message.

```
dnaSeq = raw_input("Enter a DNA sequence: ")
motif = raw_input("Enter a motif to search for: ")

if len(motif) > len(dnaSeq):
    print "Error: motif sequence is longer than DNA sequence."
```

(b) **(1 pt)** Adding to your code from part (a): If and only if the motif is shorter than the sequence, go on to check if the motif can be found somewhere within the sequence (*hint: use* `in`). Print a message saying whether it was found or not.

```
dnaSeq = raw_input("Enter a DNA sequence: ")
motif = raw_input("Enter a motif to search for: ")

if len(motif) > len(dnaSeq):
    print "Error: motif sequence is longer than DNA sequence."
else:
```

Make sure to test your code with different examples. Here are some you can try:

Example input:

Sequence: `AGCTAGCCTGCTAGAAATCGATTGGCTAGCAATCTTATTGTGTTCTACG`
Motif: `ATG` (This should pass the check from part (a) but not part (b))

Sequence: `AGCTAGCCTGCTAGAAATCGATTGGCTAGCAATCTTATTGTGTTCTACG`
Motif: `ATCGA` (should pass both checks)

Sequence: `CTAGCC`
Motif: `ATGGCTAGCTA` (code should not pass the check from part (a))

5 Password protection (3pts)

Create a script that prompts the user to guess a password and checks whether the password is correct. In your script, first create a variable called `password` and set it equal to whatever you want the password to be (any string). Then prompt the user to guess the password, and

read their input from the terminal. Check if what they entered matches your password, and print a message saying whether or not the guess was correct.

(a) **(1 pt)** Do this giving the user only one chance to guess.

```
password = "Mittens123"
guess = raw_input("Enter the password: ")

if guess == password:
    print "Correct!"
else:
```

(b) **(2 pt)** Do this giving the user 3 chances to guess. Only prompt for another guess if the previous guess was incorrect. When they guess correctly, print a confirmation message and end the program. If they never guess correctly in 3 tries, print "Access denied!" and end the program.

```
password = "Mittens123"
guess = raw_input("Enter the password: ")

if guess == password:
    print "Correct!"
else:
```

Be sure to test that your program works as expected by running it a few times with different input.

6 Coin flip simulation (4pts)

For this problem, we will simulate a coin flip using the random number generator in the `random` module. Type the following into a script:

```
import random

randNum = random.randint(0,1)
print randNum
```

Save and run this script several times. You should see that you always get either 0 or 1. We will pretend that each time we run `random.randint(0,1)` we are flipping a coin, where 1 represents heads and 0 represents tails.

(a) **(2 pt)** Write a script that "flips a coin" 10 times and counts how many times it comes up heads. At the end, print how many of the flips were heads.

```
import random

headsCount = 0

if random.randint(0,1) == 1:
    headsCount = headsCount + 1
```

(b) **(2 pt)** Change your script so that the coin is now "unfair" (i.e. the chance of heads is not equal to the chance of tails). See if you can make it so the probability of heads is 75%. *Hint: There are several ways to do this, but one simple way involves using a larger range of random ints and assigning heads/tails differently.*

```
import random

headsCount = 0

# random.randint(0,3) gives a 0, 1, 2, or 3
# if it's 0, 1, or 2, call it heads
```