Python

Lab 02: Control Structures

This lab practices the concepts of if/elif/else statements, as well as for and while loops.

Exercise 1: Written Assessment

Part A - Consider the following code (draw a flowchart diagram if it helps):

```
if x>2:
if y>2:
z=x+y
print "z is ",z
else:
print "x is ",x
```

What is the output if:

a. x = 2 and y = 5?
b. x = 3 and y = 1?
c. x = 1 and y = 1?
d. x = 4 and y = 3?

Part B - Suppose we have the following code:

```
z = x+3
if z==1:
y=0
elif z==2:
y=10
elif z==4:
y+=1
else:
y=1
```

a. What is y equal to after the switch statement if x = 3 and y = 5 entering the switch?
b. What if x = 2 and y = 5 at the beginning?

Part C - What does the following code output, and how many times do we run through the *loop body*?

```
i=0
while i < 10:
i+=1
if i%2 == 0:
print i
```

Exercise 2: Even or Odd?
Have the user enter an integer. The program should return whether the integer is even or odd, and if odd whether or not it is divisible by 3. Use `if/else` statements.

```
>>>
Enter an integer: 11
Odd
And is not divisible by 3
>>>
```

Exercise 3: Which is the Least?
Have the user enter three integers. The program should return which integer is the least. This can be done using `if/else` statements but trying doing it using `if/elif/else` statements.

```
>>>
Enter integer a: 4367
Enter integer b: 2346
Enter integer c: 394
c is the least
>>>
```

Evercise 4: Rocket Take Off
Count down the last 60 seconds before a rocket takes off! 60-10 seconds should be count down in increments of 10. 10 – 9 should be count down by one. Use the range() as discussed earlier in session.

```
>>>
60 seconds until the space shuttle takes off!
60
50
40
30
20
10
9
8
7
6
5
4
3
2
1
TAKE OFF!
>>>
```

Why does the program not print zero?

Exercise 5: Cube Root of a Perfect Square
Find the cube root of a perfect square. If the integer input is not a perfect square, the program should return that the number is not a perfect square. The output of the program should look like this:

```
>>>
Enter an integer: 64
The cube root of 64 is 4
>>>

>>>
Enter an integer: 156
156 is not a perfect cube
>>>
```

Exercise 6: First 100 Prime Number
Create a Python file that displays the first 100 prime numbers.
An integer greater than 1 is prime if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime but 4, 6, 8, and 9 are not prime. The output of your program should look like this:

```
>>>
Finding the first 100 prime numbers:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 1
07 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 2
23 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 3
37 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 4
57 461 463 467 479 487 491 499 503 509 521 523 541 547
>>>
```

You need to write a loop and test whether each new number is prime. Declare a variable count to store the number of primes encountered so far. If the number is prime, increment count by 1. When count is greater than 100 exit the loop.

Hint: To test whether a number is prime, check if the number is divisible by 2, 3, 4, up to number/2. If a divisor is found, the number is not prime. For example, for the number 17, you need to test whether each of 2, 3, 4, 5, 6, 7, and 8 are divisors of 17. Since none are divisors, 17 is prime. If a number is not prime, once you find the first divisor, you should not keep checking for additional divisors

Challenge Exercise: Successive Guess v. Bisection Search
Solve for the square root of an integer using the successive guessing method as well as bisection search. Compare the answers from the two methods as well as the number of guesses each method requires to reach the answer. The output of the program should look like the following:

```
>>>
SUCCESSIVE GUESSING METHOD
4.99899999998 is close to square root of 25
number of guesses was 499900
BISECTION METHOD:
5.00030517578 is close to square root of 25
number of guesses was 13
>>>
```

```
>>>
Failed on square root of -25
number of guesses was 0
```

Test your program with 25 and -25.

This code doesn't work for 0 < x < 1 or x < 0 -- why?
How could you change this code to work for x in (0,1) range?
How could you make this work for negative numbers?

Challenge Exercise: Number Encryption

Part A - Suppose we want to send some numbers over a phone line, but the phone line is tapped. We want to use some (very simple) encryption to stop them from stealing our information.

One quick and easy way to do this is to reverse the digits of our number. For example, 12345 becomes 54321. Think about how we can use %10 in a loop to get the last digit, and how we can make the new number using a loop. Use only basic control structures and operators (don't use String methods).
Code a small program that will take in a number and print out the encrypted number.

Part B - To make our encryption even better, we can take each digit *d* of our number, and replace it with ((*d* + 7) mod 10) before we make our new number. (Note that mod is the same as %).

For example, if we encrypt 12345:
12345 ==> 89012 (from the digit change) ==> 21098 (from reversing it)
1532972 ==> 8209649 ==> 9469028
Add this code to problem 1 so that both versions of the encryption are displayed.