MIT AITI
Python Software Development

Lab 03: Control Structures

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

Part I: Written Exercises

1. 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?

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

3. 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
```

4. How about this version of the code?

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

Part II: Programming Control Structures

Create a new python file called UsingControlStructures.py. We will be checking this python file, so be sure that everything works. There will be instructions below to mark the different questions on this part of the lab.

Copy the following code into your file:

```
theInput = raw_input("Enter an integer: ")
#Your code here
```

This code waits for the user to type an integer and press the "Enter" key, then returns what they entered as an integer. In the code we store the value in a variable called `theInput`.

5. Now insert code into the code above so that the program prints "even" if the input integer is even and "odd" if it is odd.

   For problem 6-9, use the same Python file as the previous problems. Separate the output for each of the following problem problems by printing
   "---------" to the screen.

6. Declare and initialize variables representing:
   a. The age people start primary school, example: `primarySchoolAge = 4;`
   b. The legal voting age.
   c. The age you can become president.
   d. The official retirement age.
   e. A person's age. Do this by using:

      ```
      personsAge = input("Enger an age: ")
      ```

      Again, this just waits for the user to enter a number and then stores it in some variable.

In the current file, use `if-elif` statements to print:

f. "Too young." if the person is too young for school.
g. "Remember to vote" if the person is old enough to vote.
h. "Vote for me" if the person is old enough to be president and "You can't be president" if they are not.
i. "Too old." if the person is old enough to retire.

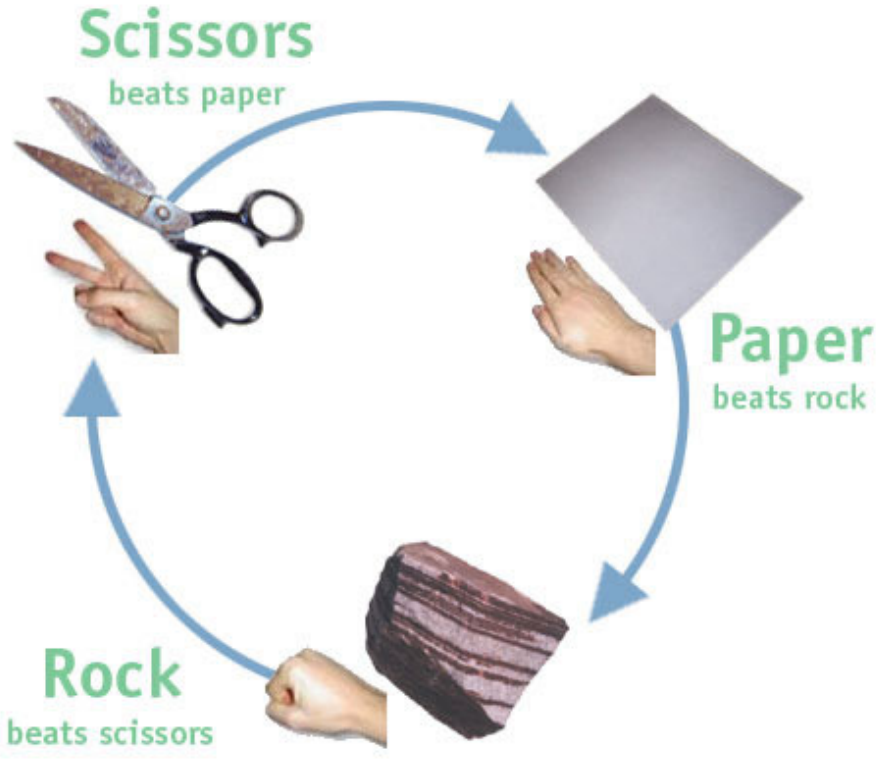
7. Write a `for` or `while` loop that prints out all the multiples of 3 down from 40 to 0 in decreasing order. That is, 39, 36, 33, …, 3, 0.

8. Write a loop that prints out all numbers between 6 and 30 that are **not** divisible by 2, 3, *or* 5.

9. Using a while loop, find the smallest positive integer *n* such that 79**n* has a remainder of 1 when divided by 97.

Part III: Rock Paper Scissors

In this exercise, we are going to practice using the if statement. We are going to write a small program that will as the user for the choice player 1 and 2 made and will print out the result of the game. Here are the rules:



1. First create a truth table for all the possible choices for player 1 and 2 and the outcome of the game, e.g.

Player 1	Player 2	Outcome
Rock	Rock	Tie
Rock	Scissors	Player 1

This should help you with the next part.

2. Create a file **rsp.py** that will generate the outcome of the rock, scissors, paper game. The program should work as follows:

```
Player 1? rock
Player 2? scissors
Player 1 wins.
```

The only valid inputs are rock, paper, and scissors. If the user enters anything else, your program should output "This is not a valid object selection". Use the truth table you created to help with creating the conditions for your if statement.

Note If you have a long condition in your if statement and you want to split it into multiple lines, you would want to enclose the entire expression in parenthesis, e.g.

```
if (player1 == 'rock' and
     player2 == 'scissors'):
     print 'Player 1 wins.'
```

Part IV: Buggy Loop

Consider the following program:

```
n = 10
i = 10


while i > 0:
   print i
   if i % 2 == 0:
      i = i / 2
   else:
      i = i + 1
```

1. Draw a table that shows the value of the variables n and i during the execution of the program. Your table should contain two columns (one for each variable) and one row for each iteration. For each row in the table, write down the values of the variables as they would be at the line containing the print statement.

2. What is problematic about this program? Suggest one way to improve its behavior.

Part V: Practice with While Loops

Write a program that will ask the user to enter a number that is divisible by 2. If the user enters a number that is not divisible by 2, the program will print out a message and then will ask the user to enter a number again. Otherwise, it will congratulate the number and stop. Save your program in a file called **loops.py**. Here is an example of what the program should do:

```
Enter a number divisible by 2: 11
The number 11 is not divisible by 2.
Enter a number divisible by 2: 6
Congratulations! 6 is divisible by 2.
```

Part VI: Secret Messages

The goal of this exercise is to write a cyclic cipher to encrypt messages. This type of cipher was used by Julius Ceasar to communicate with his generals. It is very simple to generate but it can actually be easily broken and does not provide the security one would hope for.

The key idea behind the Ceasar cipher is to replace each letter by a letter some fixed number of positions down the alphabet. For example, if we want to create a cipher shifting by 3, you will get the following mapping:

```
Plain:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:  DEFGHIJKLMNOPQRSTUVWXYZABC
```

To be able to generate the cipher above, we need to understand a little bit about how text is represented inside the computer. Each character has a numerical value and one of the standard encodings is ASCII (American Standard Code for Information Interchange). It is a mapping between the numerical value and the character graphic. For example, the ascii value of 'A' is 65 and the ascii value of 'a' is 97. To conver between the ascii code and the character value in Python, you can use the following code:

```
letter = 'a'
# converts a letter to ascii code
ascii_code = ord(letter)

# convers ascii code to a letter
letter_res = chr(ascii_code)

print ascii_code, letter_res
```

Start small. Do not try to implement the entire program at once. Break the program into parts as follows:

1. Create a file called **cipher.py**. Start your program by asking the user for a phrase to encode and the shift value. Then create a new string that contains the original phrase value using a for loop as follows:

```
encoded_phrase = ''

for c in phrase:
```

```
encoded_phrase = encoded_phrase + c
```

2. Now modify the program above to replace all the alphabetic characters with 'x'. For example:

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is:  Xxxxxx! Xxxxxx!
```

We are going to apply the cipher only to the alphabetic characters and we will ignore the others.

3. Now modify your code, so that it produces the encoded string using the cyclic cipher with the shift value entered by the user. Let's see how one might do a cyclic shift. Let's say we have the sequence: 012345

If we use a shift value of 4 and just shift all the numbers, the result will be: 456789

We want the values of the numbers to remain between 0 and 5. To do this we will use the modulus operator. The expression **x%y** will return a number in the range 0 to **y-1** inclusive**,** e.g. 4%6 = 4, 6%6 = 0, 7%6 =1. Thus the result of the operation will be:

```
450123
```

Hint: Note that the ascii value of 'A' is 65 and 'a' is 97, not 0. So you will have to think how to use the modulus operator to achieve the desired result. ***Apply the cipher separately to the upper and lower case letters.***

Here is what you program should output:

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is:  Qechec! Qechec!
```

Part VII: Number Triangle (Optional)

Write nested loops that will print the following pattern:

```
                                    1
                                1   2   1
                            1   2   4   2   1
                        1   2   4   8   4   2   1
                    1   2   4   8   16  8   4   2   1
                1   2   4   8   16  32  16  8   4   2   1
            1   2   4   8   16  32  64  32  16  8   4   2   1
        1   2   4   8   16  32  64  128 64  32  16  8   4   2   1
```

Reproduce the pattern exactly; note the spacing and how the digits align between different lines.