CMP4*266*

Lab Session 3: Iterations using loops

1. Objectives

- Understand the concept of a loop for defining repeated tasks
- Understand the structure of a while loop
- Understand the for loop structure used with the range() function
- Be able to define a for loop to implement counter-controlled repetition
- Be able to convert a for loop into an equivalent while loop and vice versa

Note: There may be more material in this lab than you can finish during the lab time. If you do not have time to finish all exercises (in particular, the programming problems at the end that should be submitted to Moodle), you can continue working on them later. You can do this at home (if you have a computer with python set up), in the BCU lab rooms outside teaching hours, or on one of the computers available in the university libraries or other teaching spaces.

1. Reminder about loops

In programming, iteration is used for repetition. When you write a program or implement an algorithm, you may need to iterate over a certain part of the code until a specific condition meets. Loop statements help you iterate over those portions of the code. In this lab you will learn about various kinds of loop statements. You can use them with conditional statements to make smart programs.
There are two types of iteration:

- Definite iteration, in which the number of repetitions is specified explicitly in advance.
- Indefinite iteration, in which the code block executes until some condition is met.

In Python, indefinite iteration is performed with a while loop.

1.1 While loop

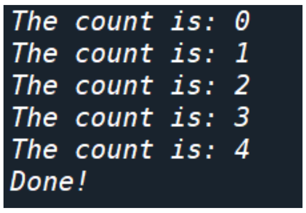
- The while loop continues executing as long as a certain condition is true. The syntax of a while loop in Python programming language is:

```
while expression:
  statement(s)
```

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

```
count = 0
while (count < 5):
   print ("The count is:", count)
   count += 1

print ("Done!")
```

When the above code is executed, it produces the following result:



```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
Done!
```

- A loop becomes infinite loop if a condition never becomes FALSE.

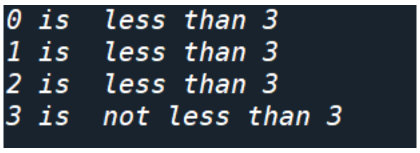
```
count = 0
while (count < 5):
   print ("The count is:", count)
   count -= 1

print ("Done!")
```

Above example goes in an infinite loop and you need to exit the program.

- Python supports an else statement associated with a loop statement. If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

```
count = 0
while count < 3:
   print (count,"is  less than 3")
   count = count + 1
else:
   print (count,"is  not less than 3")
```

When the above code is executed, it produces the following result:



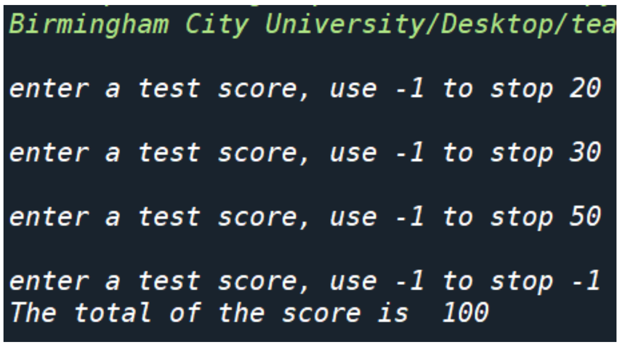
```
0 is  less than 3
1 is  less than 3
2 is  less than 3
3 is  not less than 3
```

- While loops with sentinel values

A sentinel value denotes the end of a data set, but it is not part of the data. A loop that uses a sentinel value is called a sentinel-controlled loop. In the following program, test scores are provided (via user input). Once the sentinel value of -1 is input, the loop terminates. At that point, the average of the test scores will be printed.

```
score = input("enter a test score, use -1 to stop ")    #enter the first value
total = 0
scoreCounter = 0

while score != -1:                    #loop until the sentinel value (-1) is entered
        total = total + score
        scoreCounter = scoreCounter + 1
        score = input("enter a test score, use -1 to stop ")

print ("The total of the score is ",total)
```

When the above code is executed, it produces the following result:



```
Birmingham City University/Desktop/tea

enter a test score, use -1 to stop 20

enter a test score, use -1 to stop 30

enter a test score, use -1 to stop 50

enter a test score, use -1 to stop -1
The total of the score is  100
```

1.2 For loop

In Python a for-loop is used to step through a sequence e.g., count through a series of numbers or step through the lines in a file. This is helpful in repetitive instances where a user needs to perform the same task many times; or when a user needs to iterate through a sequence. The syntax of a while loop in Python programming language is:

```
for variable in [val1, val2, val3, etc]:
   statements
```

where the variable is the target of the assignment at the beginning of each iteration.
E.g.

```
for num in [0, 1, 2, 3, 4]:
    print(num, end=' ')
```

This will give the following output: 0 1 2 3 4.

If you do need to iterate over a sequence of numbers, the built-in function range() comes in handy. It generates arithmetic progressions.
range(start, stop, step)

- If the step is omitted, the default step is 1.
    - range(0, 7) defines the sequence 0, 1, 2, 3, 4, 5, 6
    - range(6, 10) defines the sequence 6, 7, 8, 9

- If both the start and the step are omitted, the sequence starts from 0 with a step increment of 1.
    - range(5) defines the sequence 0, 1, 2, 3, 4,

- range(7) defines the sequence 0, 1, 2, 3, 4, 5, 6

The “range” function can be used to simplify the process of writing a for loop. We can rewrite the code using the “range” function and get the same result as follows:

```
for num in range(5):
    print(num, end=‘ ’)
```

This will give the same output as: 0 1 2 3 4

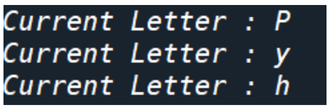
```
# Prints out 0,1,2,3,4
for x in range(5):
    print(x)

# Prints out 3,4,5
for x in range(3, 6):
    print(x)

# Prints out 3,5,7
for x in range(3, 8, 2):
    print(x)
```

1.3 Jump Statements

Jump statements are used to skip, jump, or exit the running program from inside the loop at a particular condition. With the help of the **break** statement, we can terminate the loop. We can use it to terminate the loop if the condition is true. Using the **continue** statement, we can terminate any iteration and it returns the control to the beginning again.

```
for letter in 'Python':
    if letter == 't':
        continue
    if letter=='o':
        break
    print('Current Letter : ' + letter)
```

When the above code is executed, it produces the following result:



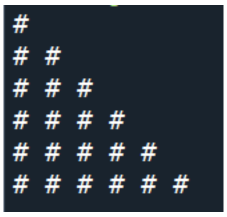
```
Current Letter : P
Current Letter : y
Current Letter : h
```

1.4 Nested loops

When you define one loop inside another loop, the resulting loop is called **Nested loop** in Python. The “inner loop” will be executed one time for each iteration of the “outer loop”:

```
rows = 6
# outer loop
for i in range(1, rows + 1):
   # inner loop
   for j in range(1, i + 1):
      print("#", end=" ")
   print(")
```

When the above code is executed, it produces the following result:



```
#
# #
# # #
# # # #
# # # # #
# # # # # #
```

2 Getting started

2.1 The following fragment of code is written using a for loop. Re-write it using a while loop.

```
sum = 0
     for k in range(1, 10, 3):
 sum = sum + k
print(sum)
```

2.2 Show the output from the following code:

```
for i in range(-10):
    print(i * i)
```

```
total = 0
for number in range(9, 20):
   if number % 2 == 0 or number % 3 == 0:
      total = total + 1
print(total)
```

3 Lab 3 Submission Exercises

3.1 A positive whole number n > 2 is prime if no number between 2 and $\sqrt{n}$ (inclusive) evenly divides n. Write a program that accepts a value of n as input and determines if the number is prime. If n is not prime, your program should print all the divisors of n.

3.2 Write a menu-driven program that allows the user to make transactions to a bank account. The options of the menu are:

- Option 1: Make a Deposit
- Option 2: Make a Withdrawal
- Option 3: Obtain a Balance

- Option 4: Quit

a) First the program asks the user for his/her name.
b) The user can make many interactions as they wish until they decide to quit by pressing Q or q from the keyboard. (Hint: while loop)
c) Assume that the account initially has a balance of £1000.
d) If the user tries to withdraw an amount more than the total balance, the program should print 'It is not possible to withdraw beyond the account balance
e) Ask the user to make a selection from the menu options.
f) Make sure the user enters a proper menu number.
g) If option one is selected, allow the addition of funds to the balance.
h) If option two is selected, subtract the amount from the balance.
i) If option three is selected, display the total balance of the checking account.

For exercise 3.2 download the lab3.py from Moodle. Complete the prompted areas, and run it to make sure it works. Of course, you are allowed to add extra lines of code if necessary. Please note that this is NOT the only way, that this program can be written. The purpose of this exercise is to get us used to breaking down problems in a systematic and logical manner and to exercise in using loops.

4 Moodle submission

At the end of this lab session, zip the python files that you created in Sections 3 (3.1 and 3.2) and then upload the zip file in the Moodle. **Note**: It is important to complete the weekly labs in particular labs 1, 2, 3, 4, 5 and 6 because it contains questions that are part of the coursework. Failure to complete and submit the answers to these labs on the Moodle before the deadline may mean giving you **zero** on for this assessment component.

If you are unable to finish all the tasks, submit whatever you've managed to produce by the due date.

The zip file should be named using the following format StudetName_studentID.zip For example: OgertaElezaj_123456789.zip

5 Additional exercises

5.1 Write a loop that computes the sum of the squares of the numbers between 1 and 100, inclusive.

5.2 Use a single for loop to display a rectangle of asterisks with a given height and a given width.

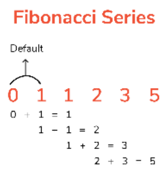
```
****************
*              *
*              *
*              *
****************
```

5.3 Write a loop that computes the sum of all the odd digits in a non-negative integer n.

5.4 Write a program that accepts integer inputs from a user as long as the word stop is not input. Compute and print the minimum of these integers.

5.5 Write a program that loads an integer and prints its second digit from the left, third digit from the right. The digits will be printed if the given number has three digits, otherwise a message should be printed.

5.7 Write a Python program to compute Fibonacci series.



5.8 Extend Exercise 3.2. Modify your exercise 3.2 to find every prime number less than or equal to n.

5.9 Extend Exercise 3.2. The Goldbach conjecture asserts that every even number is the sum of two prime numbers. Modify the program written in 3.2 that gets a number from the user, checks to make sure that it is even, and then finds two prime numbers that add up to it.