MCT-242 : COMPUTER PROGRAMMING-I

using Python 3.9

Prepared By:

Mr. Muhammad Ahsan Naeem

YouTube Playlist

https://www.youtube.com/playlist?list=PLWF9TXck7O_wMDB-VriREZ6EvwkWLNB7q

LAB 09: ITERATION: FOR LOOP: CLO 4

Sometimes, we have to repeat(iterate) some statements multiple times. For that Python has statements for iteration; generally known as loop. There are two types of loops; **`for`** loop and **`while`** loop.

for Loop:

Here is a simple code that takes a number from user and prints the square of it:

```
num=eval(input('Enter a number: '))
print(f'The square of {num} is {num**2}')
```

Suppose we want to repeat above task i.e. taking a number from user and displaying the square for ten times, then instead of copying the two lines ten times, we can do this in much better way as:

```
for i in range(10):
    num=eval(input('Enter a number: '))
    print(f'The square of {num} is {num**2}')
print('Thanks')
```

How this works:

- Line **`1`** is the syntax of **`for`** loop and line **`2`** and **`3`** are the block of the loop as we have block for **`if-else`** statements and functions etc.
- In line **`1`**→
  - **`for`** is the keyword for the loop.
  - **`i`** is variable name, known as loop variable. It can be any name.
  - **`range(10)`** specifies that it will execute the block of **`for`** loop 10 times.
- **`range()`** is a built-in function in Python. We will explore this function step by step. At this moment, you should know that **`range(x)`** sets numbers from **`0`** to **`x-1`** e.g. **`range(10)`** sets numbers from 0 to 9 (total 10).

Here is the flow execution of the above code:

- Interpreter execute line **`1`** and assigns variable **`i`** the first value of range i.e. **`0`**
- Interpreter enters into block of loop and executes line **`2`** and **`3`**.
- Interpreter then loops back to line number **`1`** (and that's why it is called loop) and assigns variable **`i`** the second value of range i.e. **`1`**.
- It then enters into block and executes line **`2`** and **`3`**.
- It then loops back to line number **`1`** and assigns next value to variable **`i`** and then enters into block again.
- It continues looping back until the variable **`i`** gets the last value of the range i.e. **`9`**, enters into block of the loop, executes line **`2`** and **`3`** and will move to line **`4`**.

Tasks:

[1] Write a program that will ask user to enter a number, say in variable `a`, and then will display asterisk box of size a-by-10.
All you need is to take a number from user in variable `a` and execute the statement `print('*'*10)` that number of time.

Indentation:

Indentation is very important while working with loop as in case of `if-else` statement. Try to figure out the output of following code and then verify:

```
print('A')
print('B')
for i in range(2):
        print('C')
        print('D')
print('E')
```

Now what if we add indentation to the last line as:

```
print('A')
print('B')
for i in range(2):
        print('C')
        print('D')
        print('E')
```

Again, find the output by hand and then verify by executing the code.

Loop variable and range():

As described in detail of above example that the built-in function `range()` assigns different values to loop variable. In every iteration the loop variable gets the next value until it reaches the last value of the range. We can view these different values by printing the loop variable within the loop block as shown here:

```
for k in range(5):
    print(k)
```

This will be the output of above code:

```
0
1
2
3
4
```

There are different ways we can use `range()` function. Instead of **0**, if we want to start the range from some other value, we can use it like:

```
for k in range(1,5):
    print(k)
```

Here is the output:

```
1
2
3
4
```

Note that the last value is still **4**.

We can also use negative values as:

```
for k in range(-10,-2):
    print(k)
```

Here is the output:

```
-10
-9
-8
-7
-6
-5
-4
-3
```

There is also the possibility to specify the difference between range values. By default, the difference is **1** but we can set it by providing a third input argument as shown here:

```
for k in range(1,21,3):
    print(k)
```

The output is:

```
1
4
7
10
13
16
19
```

Tasks:

[2] Write a program that will ask user to enter a number and then will display **1** to entered number with their cubes separated by a tab.

Sample Output is:

```
Enter a number: 4
1    1
2    8
3    27
4    64
```

[3] Write a program that prints the multiplication table of **12** from **1** to **10** using for loop as shown below:

```
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
```

[4] Modify above program such that it takes two inputs from user. First the number of which the multiplication table user wants at output and second the final limit of the table e.g. final limit of above table was 10.

Sample Output is:

```
Which multiplication table you want: 5
What is final limit: 12

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5  x 12 = 60
```

[5] Use a for loop to print a triangle like the one below. Ask the user to specify the height.

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

Sample output is:

```
Enter height of triangle: 6

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

[6] Repeat above task for upside down triangle as shown below. Again, take the height from user:

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

[7] Repeat above task for upside down triangle as shown below. Again, take the height from user:

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

for loop in User Defined Functions:

We can use for loop inside the block of user defined function if required. Suppose we want to print asterisk box of size specified by the user; it can be done as:

```
x=eval(input('Enter no. of rows: '))
y=eval(input('Enter no. of cols: '))
for i in range(x):
    print('*'*y)
```

We can create a user defined function for this task and call that in main program as shown here:

```
def drawBox(m,n):
    'Draws a asterisk box of size m-by-n'
    for i in range(m):
        print('*'*n)

### Main Program Starts Below ###
x=eval(input('Enter no. of rows: '))
y=eval(input('Enter no. of cols: '))
drawBox(x,y)
```

Tasks:

[8] Repeat task 4 by creating a function named as **`DispTable()`** with two input arguments. Use the function in main program to print table specified by the user.

Calculations within Loop

We can use loop iterations for various calculation as described below:

Counting:

Quite often we want our program to count something happening. For example, player score, number of deaths etc. in a game or count specific type of numbers and many other cases.

Let's suppose we want to ask user to enter ten numbers and count that how many entered numbers are between 10 and 20. This can be done with following logic:

- Declare a variable, say **`count`** and initialize it to **`0`**.
- Write a **`for`** loop that will take ten numbers form user.
- Within the body of the loop, apply a condition to check whether entered number is between **`10`** and **`20`**. If it is, add **`1`** to variable **`count`**.

Complete code is as under:

```
count=0
```

```python
for i in range(10):
    x=eval(input('Enter a number: '))
    if(x>=10 and x<=20):
        count+=1

print(f'{count} out of 10 entered numbers are between 10 and 20.')
```

Tasks:

[9] Write a program that prints all numbers from 1 to 100 for which the unit and tenth digits of their square are equal but not zero. For example, square of 12 is 144 which has last two digits equal and other than 0. Basic code structure is given below:

```python
for i in range(1,101):
    #Complete the logic
```

All you need is to find square of loop variable i.e. **`i`** and store it in a variable. Then use remainder division and floor division to get the last two digits of the square. Apply condition on them; if satisfied, display **`i`** and otherwise do nothing.

[10] Write a program that will ask user to enter 10 positive integers and then will display how many of entered numbers are even and how many are odd. You must declare two count variables now, say **`countEven`** and **`countOdd`**. Initialize both to **`0`** before the **`for`** loop. Within loop apply condition and accordingly increment the respective count variable. Display them at the end.

<u>Sample output is:</u>

```
Enter number 1: 23
Enter number 2: 14
Enter number 3: 20
Enter number 4: 39
Enter number 5: 44
Enter number 6: 78
Enter number 7: 99
Enter number 8: 103
Enter number 9: 52
Enter number 10: 38

You entered 6 even and 4 odd numbers!
```