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 11: NESTED FOR LOOPS: CLO4

We did different calculations us using for loop in last lab session. The last program was to find the factorial of input number by creating a function. We supposed there that user will always enter a positive number only. If we want to handle negative numbers and 0 as well, we can write it as:

```
def fact(a):
    ans=1
    if(a>=0):
        for i in range(1,a+1):
            ans*=i
        return ans

### Main Programs Starts from here ###
x=eval(input('Enter a number: '))
y=fact(x)
if (y==None):
        print('Factorial of negative numbers does not exist!')
else:
        print(f'{x}!={fact(x)}')
```

In case of negative numbers, nothing is returned by the function and **y** the main program contains **None** which is another datatype in Python. We can assign **None** to some variable meaning that variable is defined but contains nothing.

Multiple Loops:

We can have more than one loop in a program as per requirement. The following two loops are independent loops:

```
for i in range(5):
        print('Computer')
for i in range(3):
        print('Programming')
```

Just to illustrate multiple loops, if we take a number from user and print its factorial only if it is a prime number, this can be done as:

```
x=eval(input('Enter a number: '))
check=True
for i in range(2,x):
    if(x%i==0):
        check=False
if(check):
    f=1
    for i in range(1,x+1):
        f*=i
    print(f'{x}!={f}')
else:
    print("We find factorial of Prime Numbers Only!")
```

Nested for Loop:

We have seen that there can be any logic like `if-else` inside a for loop depending upon the requirements. There can also be another `for` statement inside a `for` loop!

See this code:

```
for i in range(5):
        for j in range (3):
                print('Computer')
```

Computer will be printed 15 times!

How this works?

- The loop with loop variable `i` is known as outer loop while the one with loop variable `j` is known as inner loop.
- For the first value of loop variable `i=0`, there will be complete iterations of inner loop i.e. `j` will iterate from `j=0` to `j=2` and will print `'Computer'` three times.
- After complete iterations of inner loop, the outer loop variable `i` will get the next value, `i=1` and there will be complete iterations of inner loop as earlier printing `'Computer'` three times.
- Outer loop variable will get next value and this will continue for complete range of `i`.
- To summarize, we can say that outer loop is iterating the inner loop five times. Inner loop prints `'Computer'` three times and hence the overall execution prints `'Computer'` fifteen times.

To see the different values of loop variables `i` and `j` in different iterations, let's run this code:

```
for i in range(5):
    for j in range (3):
        print(f'i={i}\tj={j}')
```

This will be the output:

```
i=0     j=0
i=0     j=1
i=0     j=2
i=1     j=0
i=1     j=1
i=1     j=2
i=2     j=0
```

```
i=2      j=1
i=2      j=2
i=3      j=0
i=3      j=1
i=3      j=2
i=4      j=0
i=4      j=1
i=4      j=2
```

Think about the output of following code and verify:

```
for i in range(5):
    print(f'i={i}')
    for j in range (3):
        print(f'j={j}')
```

The second line in above code will get executed five times while the fourth one will execute fifteen times.

Tasks:

[1] If you observe the pattern of loop variables in nested loop in above examples, it is like row number followed by column number notation of elements of matrix. A general 3x3 matrix is represented as:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Write a program that that will take all entries of a 3x3 matrix and will display the sum and product of all its entries.

Sample output is:

```
Enter the entries of 3x3 Matrix A
a11= 2
a12= -1
a13= 3
a21= 4
a22= -3
a23= 1
a31= 4
a32= 3
a33= -2

The SUM of all entries = 11
The PRODUCT of all entries = -1728
```

More on nested loop:

Think about print this table:

```
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      40
5       10      15      20      25      30      35      40      45      50
6       12      18      24      30      36      42      48      54      60
7       14      21      28      35      42      49      56      63      70
8       16      24      32      40      48      56      64      72      80
9       18      27      36      45      54      63      72      81      90
10      20      30      40      50      60      70      80      90      100
```

Actually, this can be obtained by two nested loops with loop variables **i** and **j** and the product will give the above values.

But if we try to print them as:

```
for i in range(1,11):
        for j in range(1,11):
                k=i*j
                print(k,'\t',end='')
```

All values will be printed on one line. In above code when **i=1** and **j** ranges from **1** to **10** we get the first line of above table in variable **k** and it gets printed. Next is **i=2** and the second line of above table will be variable in variable **k** but we want to print that from new line. This can be done by adding a new line print statement outside the **j** variable loop but inside **i** variable loop as shown:

```
for i in range(1,11):
        for j in range(1,11):
                k=i*j
                print(k,'\t',end='')
        print()
```

So after each line printed inside inner loop, **print()** will be executed which adds a new line followed by next iteration of the inner loop.

Example (Solving double sum):

Write a program to evaluate following double sum:

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(i^2+j^2)$$

The outer summation can be expanded as:

$$\sum_{j=1}^{n}(1+j^2)+\sum_{j=1}^{n}(4+j^2)+\sum_{j=1}^{n}(9+j^2)+\sum_{j=1}^{n}(16+j^2)\cdots\sum_{j=1}^{n}(m^2+j^2)$$

If we consider one term from above expression, say the **second** one, it can be programed as:

```
innerSum=0
for j in (1,n+1):
        innerSum+=4+j**2
```

Now what we want is the repeated iterations of above code for **`i=1`** to **`m`** with above summation statement as **`innerSum+=i**2+j**2`**. and summing those iterations. It is shown here:

```
totalSum=0
for i in (1,m+1):
        innerSum=0
        for j in (1,n+1):
                innerSum+=i**2+j**2
        totalSum+=innerSum
```

Resetting count/summing/product/flag variable:

We have seen use of count, summing, product and flag variables for respective calculations. When these calculations are repeated in iteration as in above example, it is very important to reset the corresponding variable before next calculation. In above example, **`innerSum`** is variable to hold the sum of inner summation. If we put it outside the outer loop as:

```
totalSum=0
innerSum=0
for i in (1,m+1):
        for j in (1,n+1):
                innerSum+=i**2+j**2
        totalSum+=innerSum
```

This is going to give wrong results because in every next iteration of inner summation, **`innerSum`** will hold the result of previous iteration sum. By keeping it inside outer loop, it gets reset before the start of inner loop every time.

Tasks:

[2] Write a program to find the sum of the following series:

$1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots \text{n terms}$

It can also be written as:

$$\sum_{k=1}^{n} \frac{1}{k!}$$

Take value of `n` from user.

You must use nested loops here. The outer loop will iterate from `1` to `n` and the inner loop will calculate the factorial of outer loop variable and sum the series as given.

Sample output is:

```
Enter n: 3
Sum of series is: 1.6666666666666667
```

[3] Repeat task 3 using two functions. The first function is the factorial as did in previous lab session. Name it as `fact()` with one input argument and the output will be the factorial of it. The second function should be named as `mySeries()`. It will take one input argument and will calculate the above series sum using `fact()` function. In main program take `n` from user and use `mySeries()` to find the series sum and display the result.

[4] Write a program that will take a number (assuming user will always enter positive integer) from user and will print all prime numbers from 2 to entered number in one line. You must complete the logic in main program without writing any function.

You will use a `for` loop to iterate from 2 to entered number. Inside loop block you have to check whether the loop variable is prime or not and for that you need another `for` loop that will iterate from 2 to square root of outer `for` loop variable.

Sample output is:

```
Enter number: 32
2 3 5 7 11 13 17 19 23 29 31
```

[5] Repeat task 4 by creating `isPrime()` function and using that in main program. In main program you will use one `for` loop from 2 to entered number and inside the loop block you will use `isPrime()` function on loop variable to check if it is prime or not. If prime, display it otherwise just ignore it.