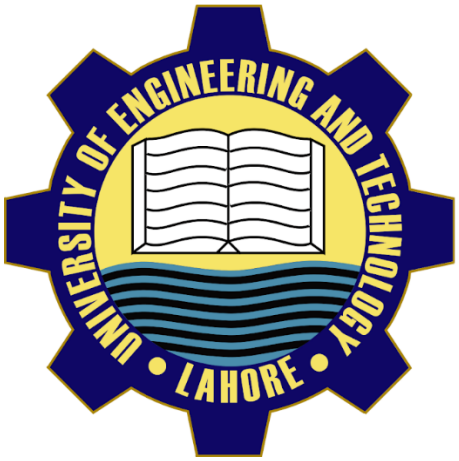
MCT-242 : COMPUTER PROGRAMMING-I

using Python 3.9



Prepared By:

Mr. Muhammad Ahsan Naeem

YouTube Playlist

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

LAB 10: CALCULATIONS WITHIN LOOP: CLO 4

Summing:

We can use `for` loop to iterate over values and keep summing those. For example, if we want to find the sum of ten numbers entered by user, it can be done as:

- Declare a variable, say `total` and initialize it to `0`.
- Write a `for` loop that will iterate ten times and take a number form user say in variable `x`.
- Within the body of the loop, add `x` into `total` variable as `total+=x`

The complete code is given as:

```
sum=0
for i in range(10):
    x=eval(input(f'Enter number {i+1}: '))
    sum+=x

print(f'Sum of entered numbers is: {sum}')
```

Tasks:

[1] Write a program that will ask user to enter ten positive numbers and then will show sum of even numbers entered and sum of odd numbers entered.

Sample output is:

```
Enter number 1: 12
Enter number 2: 4
Enter number 3: 8
Enter number 4: 20
Enter number 5: 7
Enter number 6: 6
Enter number 7: 15
Enter number 8: 9
Enter number 9: 4
Enter number 10: 3

Sum of even numbers entered is: 54
Sum of odd numbers entered is: 34
```

[2] Write a program that will ask user to enter the marks of five subjects and will then display the mean marks. You have to take input using `for` loop. To find mean, we need to have sum of marks. Find the sum within `for` loop and outside the loop divide that sum by 5 which will be the mean.

Sample output is:

```
Enter Subject-1 marks: 92
Enter Subject-2 marks: 90
Enter Subject-3 marks: 82
Enter Subject-4 marks: 86
Enter Subject-5 marks: 78

The mean of five subject marks is 85.6
```

[3] Write a program that that will take a number from user. If the number is even, it will print sum of all even numbers from 0 to that number and if the entered number is odd, it will print sum of all odd numbers from 1 to that number. Furthermore, if user enters a negative number, it should display the message that negative numbers are not allowed.

Sample output is:

```
Enter a number: 9
Sum of odd number (1-9)=25
```

Another sample output is:

```
Enter a number: 12
Sum of even numbers (0-12)=42
```

Another sample output is:

```
Enter a number: -10
Negative numbers are not allowed
```

Additional Feature:

You can add additional feature to show the output of the program as:

```
Enter a number: 9
1+3+5+7+9 = 25
```

Product using loop:

Like the summation we can use loop to calculate the product of range of numbers. The variable to hold the product, say with name **`product`** will be initialized to **`1`** and inside the loop it will be updates as **`product*=x`**. For example, to calculate the product of ten entered numbers by user, we can write as:

```
product=1
for i in range(10):
    x=eval(input(f'Enter number {i+1}: '))
    product*=x

print(f'Product of entered numbers is: {product}')
```

[4] Create a function named as `fact()` that will take one number as input and will return the factorial of that number. You can assume here that the user will enter a positive number, so need not to consider 0 and negative numbers. (We will consider them in next lab session)
The main program is given here:

```
### Main Programs Starts from here ###
x=eval(input('Enter a number: '))
print(f'{x}! = {fact(x)}')
```

The Sample Output is:

```
Enter a number: 4
4! = 24
```

Flag Variables:

Sometimes, we are required to find something happened or not. We can use variables as flag for such cases. For example, if we take ten numbers from user and are interested to find whether user entered 100 or not; once or more. This can be done as:

```
check=0
for i in range(10):
    x=eval(input(f'Enter number {i+1}: '))
    if(x==100):
        check=1
#Outside for loop
if(check==1):
        print('100 was entered by you!')
else:
        print('You didn\'t enter 100!')
```

- A variable named `check` is initialized to 0.
- Then is the `for` loop to take input ten numbers from user.
- Within the body of the `for` loop, there is a condition to see whether the entered number is `100` or not. If it is `100`, the variable `check` is being assigned a different value i.e. `1`.
- In the execution of the `for` loop, if there was `100` entered by user, the variable `check` will have value `1` and if there was not any `100` entered by the user, the variable `check` will contain `0`.
- Therefore, after the `for` loop the value of variable `check` is tested in an `if` statement and a message is printed accordingly.

Note that if user enters 4th number as `100`, variable `check` will become `1` and after that if user enters the 5th number other than `100`, the variable `check` will remain `1`. So, in above code, if `100` is entered

at any turn, variable **check** will get value **1** and it will have value **0** only if there was not any **100** entered.

Tasks:

[5] Write a program that asks user to enter a number and will display whether entered number is prime or not?
A number is a prime number if it has only two factors; 1 and the number itself e.g. 13 is a prime number. Or in other words there is no factor between two and one less than that number.
Therefore, for this task you need to follow these steps:

- After taking a number from user say in variable named as **num**, you need to check whether there is any factor between **2** and **num-1**.
- This can be done by using a **for** loop that iterates between these values and it can be simply done using **range(2,num)**.
- Within the loop body, find remainder when **num** is divided by loop variable and see if it is **0** or not (condition of factor).
- Use a flag variable to change its value if there is any factor.
- Based on the value of the flag after complete iterations of the loop display the appropriate message.

Use of Boolean variables as Flag:

Recall from the discussion of **if-else** statement that the relation expressions result in **True** or **False**. We can see the result of relation expression as:

```
x=3
y=3
print(x==y)
```

And it will have the output:

```
True
```

We can also directly assign a value **True** or **False** to a variable as **a=True** which makes it a Boolean type. So, the program to check if **100** is entered by user in ten entries can be written as:

```
check=False
for i in range(10):
    x=eval(input(f'Enter number {i+1}: '))
    if(x==100):
        check=True
#Outside for loop
if(check==1):
        print('100 was entered by you!')
else:
```

```
        print('You didn\'t enter 100!')
```

Note that in last **if** statement we still have **if(check==1)**. In programming languages **1** is treated as **True** while **0** means **False**. We could also write as **if(check==True)**.

The best way to use a Boolean variable in some relational statement is direct comprehension of a Boolean variable. As described earlier, a relational statement results in **True** or **False** so for above case we can write:

```
if(check):
        print('100 was entered by you!')
else:
        print('You didn\'t enter 100!')
```

Because **check** is a Boolean variable and will contain **True** or **False** so there is no need to apply a relation that it is equal to **True** or not. In above case if **check** contains **True**, the **if** block will execute and if it contains **False**, the **else** block will get executed.

Boolean variable as Output Argument of a function:

```
def isEven(a):
        'Returns True if a is Even and False otherwise'
        if(a%2==0):
                return True
        else:
                return False
### Main Programs Starts from here ###
x=eval(input('Enter a number: '))
if(isEven(x)):
        print(f'{x} is an Even number.')
else:
        print(f'{x} is an Odd number.')
```

Tasks:

[6] Repeat task 5 by creating a function with Boolean output to decide that the entered number is prime or not. The main program is given here:

```
### Main Programs Starts from here ###
x=eval(input('Enter a number: '))
if(isPrime(x)):
        print(f'{x} is a Prime number.')
else:
        print(f'{x} is not a Prime number.')
```

Second thing you need to add in this program is to make it computationally efficient. Previously, we iterated from two to entered number minus one. But you can figure out very easily that the maximum possible factor of any number is half of that number. For example, for 80, the maximum factor is 40. Therefore, instead of iterating from two to entered number minus one, you can iterate from two to entered number divided by two. This will save half iterations while giving the same results.

Moreover, it is also a fact that if there exists no factor till square root of a number, there will not be any after that. Hence, to further improve the code computationally, you can iterate the loop from `2` to square root of the entered number but of course converting the square root to integer to be used in `range()`.