Student Name: _____ **Roll No:** _____ **Section:** _____

Experiment No. 04

Lab 04 –Conditional Statements with if - else and Simple loops.

Lab Objectives:

1. Simple if condition
2. If with else condition
3. Simple loop
4. Nesting with condition and loop

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement −

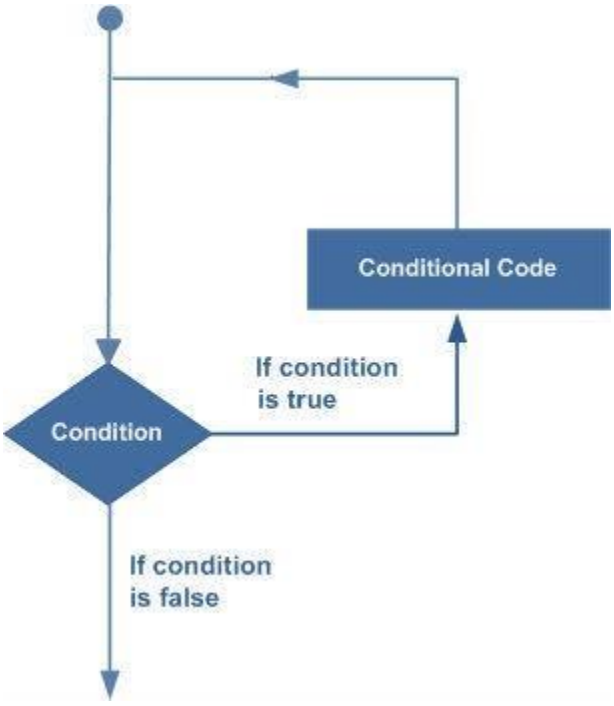


Figure 1: A simple loop architecture

Student Name: _____ **Roll No:** _____ **Section:** _____

Python programming language provides following types of loops to handle looping requirements.

Sr.No.	Loop Type & Description
1	**while loop** Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2	**for loop** Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	**nested loops** You can use one or more loop inside any another while, for or do..while loop.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements.

Sr.No.	Control Statement & Description
1	**break statement** Terminates the loop statement and transfers execution to the statement immediately following the loop.
2	**continue statement** Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
3	**pass statement** The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

1. Simple *if* Condition

Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.

Following is the general form of a typical decision making structure found in most of the programming languages.

Student Name: _____ Roll No: _____ Section: _____

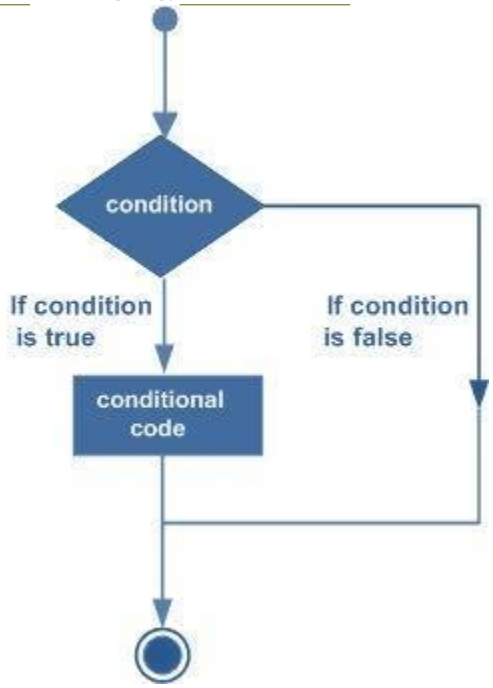


Figure 1: Shows the flow structure of *if* statement

Python programming language assumes any non-zero and non-null values as TRUE, and if it is either zero or null, then it is assumed as FALSE value.

Python supports the usual logical conditions from mathematics:

Logics	Logical Conditions
Equals	a == b
Less than	a < b
Greater than	a > b
Not Equals	a != b
Less than or equal to	a <= b
Greater than or equal to	a >= b

Program 1: Practicing with simple if condition. Execute the following program with a = 400 and b = 500. Then change the value of a =500 and b = 400.

Code:

```
a = 400
b = 500
if a == 400:
        print("The value of a is equal to 400")
if a < b:
        print("The value of a is less than b")
if a > b:
        print("The value of a is greater than b")
```

Student Name: _____ **Roll No:** _____ **Section:** _____

```
if a != b:

        print("The value of a is not equal to b")
if a <= b:
        print("The value of a is less than or equal to b")
if a >= b:
        print("The value of a is greater than or equal to b")
```

Output 1:

Output 2:

Conclusion of both the results:

Student Name: _____ Roll No: _____ Section: _____

Program 2: Practicing with simple if condition if both conditions are true using AND operator.

Code:

```
a = 10
b = 5
c = 20

if a > b and c > a:
  print("Both conditions are True")
```

Output:

Program 3: Practicing with simple if condition if any of the condition is true using OR operator.

Code:

```
a = 105
b = 50
c = 200

if a > b or a > c:
  print("At least one of the conditions are True")
```

Output:

2. *If* with *else* Condition:

An else statement can be combined with an if statement. An else statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.

The else statement is an optional statement and there could be at most only one else statement following if.

Student Name: _____ Roll No: _____ Section: _____

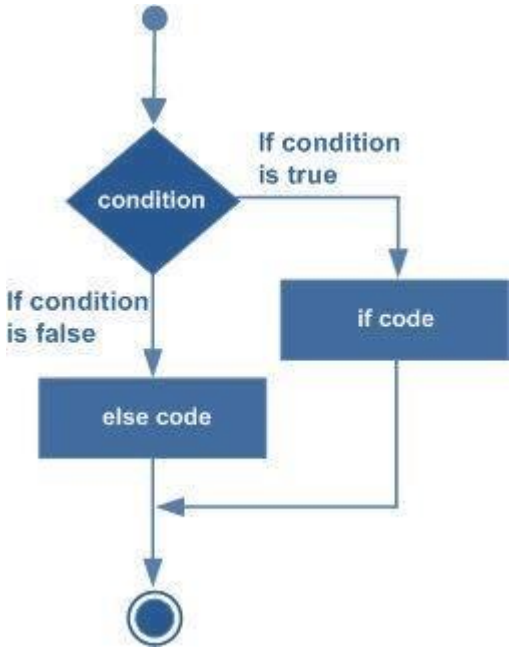


Figure 2: The flow structure of *if* with *else* statement

Program 4: Write a program which takes the lower limit and upper limit then find which of the number are prime number.

Code:

```
# Python program to display all the prime numbers within an interval

l_limit = int(input("Enter lower limit range: "))
u_limit = int(input("Enter upper limit range: "))

print("Prime numbers between",l_limit,"and",u_limit,"are:")
for number in range(l_limit,u_limit + 1):

   if number > 1:

      for i in range(2,number):
         if (number % i) == 0:
            break
      else:
            print(number)
```

Output:

Student Name: _____ Roll No: _____ Section: _____

3. Simple Loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). This is less like the for keyword in other programming language, and works more like an iterator method as found in other object-orientated programming languages. With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Program 5: Write a program which takes the initial and final values from the user then print the sum of all the number.

Code:

```
initial_value = eval(input("Enter the initial value for the range :"))
final_value = eval(input("Enter the final value for the range :"))
numbers = range(initial_value,final_value)
sum = 0

for value in numbers:
        sum = sum+value

print("The sum is", sum)
```

Output:

Program 6: Write a program which takes the number of rows and columns from the user and generate the values in form of list.

Code:

```
row_num = int(input("Input number of rows: "))
col_num = int(input("Input number of columns: "))
multi_list = [[0 for col in range(col_num)] for row in range(row_num)]

for row in range(row_num):
   for col in range(col_num):
      multi_list[row][col]= row*col

print(multi_list)
```

Student Name: _____ Roll No: _____ Section: _____

Output:

Program 7: Write a program which will check the data type of given data in a loop.

Code:

```
datalist = [300, 12.65, 5+6j, True, 'Faisal', (5, -7), [8, 52],{"color":'blue', "color":'red'}]
for item in datalist:
   print ("Type of ",item, " is ", type(item))
```

Output:

Program 8: Write a program to generate the ASCII Chart from 0 to 256.

Code:

```
print("\t\t\t ASCII Character")


for i in range(0, 256):
    print(i , "=",chr(i), end ="\t") #end="\t" is used to place a tab after the displayed string instead
of a newline.
print("\n")
```

Output:

Student Name: _____ Roll No: _____ Section: _____

Program 9: Write a program to convert digital number from 0 to 16 into binary, octal and hexa-decimal number system.

Code:

```
print("Python program to convert decimal number into binary, octal and hexadecimal number system")

for i in range(0, 17):
    print("The decimal value of",i,"is:", "in binary its :",bin(i), "in octal its:", oct(i), "and in Hexa-Decimal its:",hex(i))
print("That's the end of the Program with range from 1 to 16")
```

Output:

4. Nesting with Condition and Loop

Python programming language allows the usage of one loop inside another loop. The following section shows a few examples to illustrate the concept.

For Example:

```
for iterating_var in sequence:
  for iterating_var in sequence:
    statements(s)
  statements(s)
```

The syntax for a nested for loop statement with if condition in Python programming language is as follows −

Student Name: _____ Roll No: _____ Section: _____

For Example:

```
for iterating_var in sequence:
        if(test condition):
                statements(s)
                statements(s)

        else:
                statements(s)
  statements(s)
```

A final note on loop nesting is that you can put any type of loop inside any other type of loop. For example, a for loop can be inside another for loop or inside condition or condition may be called inside a loop and vice versa.

Program 10: Write a Python program to construct the following pattern, using a nested for loop.

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

Code:

```
n=5;
for i in range(n):
    for j in range(i):
        print ('* ', end="")
    print('')

for i in range(n,0,-1):
    for j in range(i):
        print('* ', end="")
    print('')
```

Student Name: _____ Roll No: _____ Section: _____

Output:

Program 11: Write a program which calculates the vowels from the given string.

Code:

```
print("This program will count total number of vowels from user defined sentence")
string=input("Enter your string:")
vowels=0
for i in string:
    if(i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i=='A' or i=='E' or i=='I' or i=='O' or i=='U'):
        vowels=vowels+1
print("Number of vowels are:")
print(vowels)
```

Output:

Student Name: _____ **Roll No:** _____ **Section:** _____

Programming Exercise

1. Write a program which solves the quadratic equation. The user will enter the value of a, b and c. The program will then check the denominator that if denominator is zero or not. If its zero it can reply the equation cannot solve as there is a zero division else, it will execute the program and will generate two solutions.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2. Calculate the arithmetic sequence of n numbers. The program will generate the nth term of the sequence, whereas the user will enter the first term and the common difference. The program will then ask either to continue or not, if the user will enter yes it will ask the next nth term to calculate. **Example**: you have entered the first term as 3 and common difference 6 you are interesting in 35^{th} term. So it will calculate and generate the answer as 207. Now it will again ask for you to continue if you agree it will ask next term like 45^{th} or 96^{th} term to calculate.

$$a_n = a_1 + (n-1)d$$

3. Write a program which will check either the giving string is Palindrome or not. Palindrome is a string when we reverse the string it will generate the original string. Example CIVIC, MOM, 010, 1001, etc. So if you enter the word which is Palindrome it will say yes your string is Palindrome otherwise it will generate sorry message.

[Hint: As user may enter upper or lower case, so you may use myString.casefold(). A casefold() will remove all the upper and lower case problems. To generate the reverse of string you can use reversed() function on your string.]

4. Write a program which will collect your name, your father's name, your roll number and your subjects (5 Subjects with name and numbers). At the end it will generate a result with your name, your father's name, your details subjects, marks you have obtained with total marks with grade and percentage.

[Hint: Your creativity and analysis approach will be checked]

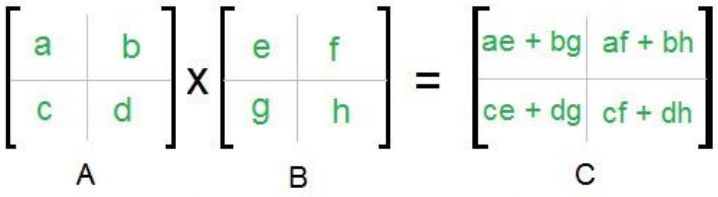
5. Generate a table from initial value to final, depending upon the user starting and ending range in matrix form such as:

```
1  2  3  4  5
2  4  6  8 10
3  6  9 12 15
4  8 12 16 20
5 10 15 20 25
```

Student Name: _____ **Roll No:** _____ **Section:** _____

6. Write a program which will add two square matrices.
7. Write a program which will multiply two square matrices.

[Hint: The following operation will be helpful for you when multiplying matrices.



A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2