Python Assignment No. 2

Q1.} Explain in details.

a)Boolean Not operator.

Ans.

The not operator is a unary operator. It is applied to just one value. The not operator takes a single operand and negates or inverts its Boolean value. If we apply the not operator on an expression having false value then it returns it as true. Similarly, if we apply the not operator on an expressionhaving true value then it returns it as false.

Example

Use of the not operator on a simple Boolean expression in Python, i.e. true and false.

```
>>> True
True
>>> not True
False
>>> False
False
>>> not False
True
```

b) Boolean And operator.

Ans.

The and is a binary operator. The and operator takes two operands and performs left to right evaluation to determine whether both the operands are true. Thus, and of Boolean operand is trueif and only if both operands are true.

X	*Y*	*X and Y*
True	True	True
True	False	False
False	True	False
False	False	False

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
```

c) Boolean Or operator.

Ans.

The or of two Boolean operands is true if at least one of the operands is true.

X	*Y*	*X or Y*
True	True	True
True	False	True
False	True	True
False	False	False

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

Q2.} Define using numbers with Boolean Operators.

Ans.

A programmer can use numbers with Boolean operators in Python.

```
>>> not 1
False

>>> 5
5
>>> not 5
False
>>> not 0
True
>>> not 0.0
True
```

Here, Python uses the Boolean operator not on the numbers and treats all numbers as True. Therefore, by writing not 1, Python substitutes 1 as True and evaluates not True, which returns False. Similarly, not is used before 5 and Python substitute True in place of 5 and it again evaluates the expression not True, which returns False. But in case of the numbers 0 and 0.0, Python treats them as False. Therefore, while evaluating not 0, it substitutes False in place of 0 andagain evaluates the expression not False, which returns True.

Q3.} Define using strings with Boolean Operators.

Ans.

Like numbers, a programmer can use strings with Boolean operators in Python.

```
>>> not 'hello'
False
>>> not ''
True
```

Here, Python uses the Boolean operator not on string. The expression not hello returns True since Python treats all strings as True. Therefore, it substitutes True in place of 'hello'and again reevaluates the expression not True, which returns False. However, if it is an empty string, Python will treat it as False. Therefore, it substitutes False in place of an empty string ''andreevaluates the expression not False, which in turn returns True.

Q4.} Explain various decision making statement:

a)if statement.

Ans.

The if statement executes a statement if a condition is true.

```
if condition:
    statement(s)
```

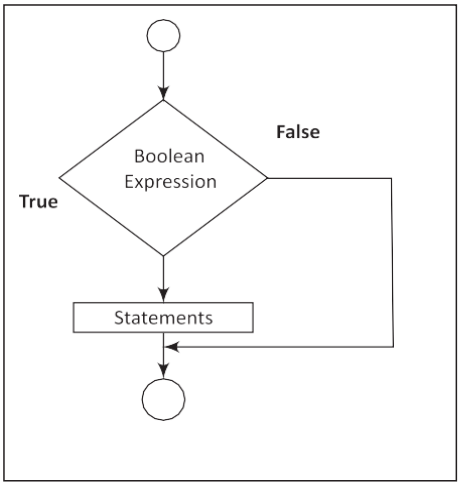
OR

```
if condition:
    Block
```

Figure 4.1 Syntax for if statement

Details of the if Statement

The keyword if begins the if statement. The condition is a Boolean expression which determines whether or not the body of if block will be executed. A colon (:) must always be followed by the condition. The block may contain one or more statements. The statement or statements are executed if and only if the condition within the if statement is true.



```
num1=eval(input("Enter First Number: "))
num2=eval(input("Enter Second Number:  "))
if  num1-num2==0:
    print("Both the numbers entered are Equal")

Output

Enter First Number: 12
Enter Second Number:  12
Both the numbers entered are Equal
```

b)if-else statement.

Ans.

The execution of the if statement has been explained in the previous programs. We know, the if statement executes when the condition following if is true and it does nothing when the conditionis false. The if-else statement takes care of a true as well a false condition.

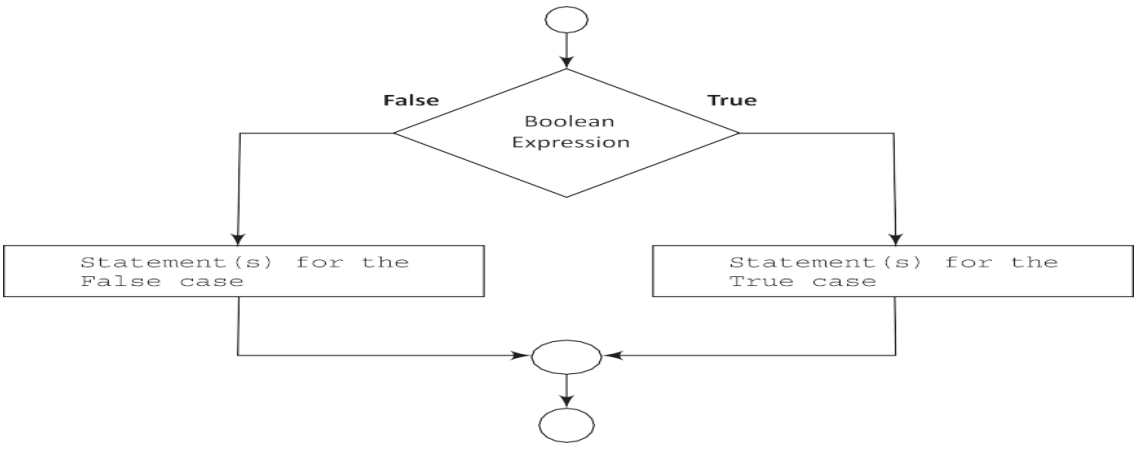
```
if condition:
    statement(s)
else:
```

OR

```
if condition:
    if_Block
else:
```

Details of if-else Statement

The if-else statement takes care of both true and false conditions. It has two blocks. One block is forif and it may contain one or more than one statements. The block is executed when the condition is true. The other block is for else. The else block may also have one or more than one statements. It is executed when the condition is false. A colon (:) must always be followed by the condition. The keyword else should also be followed by a colon (:).



```
num1=int(input("Enter the First Number:"))
num2=int(input("Enter the Second Number:"))
if num1>num2:
    print(num1,"is greater than ",num2)
else:
    print(num2,"is greater than ",num1)
```

Output

```
Enter the First Number:100
Enter the Second Number:43
100 is greater than 43
```

c)nested if statement.

Ans.

When a programmer writes one if statement inside another if statement then it is called a nestedif statement. A general syntax for nested if statements is given as follows:

```
if Boolean-expression1:
    if Boolean-expression2:
        statement1


      else:
          statement2
else:
statement3
```

In the above syntax, if the Boolean-expression1 and Boolean-expression2 are correct then statement1 will execute. If the Boolean-expression1 is correct and Boolean-expression2 is incorrectthen statement2 will execute. And if both Boolean-expression1 and Boolean-expression2 areincorrect then statement3 will execute.

d) multiway if-elif-else statement:

```
If Boolean-expression1:
          statement1
     elif Boolean-expression2 :


          statement2
     elif Boolean-expression3 :
          statement3
-    - - - - - - - - - - - - -
-    - - - - - - - - - - -- -
elif Boolean-expression n :
     statement N
else :
     Statement(s)
```

In this kind of statements, the number of conditions, i.e. Boolean expressions are checked from top to bottom. When a true condition is found, the statement associated with it is executed and therest of the conditional statements are skipped. If none of the conditions are found true then the last else statement is executed. If all other conditions are false and if the final else statement is not present then no action takes place.

Q.5} Explain the importance of loop control statements.

Ans.

In our day-to-day life, we perform certain tasks repeatedly. It can be tedious to perform such tasksusing pen and paper. For instance, teaching multiplication tables to multiple classes can become easier if the teacher uses a simple computer program with loop instructions instead of pen and paper.

Let us try to understand the concept of control statements in this context. Suppose a programmer wants to display the message, “I Love Python” 50 times. It would be tedious for him/her towrite the statement 50 times on a computer screen or even on paper. This task can become very easy, quick and accurate if the programmer completes it using loop instructions in a computer programming language. Almost all computer programming languages facilitate the use of control loop statements to repeatedly execute a block of code until a condition is satisfied.

Q.6} Explain range function in details.

Ans. There is a inbuilt function in Python called range(), which is used to generate a list of integers. The range function has one, two or three parameters. The last two parameters in range() are optional.
The general form of the range function is:

range(begin, end, step)

The 'begin' is the first beginning number in the sequence at which the list starts.

The 'end' is the limit, i.e. the last number in the sequence.

The 'step' is the difference between each number in the sequence.

```
>>> list(range(1,6))
    [1,2,3,4,5]
```

```
>>> list(range(1,20,2))
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

Q.7} Explain for loop in details.

Ans. The for loops in Python are slightly different from the for loops in other programming languages. The Python for loop iterates through a sequence of objects, i.e. it iterates through each value in a sequence, where the sequence of object holds multiple items of data stored one after another.

```
for var in sequence:
    statement(s)
    ……
    ……
        ……
```

The for loop is a Python statement which repeats a group of statements for a specified numberof times. As described in the syntax, the keywords for and in are essential keywords to iterate the sequence of values. The variable var takes on each consecutive value in the sequence and the statements in the body of the loop are executed once for each value. A simple example of for loop is:

```
for var in range(m,n):
    print var
```

```
for j in range(1,6):
    print(j)
print("End of The Program")
```

Output

```
1
2
3
4
5
End of The Program
```

```
for j in range(1,6):
    square=j*j
    print("Square of ",j," is: ",square)
print("End of Program")
```

Output

```
Square of 1 is:  1
Square of 2 is:  4
Square of 3 is:  9
Square of 4 is:  16
Square of 5 is:  25
End of Program
```

```
sum=0
print("Even numbers from 0 to 10 are as follows")
for j in range(0,11,1):
    if i%2==0:
        print(j)
        sum=sum+j
print("Sum of Even numbers from 0 to 10 is = ",sum)
```

Output

```
Even numbers from 0 to 10 are as follows
0
2
4
6
8
10
Sum of Even numbers from 0 to 10 is = 30
```

Q.8} Explain nested loops and write a program to display multiplication table from 1 to 10.

Ans.

The for and while loop statements can be nested in the same manner in which the if statements are nested. Loops within the loops or when one loop is inserted completely within another loop, then it is called nested loop.

```
Print("Multiplication Table from 1 to 5 ")
for j in range(1,11,1):              #Outer Loop
    for j in range(1,6,1):           #Inner Loop
```

(*Contd.*


```
        print(format(j * j,"4d"),end=" ")
    print()
print("End of Program")
```

Output

```
Multiplication Table from 1 to 5
   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
   6   12   18   24   30
   7   14   21   28   35
   8   16   24   32   40
   9   18   27   36   45
  10   20   30   40   50
End of Program
```