Unit - II

Control Structures, Lists, Dictionaries and Sets

Control Structures:

A control statement is a statement that determin the control flow of a set of instructions. There are three fundamental forms of control the programming languages provide

1. Sequence control
2. Selection control
3. Iterative control

Control statements are statements which control or change the flow of execution. The following are the control statements available in Python.

1. if statement.
2. if - else statement.
3. if - ~~else~~ elif - else statement.
4. while loop.
5. for loop.
6. else suite.
7. break statement.
8. continue statement.
9. pass statement.

10. assert statement

11. return statement.

1. The if statement:

This statement is used to execute one or more statements depending on whether a condition is true or not.

Syn: if condition:

statement.

Ex: 1. A Python Program to express a digit in a word.

Sol:
```
num = 1
if num == 1:
    print ("one")
```

O/P: one.

2. A Python program to display a group of messages when the condition is true.

Sol:
```
str = 'yes'
if str == 'yes':
    print ("Hi \n")
    print ("welcome \n")
    print ("Good morning \n")
```

O/P: Hi
welcome
Good morning.

2. The if-else Statement:

The if-else statement executes a group of statements when a condition is True, otherwise it will execute another group of statements i.e else.

Syn:
```
if condition:
    statement 1
else:
    statement 2
```

Ex: 1. A Python program to test whether a given number is even or odd.

Sol:
```
x = 10
if x%2 == 0:
    print(x, "is even number")
else:
    print(x, "is odd number")
```

o/p: 10 is even number.

2. A Python program to test whether a person is eligible for vote.

Sol:
```
x = int(input("Enter ur age"))
if age>18:
    print("Person is eligible for vote")
```

```
    else :
        Print ("Person is not eligible for vote")
```

o/p : enter ur age : 24

Person is eligible for vote.

3. The if...elif...else Statement :

Sometimes, the Programmer has to test multiple conditions and execute statements depending on those conditions.

Syn :
```
if condition1:
    statement1
elif condition2:
    statement2
elif condition3:
    statement3
else:
    statement4
```

Ex : 1. A python program to know if a given no is zero, positive or negative.

Sol :
```
num = int (input ("Enter no"))
if num == 0:
    Print (num, "is zero")
elif num > 0:
    Print (num, "is positive")
```

```
else:
    Print (num, "is negative")
```

op: Enter the no : -5

-5 is negative

2. A Program to accept a numeric digit from keyboard and displays in words.

Sol:
```
x = int (input ("Enter a digit"))
if x==0:
    Print ("zero")
elif x==1:
    Print ("one")
elif x==2:
    Print ("Two")
:
elif x==9:
    Print ("Nine")
```

op: Enter a digit: 2

Two

Note: else Part is not compulsory in if... elif... else Statemen

The break statement:

The break statement can be used inside a for loop or while loop to come out of the loop. when 'break' is executed, the python interpreter jumps out of the loop to proceed the next statement in the program.

Ex: 1. A Python Program to display numbers from 10 to 6 and break the loop when the number about to display is 5.

Sol:
```
x=10
while x>=1:
    print('x=',x)   # print('x=',x,end=" ")
    x=x-1
    if x==5:
        break
print("out of loop")
```

o/p:
```
x=10
x=9
x=8
x=7
x=6
out of loop
```

The Continue Statement:

The Continue statement is used in a loop to go back to the beginning of the loop. It means, when Continue is executed the next repitition will start. when continue is executed, the subsequent statements in the loop are not executed.

Ex: 1. A Python Program to display numbers from 1 to 5 using continue statement.

Sol:
```
x = 0
while x < 10:
    x = x + 1
    if x > 5:
      continue
    print("x =", x)
print("out of the loop")
```

O/P:
```
x = 1
x = 2
x = 3
x = 4
x = 5
out of the loop
```

The Pass Statement:

The Pass statement does not do anything. It is used with 'if' statement or inside a loop to represent no operation. We use Pass statement when we need a statement syntactically but we do not want to do any operation.

Ex: 1 A Program to know that Pass does nothing.

Sol:
```
x=0
while x<5:
    x=x+1
    if(x>3):
        Pass
    Print('x=',x)
Print("out of loop")
```

O/P:
```
x=1
x=2
x=3
x=4
x=5
out of loop
```

2. A python program to retrieve only -ve numbers from a list of numbers.

Sol:
```
num=[1,2,3,-4,-5,-6,-7,8,9]
```

```
for i in num:
    if (i>0):
      pass
    else:
      print(i)
```

o/p: -4
-5
-6
-7

The while loop:

A loop is useful to execute the statement repeatedly. It is a pre-test loop. The while loop is useful to execute a group of statements several times repeatedly depending on whether a condition is True or False. The syntax of while loop is as follows

Syn: while condition:
    Statements.

Here the statements represents one statement or a set of statements. Python interpreter first checks the condition. If the condition is True, then it will execute the statements written after colon (:). After executing the statements, it will go back and check the condition again. If the condition is again found to be True, then it will again execute the statements. In this way, as long as

the condition is True, Python interpreter executes this statement again and again. once the condition is found to be False, then it will come out of the while loop.

Ex: 1. A Python Program to display no's from 1 to 10 using while loop

Sol:
```
x = 1
while x <= 10:
    Print(x)
    x += 1
Print("End")
```

o/p:
```
1
2
:
10
End
```

2. A Python Program to display even no's blw 100 and 200.

Sol:
```
x = 100
while x >= 100 and x <= 200:
    Print(x)
    x += 2
```

o/p:
```
100
102
:
200
```

The for loop:

The for loop is useful to iterate over the elements of a sequence. It means, the for loop can be used to execute a group of statements repeatedly depending upon the number of elements in the sequence. The for loop can work with sequence like string, list, tuple, range etc.

Syn: for var in sequence:
    statements

The first element of the sequence is assigned to the variable written after 'for' and then statements are executed. Next, the second element of the sequence is assigned to the variable and then the statements are executed second time. In this way, for each element of the sequence, the statements are executed once. So, the for loop is executed as many times as there are no. of elements in the sequence.

Ex: 1. A python program to display characters of a string using for

Sol:
```
str = "Hello"
for ch in str:
    print(ch)
```

O/P:
```
H
e
l
l
o
```

We can use range(n) object that generates numbers from 0 to n-1.

Ex: 1 A Python Program to display each character from a string using sequence index.

Sol:
```
str = 'Hello'
n = len(str)
for i in range(n):
    print(str[i])
```

O/P:
```
H
e
l
l
o
```

The range() object is also known as range() function, in Python is useful to provide a sequence of numbers. range(n) gives numbers from 0 to n-1. For ex, if we write range(10), it will return numbers from to 0 to 9. we can also mention the starting number, ending number, as: range(5,10). In this case, it will give numbers from 5 to 9. we can also specify the step size. The step size represents the increment in the value of the variable at each step. For ex, range(1,10,2) will give no's from 1 to 9 in steps of 2.

Ex: 1. A Python Program to display odd numbers from 1 to 10 using range() function.

Sol:
```
for i in range(1,10,2):
    print(i)
```

O/P:
```
1
3
5
7
9
```

2. A Program to display numbers from 10 to 1 in decending order

Sol:
```
for x in range(10,0,-1):
    print(x)
```

O/P:
```
10
9
8
:
1
```

3. A Program to display the elements of a list using for loop.

Sol:
```
list = [10, 20.5, 'A', 'India']
for i in list:
    print(i)
```

O/P:
```
10
20.5
A
India
```

4. A Python program to display and find the sum of a list of numbers using for loop.

Sol:
```
list = [10, 20, 30, 40, 50]
Sum = 0
for i in list:
    Print(i)
    Sum = Sum + i
Print ('sum =', sum)
```

O/P:
```
10
20
30
40
50
Sum = 150
```

Nested Loops:

It is possible to write one loop inside another loop.

Ex:
```
for i in range(2):
    for j in range(2):
        Print ('i=', i, 'If' 'j=', j)
```

O/P:
```
i = 0  j = 0
i = 0  j = 1
i = 1  j = 0
i = 1  j = 1
```

2. A Python Program that displays stars in right angled triangle form using nested loops.

Sol :
```
for i in range(1,6):
    for j in range(1, i+1)
        print('* ', end='')
    print()
```

o/p :
```
*
* *
* * *
* * * *
* * * * *
```

The else suit :

In python, it is possible to use 'else' statement along with for loop or with while loop in the following form.

Syn : for with else

```
for var in Sequence:
    Statements
else:
    Statements
```

while condition:

```
    Statements
else:
    Statements
```

Ex: 1. Write a Program to Print Yes and No.

Sol:
```
for i in range(5):
    Print ("Yes")
else:
    Print ("No")
```

O/P:
```
Yes
Yes
Yes
Yes
Yes
No
```

2. A Python Program to Search for an element in the list of element.

Sol:
```
list = [1, 2, 3, 4, 5]
Search = int (input ("Enter Key element"))
for i in list:
    if Search == i :
        Print ("Element found in the list")
        break
← else:
        Print ("Element not found in the list")
```

O/P: Enter Key element : 4

Element found in the list.