PYTHON

Looping : A loop is a process where the statements are executed one or more times. There are three looping methods for executing loops. All methods work on same functionality called executing every statement and the syntax is different with one another.

Different looping statements are:

- While loop
- For loop
- Nested loop

While loop : while loop is used to execute a block of statements repeatedly till the condition specified is **True**. If the condition become **False** then statements that are immediately after the loop will be executed.

- If there are no statements after loop then the process will be terminated.

Syntax:

```
While condition:
    Statements
```

- First evaluate the condition and check condition True or False.
- If the condition is False go to next statements after the loop and execute them.
- If the condition is True execute the while block statements and again go to condition for checking.

Example:

```
n=10
sum =0
i=1
while i<n:
    sum = sum + i
```

```
        i = i + 1

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

- It first check i<n condition if it is True . It will make sum action and increment i value by 1 and print the sum.

- Once the value of i become 10 the condition will be False and as there are no statements after while loop the execution will be terminated.

While using Else : Once the while condition is False the process goes to statements after loop where we can include else block with statements.

- else block statements will be executed only when the while condition is False.

Example:

```
n=10

sum=0

i=1

while i<n:

        sum = sum + i

        i = i + 1

        print ("The sum is" , sum)

else:

        print ("i is greater than/equal to n")
```

- After reaching i=10 as the condition is false it will go to else block and execute and return "i is greater than/equal to n".

Single statement while block : When the while block contain single line statement then we can write the complete loop in a single line.

Example:

```
language='python'
while(language=='python'): print('python is
                                Good language")
```

- It is recommended not to use single line statement become the loop will infinite as the condition is always **True**.

For loop : This loop processes through each items present in sequence, due to which it is mostly used in python sequence i.e,(list , strings , tuples).

- At a time it process single item and goes to body of the loop for execution and goes to next item in sequence.

- The loop stops once the last iteration in sequence is reached.

Syntax:

```
for items in sequence:
    statements
```

Example:

```
L=["red","green","black"]
for i in L:
    print (i)
```

- Here L is a list with different colours and for loop is used to iterate over the list(L).

- It will provide the output as red green black each in a new line.

- We can use this **for loop** to iterate over range function.

Iterating by index of sequences : Indexing can be used for iterating in the sequence. To do this first need to calculate length of sequence and iterate over the sequence within the range of the length.

Example:

```
L=["red","green","black"]
```

```
for i in range(len(L)):
    print (L(i))
```

- It will calculate the length of list and by using range function it is going to iterate over the list.

Else statement with for loop : As used in the **while loop** we can use **else** in **for loop** also that provide same functionality but in **for loop** there is no **condition**. So, **else block** will be executed only when all the for loop iteration are executed.

Example:

```
L=["red","green","black"]
for i in range(len(L)):
    print (L(i))
else:
    print "iterator reaches else block"
```

for with range : range is a in built function in python that is used for iteration with specifying the condition.

- range(start , stop , jump) which indicates where we can specify the start and ending with number of jump to be performed.

- If we specify on one argument in range function it indicated the end point of the range.

Syntax:

```
for items in range(start , stop , jump):
    Statements
```

Example:

```
for num in range(3,20,3)
    print (num)
```

· This will give output as 3,6,9,12,15,18 as iteration starts from 3 till 20 with 3 jumps at a time.

Nested Loop : Loop written inside another loop is called nested loop. The process of this nested loop is as follows:

· The outside loop will be iterated and goes to inside loop and inside is iterated and statements in inside loop block will be executed.

· All inside iteration will be performed and after completing it goes to outside loop for next iteration.

· This continuous till outer loop completes iteration.

Syntax for nested for loop:

```
for items in sequence:

    for items in sequence:
        statements
        statements
```

Syntax for nested while loop:

```
while condition:

    while condition:

        statements

        statements
```

` · If the outer condition in while loop is False it will not execute inner while loop and direct goes to next statement after loop.

· Inner while loop will be executed only when outer condition is True.

· if the condition is True it checks inner condition if inner condition is True then inner statements will be executed otherwise next statement after inner loop will be executed if any.

Note: We can put a while loop inside a for loop (or) a for loop inside a while loop also (or) any loop inside any other loop.

Example:

```
country=['india','pakistan','srilanka']
player = ["Kapildev" , "Akthar" , "Jayasurya"]
for i in country:
    print (i)
    for j in player:
        print (j)
```

- First it iterate the country list and then goes to player list.
- It print all player names and goes back to country list for next iteration and again prints player names until iteration in country list are executed.