03/02/2022

2.Iterative statements:-

If we want to execute a group of statements multiple times then we should go for iterative statements.

python supports 2 types of iterative statements.

1.for loop
2.while loop

1.for loop:-

If we want to excute some action for every element present in some sequence (it may be string or collection) then we should go for for loop.

syntax:-

for i in variable/sequence:
statements

ex:-

```
l = [1,9+8j,79.890,[1,2,3],(8,9,7),'uk,usa',{"n":"jk"}]
for i in l:
print(i)
```

```
1
(9+8j)
79.89
[1, 2, 3]
(8, 9, 7)
uk,usa
{'n': 'jk'}
```

where sequence can be string or any collection
Body will be executed for every element present in the sequenece.

```
l = [1,9+8j,79.890,[1,2,3],(8,9,7),'uk,usa',{"n":"jk"}]
for i in l:
print(i)
```

```
1
(9+8j)
79.89
[1, 2, 3]
(8, 9, 7)
uk,usa
{'n': 'jk'}
```

```
l[0]
1
```

```
i[0]
Traceback (most recent call last):
File "<pyshell#6>", line 1, in <module>
i[0]
KeyError: 0
i
{'n': 'jk'}

l = [1,2,3,4,5]
for i in /l
SyntaxError: invalid syntax
for i in l:
a = i+i
print(a)
```

```
2
4
6
8
10
```

manipulation

```
a = []
for i in l:
b = i+i
a.append(b)
```

```
a
[2, 4, 6, 8, 10]
```

###range data type(range()):-

It's genereting the sequence and follwed the indexing rule.

(start,end+1,step)

```
#range
range(10)
range(0, 10)
for i in range(10):
print(i)
```

```
0
1
2
3
4
5
6
```

7
8
9

#Even

```
for i in range(1,21):
if i%2==0:
print(i)
```

2
4
6
8
10
12
14
16
18
20

#reverse numbers:-

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

10
9
8
7
6
5
4
3
2
1

#even and odd

```
for i in range(10):
if i%2==0:
print('even',i)
if i%2!=0:
print('odd',i)
```

even 0
odd 1
even 2
odd 3
even 4
odd 5

even 6
odd 7
even 8
odd 9

2.while loop:-

If we want to execute a group of statements iteratively unitl some condition false,then should go for while loop.

syntax:-

```
while condition:
statement
```

#To print numbers from 1 to 10 by using while loop.

```
x = 1
while x<=10:
print(x)
x+=1
```

1
2
3
4
5
6
7
8
9
10

#To display the sum of first n numbers:-

```
n = int(input("Enter your number"))

sum = 0
intial = 1

while intial<=n:
sum =sum+intial
intial = intial+1
print("The sum of first",n,"number is:",sum)
```

3.Transfer statement:

1.break:- we can use break statement inside loop execution based on some condition.

ex:-

```
for i in range(11):
if i == 7:
```

```
break
print(i)
```

```
0
1
2
3
4
5
6
```

2.continue:- we can use continue statement to skip current iteration and continue next iteration.

To print odd number in range(0,20)

```
for i in range(21):
if i%2 ==0:
continue
print(i)
```

```
1
3
5
7
9
11
13
15
17
19
```

3.pass:- Just pass this block of code.

pass is a keyword in python

In our programming syntactically if block is required whicj won't do anything then we can define that empty block with pass keyword.

pass:-

It is an empty statement
It is null statement
it won't do anything

ex:-

```
if True:
```

```
File
"C:\Users\chand\PycharmProjects\testing_17_18_19_python\FlowControl\confitional_statements.py",
line 111
```

```
^
```

IndentationError: expected an indented block

```
if True:
pass
```

Process finished with exit code 0