Loops in Python

Loops enable you to perform repetitive tasks efficiently without writing redundant code. They iterate over a sequence or execute a block of code as long as a specific condition is met.

Loops are used to execute a block of code repeatedly until a certain condition is met.

Python mainly provides two types of loops.

1) for loop
   - i) for loop with range
   - ii) for loop with sequence.
2) while loop
3) nested loops
   - a) for loop inside for loop
   - b) for loop inside while loop
   - c) while loop inside while loop
   - d) while loop inside for loop.

For loop

Syntax:

```
for i in sequence_name:    # i iteration variable.
    statements
```

eg language = 'Python'

```
for i in language:
    print(i)
```

o/p

```
P
y
t
h
o
```

Using range() Function :

To repeat a block of code a specified number of times, we use the range() function. The range() function returns a sequence of numbers, starting from 0 by default, increments by 1 (by default) and stops before a specified number.

Syntax : range(stop)
range(start, stop)
range(start, stop, step).

eg for i in range(5):
print(i) o/p - 0 1 2 3 4

for i in range(1,10,2):
print(i) o/p 1 3 5 7 9

2- While loop : repeatedly executes a block of code as long as a given condition remains True. It checks the condition before each iteration.

Syntax : while condition:
statements

eg. print numbers from 0 to 3
count = 0 o/p 0 1 2 3
while count < 4:
print(count)
count += 1

else statement : An else clause can be added to loops. It executes after the loop finishes normally

count = 3
while count > 0:
print("countdown:", count)
else:
print("Liftoff!")

single statement while block.

count = 0 o/p Hello Geek
while (count < 5): Hello Geek
count += 1; Hello Geek
print("Hello Geek") Hello Geek.
Hello Geek.

while loop vs for loop

while loop - keeps running as long as condition is true. generally used when you don't know how many iterations will be need beforehand and loop continues based on a condition.

for loop : iterates over a sequence and runs the loop for each item in that sequence. It is used when you know in advance how many times you want to repeat a block of code

Control flow statements

Loop control statements :- allow you to alter the normal flow of a loop.

- → pass statement
- → break statement
- → continue statement

1) **Pass statement** : used as a placeholder (it does nothing) for the future code, and runs entire code without causing any syntax error

```
eg. for i in range(5):
        statements
        pass
```

2) **break statement** - terminates the loop entirely, exiting from it immediately.

```
eg. for i in range(5):
        if i == 3:
            break
        print(i)
```

o/p
0 1 2

the loop terminated when condition true i=3

3) **continue statement**.

skips the current iteration and moves to the next one

```
for i in range(5):
    if i == 3
        continue
    print(i)
```

o/p
0 1 2 4

loop skips when i==3

3 **Nested loops** : Loop inside another loop is nested loop. For every single time outer loop runs, inner loop runs all of its iterations

used for :) Handling Multi-Dimensional Data
- :) Complex iteration
- :) Pattern Generation.

Syntax

```
for i in range():  # outer loop
    for j in range():  # inner loop.
        statements of inner loop
    statements of outer loop
```

```
for i in range(1,6,1):
    for j in range(1,6,1):
        print(i, end=" ")
    print()
```

o/p
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5

```
for i in range(1,6,1):
    for j in range(1,6,1):
        print(j, end=" ")
    print()
```

o/p
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5.