For Loop in Python

Loops in Python for Data Science

Loops are essential in Python for automating repetitive tasks, which is particularly useful in data science for processing datasets, running simulations, and training models. Python provides two primary types of loops: for and while.

1. Types of Loops in Python

For Loop

A for loop is used when we want to iterate over a sequence (like lists, tuples, dictionaries, sets, or strings).

```
for variable in iterable:
    # Code block to execute
```

- It executes a block of code **for each item** in the sequence.

Basic Syntax:

for variable in sequence:

**   # Code to execute**

- variable → Takes the value of each item in the sequence **one by one**.
- sequence → Can be a list, string, range, etc.
- The loop runs **once for each item** in the sequence.

Example 1: Looping Through a List

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Output:

```
apple
banana
```

cherry

➡ The loop picks each fruit from the list and prints it.

Example 2: Using range() with for loop

- range(n) generates numbers from 0 to n-1.

```
for i in range(5):
   print(i)
```

Output:

```
0
1
2
3
4
```

➡ The loop runs 5 times, printing numbers from 0 to 4.

Example 3: Looping Through a String

```
for letter in "Python":
   print(letter)
```

Output:

```
P
y
t
h
o
n
```

➡ Each character in the string is printed separately.

Example 4: Looping Through a Dictionary

```
student = {"name": "John", "age": 20, "grade": "A"}
for key, value in student.items():
   print(key, ":", value)
```

Output:

```
name : John
age : 20
grade : A
```

➡ Loops through **keys and values** in the dictionary.

Using else with for

```
for i in range(3):
    print(i)
else:
    print("Loop finished!")
```

Output:

```
0
1
2
Loop finished!
```

➡ The else block runs when the loop **completes normally**.

While Loop

A while loop runs as long as a condition is True.

Syntax:

```
while condition:
    # Code block to execute
```

Example: Loop Until a Condition is Met

```
count = 0
while count < 5:
    print(count)
    count += 1  # Increment to avoid infinite loop
```

Output:

```
0
1
2
```

3
4

2. Loop Control Statements

(A) Break Statement

Stops the loop immediately.

Example: Stop when number is 3

```
for num in range(5):
    if num == 3:
        break
    print(num)
```

Output:

0
1
2

(B) Continue Statement

Skips the current iteration and moves to the next.

Example: Skip number 3

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

Output:

0
1
2
4

(C) Pass Statement

Acts as a placeholder when no action is needed.

Example: Using Pass in a Loop

```
for num in range(5):
    if num == 3:
        pass  # Does nothing
    print(num)
```

Output:

```
0
1
2
3
4
```

3. Nested Loops

A loop inside another loop.

Example: Looping Through a Matrix

```
matrix = [[1, 2], [3, 4]]
for row in matrix:
    for num in row:
        print(num)
```

Output:

```
1
2
3
4
```