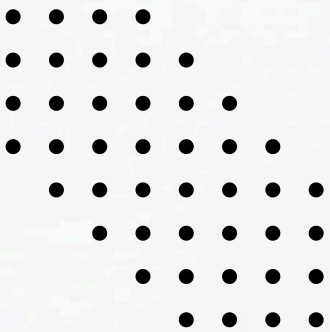
TOPSTechnologies

Looping In Python

Presented for :

TOPs Technologies

Presented by :
Sumit B Yadav



Que 1

Introduction to For Loop

for Loops

A for loop is used for iterating over a sequence (such as a list, tuple, dictionary, set, or string) or other iterable objects. It is typically used when you know in advance how many times you want to execute a statement or a block of statements.

Basic Syntax Of For Loops -

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

while Loops

A while loop executes a set of statements as long as a condition is true. It is useful when the number of iterations is not known in advance and depends on some condition.

Basic Syntax Of While Loops-

```
count = 0
while count < 5:
print(count)
count += 1
```

Que.2

How Loops works in python -

Loops in Python allow you to execute a block of code repeatedly based on a condition or for each item in a sequence. They are powerful constructs that enable efficient and concise coding for repetitive tasks.

How for Loops Work-

A for loop in Python iterates over the elements of a sequence (such as a list, tuple, dictionary, set, or string) or other iterable objects. During each iteration, the loop variable is assigned the next value from the sequence, and the code block within the loop is executed.

Example: Using range()

The range() function generates a sequence of numbers, which is particularly useful for iterating a specific number of times.

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

How while Loops Work

A while loop executes a block of code as long as a specified condition is true. The condition is evaluated before each iteration, and if it evaluates to True, the loop continues; otherwise, it stops.

Que.3

Using for Loop with Lists --

A for loop is commonly used to iterate over elements in a list.

SYNTAX-

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

Using while Loop with Lists

You can also use a while loop to iterate over a list using an index.

SYNTAX-

```
fruits = ["apple", "banana", "cherry"]
index = 0
while index < len(fruits):
print(fruits[index])
index += 1
```

Using for Loop with Tuples--

A for loop can iterate over the elements in a tuple in the same way as it does with a list.

BASIC SYNATX--

```
colors = ("red", "green", "blue")
for color in colors:
print(color)
```

Using while Loop with Tuples--

You can use a while loop to iterate over a tuple similarly to a list.

BASIC SYNTAX--

```
colors = ("red", "green", "blue")
index = 0
while index < len(colors):
print(colors[index])
index += 1
```

Using for Loop with Dictionaries--

A for loop can be used to iterate over the keys, values, or key-value pairs in a dictionary.

BASIC SYNTAX--

```
person = {"name": "Alice", "age": 30, "city": "New York"}
for key in person:
print(key)
```

Using while Loop with Dictionaries

While it's less common, you can use a while loop with dictionaries, typically by iterating over the keys or converting keys to a list first.

BASIC SYNTAX--

```
person = {"name": "Alice", "age": 30, "city": "New York"}
keys = list(person.keys())
index = 0
while index < len(keys):
key = keys[index]
print(f"{key}: {person[key]}")
index += 1
```

Using for Loop with Sets--

A for loop can iterate over the elements in a set.

BASIC SYNTAX--

```
unique_numbers = {1, 2, 3, 4, 5}
for number in unique_numbers:
print(number)
```

Using while Loop with Sets--

It's uncommon to use a while loop with sets because sets are unordered collections and don't support indexing. However, you can convert a set to a list first.

BASIC SYNTAX--

```
unique_numbers = {1, 2, 3, 4, 5}
numbers_list = list(unique_numbers)
index = 0
while index < len(numbers_list):
    print(numbers_list[index])
    index += 1
```