Introduction

A `for` loop in Python is used to iterate over a sequence such as a list, tuple, dictionary, set, or string. It allows execution of a block of code multiple times, once for each item in the sequence.

Syntax

```python
for variable in sequence:
 # Code block to execute
```

- `variable` takes the value of each item in `sequence`.
- The loop continues until all elements in `sequence` are processed.

Examples

1. Iterating Over a List

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

Output:

```
apple
banana
cherry
```

2. Using `range()` to Loop a Fixed Number of Times

```python
for i in range(1, 6): # Loops from 1 to 5
 print(i)
```

Output:

```
```

1

2

3

4

5

```

#### 3. Iterating Over a String

```python

word = "Python"

for letter in word:

 print(letter)

```

Output:

```

P

y

t

h

o

n

```

4. Iterating Over a Dictionary

```python

person = {"name": "Alice", "age": 25, "city": "New York"}

for key, value in person.items():

 print(key, ":", value)

```

Output:

```

name : Alice

age : 25

city : New York

```

5. Nested `for` Loop

```python

for i in range(1, 4):

 for j in range(1, 4):

 print(f"i={i}, j={j}")

```

Output:

```

i=1, j=1

i=1, j=2

i=1, j=3

...

i=3, j=3

```

Using `break` and `continue`

- `break` stops the loop.

- `continue` skips the current iteration.

```python

for num in range(1, 6):

 if num == 3:

 break # Stop when num is 3

 print(num)

```

Output:

```

1

2

```

```python

```
for num in range(1, 6):
 if num == 3:
 continue # Skip when num is 3
 print(num)
```

Output:

```
1
2
4
5
```

Using `else` with `for` Loop

The `else` block runs when the loop completes normally (without `break`).

```python
for num in range(1, 4):
 print(num)
else:
 print("Loop finished!")
```

Output:

```
1
2
3
Loop finished!
```

Conclusion

- `for` loops iterate over sequences.
- `range()` helps loop a fixed number of times.

- `break` and `continue` control loop execution.
- `else` executes if the loop completes without `break`.

This knowledge of `for` loops will help you write efficient Python programs. Happy coding!