The for Loop

In Python, the `for` loop is used to iterate over a sequence (such as a list, tuple, string, or range) or other iterable objects. The `for` loop is often used when the number of iterations is known or can be determined in advance. Here are the details along with examples:

Basic Syntax:

```python
for variable in iterable:
    # code to be executed in each iteration
```

- The `iterable` is any object capable of producing its elements one at a time.
- The `variable` takes on the value of each element in the sequence during each iteration.

Example 1: Iterating Over a List

```python
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

Output:
```
apple
banana
cherry
```

Example 2: Iterating Over a String

```python
message = "Hello, Python!"

for char in message:
    print(char)
```

Output:
```
H
e
l
l
o
```

,

P
y
t
h
o
n
!
```

### Example 3: Iterating Over a Range

```python
for num in range(1, 6):
    print(num)
```

Output:
```
1
2
3
4
5
```

Example 4: Using `enumerate` for Index and Value

```python
fruits = ["apple", "banana", "cherry"]

for index, fruit in enumerate(fruits):
    print(f"Index: {index}, Fruit: {fruit}")
```

Output:
```
Index: 0, Fruit: apple
Index: 1, Fruit: banana
Index: 2, Fruit: cherry
```

Example 5: Nested Loops

```python
for i in range(3):
    for j in range(2):
        print(f"({i}, {j})")

```

Output:
```
(0, 0)
(0, 1)
(1, 0)
(1, 1)
(2, 0)
(2, 1)
```

Example 6: Using `break` and `else` with a `for` loop

```python
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num == 3:
        print("Found 3!")
        break
else:
    print("3 not found in the list.")
```

In this example, the `else` block is executed if the `for` loop completes without encountering a `break`. It is not executed if the loop is terminated prematurely by a `break` statement.

Example 7: Using `zip` to Iterate Over Multiple Lists

```python
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]

for name, age in zip(names, ages):
    print(f"{name} is {age} years old.")
```

Output:
```
Alice is 25 years old.
Bob is 30 years old.
Charlie is 35 years old.
```

The `for` loop is a powerful and versatile construct in Python, providing an elegant way to iterate over elements in a sequence or other iterable objects. It is widely used in various programming scenarios for data processing, manipulation, and control flow.
```