Loops in Python

Loops allow you to repeat a block of code multiple times. Python supports two main types of loops: `for` loops and `while` loops.

1. `for` Loop

The `for` loop is used to iterate over a sequence (like a list, tuple, dictionary, set, or string) and execute a block of code for each item in that sequence.

Syntax:

```
for variable in sequence:
    # Block of code to execute
```

Example:

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

This loop will print each fruit from the list.

2. `while` Loop

The `while` loop continues to execute a block of code as long as a condition is `True`.

Syntax:

```
while condition:
    # Block of code to execute
```

Example:

```
i = 1
while i < 5:
    print(i)
    i += 1  # Increment the counter to avoid infinite loop
```

This loop will print numbers from 1 to 4.

3. `break` and `continue` Statements

`break`:

The `break` statement is used to exit the loop entirely, even if the condition is still `True` in the case of a `while` loop, or items are left to iterate over in a `for` loop.

Example:

```python
for i in range(1, 6):
    if i == 3:
        break  # Exit loop when i equals 3
    print(i)
```

This will print `1` and `2`, then exit the loop.

`continue`:

The `continue` statement is used to skip the current iteration and continue with the next one.

Example:

```python
for i in range(1, 6):
    if i == 3:
        continue  # Skip the current iteration when i equals 3
    print(i)
```

This will print `1`, `2`, `4`, and `5` (skips `3`).

4. Nested Loops

A nested loop is a loop inside another loop.

Example:

```python
for i in range(1, 4):
    for j in range(1, 3):
        print(f"i: {i}, j: {j}")
```

This will print all combinations of `i` and `j`.