Loops in Python

1. What are Loops?

A **loop** is a programming construct used to repeat a block of code multiple times, either for a set number of iterations or until a specific condition is met. Loops are fundamental in automating repetitive tasks.

2. Types of Loops in Python

Python has two primary types of loops:

1. **`for` loop** – Iterates over a sequence (e.g., list, string, range).
2. **`while` loop** – Repeats as long as a specified condition is `True`.

3. `for` Loop

The `for` loop is used to iterate over an iterable (e.g., list, tuple, dictionary, string, or range).

Syntax:

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

Example: Iterating over a list

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

Example: Using `range()`

```
# Print numbers from 0 to 4
for i in range(5):
    print(i)
```

Example: Iterating over a string

```
word = "Python"
for char in word:
    print(char)
```

4. `while` Loop

The `while` loop is used when the number of iterations is not known in advance. It repeats as long as the condition remains `True`.

Syntax:

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

Example: Counting with a condition

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

Caution: Ensure the loop condition eventually becomes `False`, or the loop will run infinitely.

5. Loop Control Statements

These statements modify the behavior of loops:

1. **`break`**: Terminates the loop prematurely.
2. **`continue`**: Skips the rest of the code in the current iteration and moves to the next iteration.
3. **`pass`**: Does nothing; a placeholder for empty loops.

Examples:

```
# Using break
for num in range(10):
    if num == 5:
        break
    print(num)

# Using continue
for num in range(5):
    if num == 2:
        continue
    print(num)

# Using pass
for num in range(3):
    pass  # No operation performed
```

6. Nested Loops

A loop inside another loop is called a **nested loop**. These are useful when working with multi-dimensional data (e.g., matrices).

Example:

```python
for i in range(3):  # Outer loop
    for j in range(2):  # Inner loop
        print(f"i = {i}, j = {j}")
```

7. The `else` Clause in Loops

In Python, loops can have an optional `else` block. The code inside the `else` block executes when the loop terminates **naturally** (i.e., not via `break`).

Example with `for` loop:

```python
for num in range(5):
    if num == 3:
        break
    print(num)
else:
    print("Loop completed without break.")
```

Example with `while` loop:

```python
count = 0
while count < 3:
    print(count)
    count += 1
else:
    print("Condition no longer true.")
```

8. Infinite Loops

An infinite loop occurs when the condition of a `while` loop never becomes `False`. Use with caution.

Example:

```python
while True:
    print("This will run forever unless stopped!")
```

To exit an infinite loop, use `break` or interrupt the program manually.

9. Practical Applications of Loops

- **Data Processing**: Iterating over files, lists, or database records.
- **Automation**: Automating repetitive tasks like printing, calculations, or sending emails.
- **Games**: Updating player positions, scores, or other game elements.
- **Simulations**: Simulating iterative processes like physics or economics models.

10. Summary Table

Loop Type	When to Use	Example Code
`for` loop	When you know the number of iterations.	`for i in range(5): print(i)`
`while` loop	When the number of iterations is unknown.	`while count < 5: count += 1`
`break`	To exit a loop prematurely.	`if condition: break`
`continue`	To skip the current iteration.	`if condition: continue`
`pass`	To do nothing, e.g., a placeholder for the loop.	`for i in range(5): pass`
Nested loops	When working with multi-dimensional data.	`for i in range(3): for j in range(2): print()`

11. Exercises

1. Write a `for` loop that prints all even numbers between 1 and 20.
2. Write a `while` loop that keeps asking for user input until they type "exit".
3. Use nested loops to create a multiplication table from 1 to 10.
4. Write a program that uses `break` to stop a loop when the sum of numbers exceeds 50.