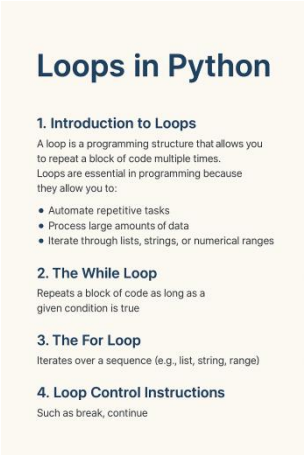
Chapter 4:

Loops in Python



1. Introduction to Loops

A **loop** is a programming structure that allows you to **repeat a block of code multiple times**. Loops are essential in programming because they allow you to:

- Automate repetitive tasks
- Process large amounts of data
- Iterate through lists, strings, or numerical ranges
- Control program flow efficiently

In Python, there are **two main types** of loops:

1. **while loop**
2. **for loop**

Python also provides **control statements** such as:

- break
- continue

2. The While Loop: Syntax and Working Mechanism

A **while loop** repeats a block of code **as long as a given condition is true**.

Syntax

```
while condition:
    # block of code
```

How it works

1. Python checks the **condition**.

2. If it is **True**, the code inside the loop executes.
3. After execution, Python goes back to check the condition again.
4. The loop stops when the condition becomes **False**.

Example 1 — Counting from 1 to 5

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

Output:

```
1
2
3
4
5
```

Important: Infinite Loop

If the condition never becomes False, the loop runs forever.

Example of **infinite loop**:

```
i = 1
while i <= 5:
    print(i)
```

Example 2 — Sum of numbers from 1 to N

```
n = int(input("Enter a number: "))
i = 1
total = 0

while i <= n:
    total += i
    i += 1

print("Sum =", total)
```

3. The For Loop: Syntax and Iteration Over Sequences

A **for loop** is used to iterate over a **sequence**, such as:

- lists
- strings
- ranges
- tuples
- dictionaries

Syntax

```
for variable in sequence:
    # block of code
```

A. Using range()

Example 1 — Print numbers from 1 to 5

```
for i in range(1, 6):
    print(i)
```

B. Iteration over a List

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

C. Iteration over a String

```
for letter in "CHEMISTRY":
    print(letter)
```

D. Using a For Loop to Calculate Factorial

```
n = int(input("Enter a number: "))
fact = 1
for i in range(1, n + 1):
    fact *= i
print("Factorial =", fact)
```

4. Loop Control Instructions (break, continue)

Python provides statements that control the loop execution.

A. break Statement

Stops the loop **immediately**, even if the condition or sequence is not finished.

Example — Stop when number reaches 5

```
for i in range(1, 11):
    if i == 5:
        break
    print(i)
```

Output:

```
1
2
3
4
```

B. continue Statement

Skips the current iteration and continues with the next one.

Example — Skip the number 5

```
for i in range(1, 11):
    if i == 5:
        continue
    print(i)
```

Output:

```
1
2
3
4
6
7
8
9
10
```

C. Example with while loop

```
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
```

```
print(i)
```

This prints all **odd numbers** from 1 to 9.

Exercises (for Students)

Exercise 1 — Sum of Even Numbers

Write a program that uses a **while loop** to calculate the sum of all even numbers from 1 to 100.

Exercise 2 — Count Vowels in a String

Write a program that:

- takes a word from the user
- uses a **for loop** to count how many vowels it contains

Exercise 3 — Multiplication Table

Write a program that:

- asks the user for a number
- prints its multiplication table from 1 to 10 using a **for loop**

Exercise 4 — Temperature Conversion

Given a list of temperatures in Celsius:

temps = [10, 20, 30, 40, 50]

Use a **for loop** to convert each value to Fahrenheit using:

$F=95C+32$

Exercise 5 — Break and Continue

Write a program that prints numbers from 1 to 20 but:

- stops the loop when the number is 15 (break)
- skips all multiples of 3 (continue)