1. What is a Loop in Programming?

A **loop** is used to **repeat** a block of code **again and again**, until we want to stop.

Real-Life Explanation:

We all repeat things in real life:

- While watching reels, you scroll again and again
- While your alarm rings, you keep snoozing
- While the phone is charging, the percentage increases
- While playing a game, you loop levels until you win

In programming, **loops help us avoid writing the same code multiple times.**

🔹 2. Why use a loop?

Without loop:

```
print("Hello")
print("Hello")
print("Hello")
print("Hello")
print("Hello")
```

With loop:

```
i = 0
while i < 5:
```

```
    print("Hello")
    i += 1
```

✅ Saves time
✅ Makes your code shorter
✅ Helps when repeating something 100s or 1000s of times

---

3. `while` Loop – Repeat Until Condition Becomes False

Syntax:

```
while condition:
    # do something
```

- It **checks the condition first**
- If condition is True → code runs
- If condition becomes False → loop stops

Example 1: Countdown

```
count = 5

while count > 0:
    print("Countdown:", count)
    count -= 1
```

Explanation:
The loop prints countdown numbers from 5 to 1.

Example 2: College Reminder Bot

```
days_left = 3

while days_left > 0:
    print(f"{days_left} days left for the exam!")
    days_left -= 1
```

Motivation reminder until exam starts 😅

Example 3: Ask Until Correct Password

```
password = ""

while password != "1234":
    password = input("Enter password: ")

print("Access Granted ✅")
```

Keeps asking until correct password is entered

Infinite Loop Warning

```
while True:
    print("I will run forever!")
```

Be careful. This will never stop unless you force it to stop.

Practice Questions

1. Print "I Love Python" 5 times using a `while` loop
2. Print numbers from 10 to 1 (reverse countdown)
3. Take input from user. Keep asking until they type `"exit"`
4. Simulate a countdown timer (start from 3, stop at 0)
5. Ask the user their age until they enter an age ≥ 18

What is a `for` loop?

A `for` loop is used when you want to **do something a fixed number of times**, or **go through a list or range of values**.

Think of it like:

> "Do this 5 times"
> "Check each item one by one"

Real-Life Examples:

- For every student in class → call their name
- For every item in cart → show total price
- For each friend in WhatsApp group → send birthday wish

---

1. What is `range()`?

Simple Definition:

`range()` is a **built-in Python function** that creates a list of numbers in a **sequence** — one after another.

But it doesn't show a real list — it just gives you numbers one by one when you loop through it.

Think of it like:

A **virtual number counter** in Python that starts from one number and counts up (or down), step by step.

---

2. Basic Use of `range()`

Example:

```
for i in range(5):
    print(i)
```

What this does:

- It gives numbers: `0, 1, 2, 3, 4`
- Starts at `0` by default
- Ends **just before** `5` (not including it)

---

3. `range(start, stop)` – Start from where you want

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

```
1
2
```

```
3
4
5
```

Starts at 1
Stops before 6

---

4. `range(start, stop, step)` – Go with a step

```
for i in range(2, 11, 2):
    print(i)
```

```
2
4
6
8
10
```

Starts at 2
Goes up to 10
Steps by 2 (even numbers)

---

5. Use Negative Step – Count Backwards

```
for i in range(5, 0, -1):
    print(i)
```

```
5
4
3
2
```

`1`

Counts **backwards**
Starts at `5`, ends just **before** `0`
Step = `-1` (goes down)

Syntax:

```
for variable in range(start, stop, step):
    # code to repeat
```

- `start`: Where to begin (default is 0)
- `stop`: Where to stop (not included)
- `step`: How much to increase each time (default is 1)

Example 1: Print "Hello" 5 times

```
for i in range(5):
    print("Hello", i)
```

```
Hello 0
Hello 1
```

```
Hello 2
Hello 3
Hello 4
```

Example 2: Print numbers from 1 to 10

```
for num in range(1, 11):
    print(num)
```

Example 3: Print even numbers from 2 to 20

```
for i in range(2, 21, 2):
    print(i)
```

Example 4: Print reverse countdown from 5

```
for i in range(5, 0, -1):
    print("Countdown:", i)
```

Example 5: Total marks of 5 students

```
total = 0

for i in range(5):
    marks = int(input(f"Enter marks of student {i+1}: "))
    total += marks

print("Total Marks:", total)
```

Practice Questions for College Students

1. Print your name 10 times
2. Print all odd numbers between 1 to 20
3. Ask user how many stars to print and print that many `*`
4. Calculate sum of numbers from 1 to 100
5. Take number of subjects and calculate average marks
6. Print "Welcome to Python" message with numbers 1 to 5
7. Create a loop that prints multiplication table of 7

---

1. Why Use `break` and `continue`?

When we use loops, sometimes we want to:

- Stop the loop early → use `break`
- Skip one round in the loop → use `continue`

---

2. `break` – Exit the Loop Early

Meaning:

🔴 "Stop everything and get out of the loop."

Example 1: Exit when number found

```
for i in range(1, 10):
    print(i)
    if i == 5:
        break
```

```
1
2
3
4
5
```

As soon as `i == 5`, loop stops.

Example 2: Password guessing (stop after correct)

```
while True:
    pwd = input("Enter password: ")
    if pwd == "1234":
        print("Access Granted ✅")
        break
```

🔹 3. `continue` – Skip One Round

Meaning:

"Skip this one and go to the next loop turn."

Example 1: Skip even numbers

```
for i in range(1, 6):
    if i % 2 == 0:
        continue
    print(i)
```

```
1
3
5
```

It skips 2 and 4, but doesn't stop the loop.

Example 2: Skip blank input

```
for i in range(3):
    name = input("Enter your name: ")
    if name == "":
        print("Name can't be blank! Skipping...")
        continue
    print("Hello", name)
```

Practice Questions

1. Print numbers from 1 to 10, but stop when number reaches 6
2. Print all numbers from 1 to 20 but **skip multiples of 3**
3. Loop 10 times but print only **odd** numbers