📘 Chapter 7: `for` Loops in Python

> *“When something repeats in life, it’s called a routine. In Python, it's a for loop.”*

🎯 What You'll Learn

- What `for` loops are and why we use them
- How to use the `range()` function
- How to control loop behavior with `break`, `continue`, and `else`
- How to create nested `for` loops
- Real-world patterns: stars, grids, and tables
- Common errors and debugging tips

🧠 Introduction: Why Loops Exist

Suppose you want to print “Hello” five times.

You could write:

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

But that’s ✂️ inefficient and 🔁 boring.

Instead:

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

✅ Outcome:

- Less code
- More control
- Dynamic and reusable

🔁 The for Loop and range()

📌 Syntax:

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

- **start** → where to begin (default = 0)
- **stop** → where to end (non-inclusive)
- **step** → how much to increment (default = 1)

🔍 Example:

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

🔽 Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

🧭 Visual: How for + range() Works

```
[range(1, 6)] → [1, 2, 3, 4, 5]
      ↓
for i in range:
      |
      → run block using i
      → increase i by 1
      → repeat until i = 6 (stop)
```

🌟 Examples

```
for i in range(3):
    print("Hello!")
```

🔽 Output:

```
Hello!
Hello!
Hello!
```

```python
for i in range(1, 10, 2):  # step = 2
    print(i)
```

🔽 **Output:**

```
1
3
5
7
9
```

🛑 break – Stop the Loop Early

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

🔽 **Output:**

```
0
1
2
3
4
```

> 📌 `break` cuts the loop immediately when condition is met.

🔄 continue – Skip One Iteration

```python
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

🔽 **Output:**

```
1
2
4
5
```

> 📌 `continue` skips the rest of the loop for this round and moves to the next.

✅ else with for Loops

```python
for i in range(3):
    print(i)
else:
    print("Loop completed.")
```

🔽 Output:

```
0
1
2
Loop completed.
```

> 📌 `else` runs only if the loop wasn't broken by `break`

🧷 Visual Summary

```
for i in range:
    ├─ if condition → break → exit
    ├─ if condition → continue → skip this round
    └─ otherwise → run block
else:
    └─ runs only if no break occurred
```

🧱 Nested for Loops

> Loop inside a loop = perfect for grids, tables, and patterns

```python
for i in range(1, 4):        # Outer loop
    for j in range(1, 4):    # Inner loop
        print(i, "*", j, "=", i * j)
```

🔽 Output:

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
2 * 1 = 2
...
```

🌟 Pattern Example – Triangle of Stars

```python
for i in range(1, 5):
    for j in range(i):
        print("*", end=" ")
    print()
```

🔽 Output:

```
*
* *
* * *
* * * *
```

🧠 Mini Quiz (10 Questions)

1. What does `range(5)` return?
2. What is printed by:

    ```python
    for i in range(3):
        print(i)
    ```

3. What does `break` do inside a loop?
4. What does `continue` do?
5. How many times does this run?

    ```python
    for i in range(1, 10, 3):
        print(i)
    ```

6. What happens if you use `range(10, 0, -1)`?
7. When does the `else` part of a `for` loop execute?
8. Write a loop that prints only even numbers from 1 to 20.
9. What will this print?

    ```python
    for i in range(2):
        for j in range(2):
            print(i, j)
    ```

10. Fix the bug:

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

✅ Basic Practice (15 Problems)

- Print numbers 1 to 10
- Print only odd numbers between 1 and 20
- Print numbers in reverse: 10 to 1
- Print squares of numbers 1 to 5
- Print each character in the word "Python"

- Print even numbers using step in range()
- Print multiplication table of 7
- Use `break` to stop when number reaches 5
- Use `continue` to skip number 3
- Loop from 1 to 10, but skip 6 and 8
- Print sum of numbers from 1 to 50
- Print all numbers divisible by 3 between 1 and 30
- Ask user for number and print its table
- Print triangle pattern of `#` symbols
- Print grid of `*` of size 3x3

🚀 Intermediate Practice (5 Problems)

- Print the Fibonacci sequence (first 10 terms using loop logic)
- Create a triangle of numbers:

```
1
1 2
1 2 3
```

- Print FizzBuzz from 1 to 30
- Loop through a list of names and print greetings
- Print all prime numbers from 1 to 50 (nested loop logic)

🏁 You've Mastered the for Loop!

✅ You Now Know:

- How to use `range()` to loop through numbers
- How to stop or skip loop execution using `break` and `continue`
- How to build patterns and sequences
- How to write clean, readable, and reusable loop logic