📘 Chapter 8: `while` Loops in Python

> *"When you don't know how many times you'll repeat — `while` is your weapon."*

🎯 What You'll Learn

- How `while` loops work
- How they differ from `for` loops
- Looping until a condition becomes false
- Avoiding infinite loops
- Using `break`, `continue`, `else` with `while`
- Real examples: input validation, games, menus

🧠 Why `while` Loops?

Unlike `for`, which repeats a fixed number of times...

Use `while` when:

- You don't know how many repeats
- You want to keep asking until a condition is met

🔁 Basic `while` Syntax

```python
while condition:
    # run this block
```

> The block runs as long as the condition is `True`.

🧪 Example:

```python
count = 1

while count <= 5:
    print("Count is:", count)
    count += 1
```

🔽 Output:

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

🔍 Visual Flow

```
[condition] → True → run block → check again
            → False → exit loop
```

🚩 Infinite Loops: BEWARE!

```python
while True:
    print("This goes forever...")
```

> ⚠️ This will never stop unless:

- You use `break`
- Or press Ctrl + C (manual stop)

❌ Common Mistake:

```python
x = 5
while x > 0:
    print(x)
# forgot to change x → infinite loop!
```

✅ Fix:

```python
    x -= 1
```

🌟 Using `break` in while

```python
while True:
    password = input("Enter password: ")
    if password == "secret":
        print("Access granted.")
        break
```

> 📌 `break` exits the loop immediately — even if condition is still `True`.

🌟 Using `continue` in while

```python
i = 0
while i < 5:
    i += 1
    if i == 3:
        continue
    print(i)
```

🔽 Output:

```
1
2
4
5
```

> 📌 `continue` skips the rest of that loop cycle.

✅ Using `else` with while

```python
x = 0
while x < 3:
    print(x)
    x += 1
else:
    print("Loop completed!")
```

> ✅ `else` runs only if the loop finishes without `break`

🧠 Real Use Case: Number Guessing

```python
secret = 7
guess = int(input("Guess the number: "))

while guess != secret:
    guess = int(input("Wrong! Try again: "))

print("You guessed it!")
```

🔄 Visual: while Loop Logic

```
START
  ↓
[ condition? ] → No → exit
      |
     Yes
      ↓
 [ run block ]
      ↓
 [ update something ]
      ↓
[ check again... ]
```

🧠 Mini Quiz (10 Questions)

1. What's the key difference between `while` and `for`?
2. What can happen if your condition never changes?
3. When does `else` run in a `while` loop?
4. Write a loop that keeps printing until user types "stop"
5. What does `break` do?
6. How would you skip even numbers in a loop from 1 to 10?
7. Predict output:

```python
x = 3
while x:
    print(x)
    x -= 1
```

8. Fix the bug:

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

9. What does this print?

```python
i = 0
while i < 3:
    i += 1
    if i == 2:
        break
else:
    print("Done")
```

10. Why do we need `x += 1` or similar in a `while` loop?

✅ Basic Practice (15 Problems)

- Print numbers 1 to 10 using `while`
- Print numbers in reverse from 10 to 1
- Ask the user to guess a secret number (until correct)
- Print even numbers from 1 to 20
- Keep taking input until the user types "exit"
- Ask the user for a number. Keep looping until it's positive
- Print the sum of numbers from 1 to 100 using `while`
- Ask for marks until user enters -1
- Count and print digits of a number
- Validate password (loop until correct)
- Print multiples of 5 from 1 to 50
- Use `break` to stop loop at number 7
- Use `continue` to skip number 3
- Loop to print digits of a number in reverse
- Ask for age and check for valid (1–120 range)

🚀 Intermediate Practice (5 Challenges)

- Login system: 3 tries max, then lockout
- Build a mini calculator (runs until user types "stop")
- Loop to display a countdown with delay (optional `time.sleep`)
- Number guess game with hints: "too high" / "too low"
- Track highest number entered before quitting

🛠️ Debug Challenges

Infinite loop:

```python
x = 10
while x > 0:
    print(x)
# what's missing?
```

`continue` skips everything:

```python
x = 0
while x < 5:
    continue
    print(x)
```

`else` not executing — why?

```python
i = 0
while i < 5:
    if i == 3:
        break
    i += 1
else:
    print("Done")
```

💡 Mini Project: Password Gatekeeper 🔐

Build a security gate system.

🎯 Features:

- Ask user for a password
- Allow up to 3 attempts
- If password is correct → print "Access granted"
- If failed 3 times → print "Account locked"

✅ Sample Code Logic:

```python
correct_password = "python123"
attempts = 0

while attempts < 3:
    entry = input("Enter password: ")
    if entry == correct_password:
        print("Access granted ✅")
        break
    else:
        print("Wrong password ❌")
        attempts += 1
else:
    print("Account locked 🚫")
```

🏁 You’ve Mastered `while` Loops!

✅ You now know how to:

- Loop based on conditions
- Handle dynamic repetition
- Prevent infinite loops
- Use `break`, `continue`, and `else` like a pro

> 🔍 In the real world, `while` is your go-to for:

- Input loops
- Games
- Security checks
- Validation
- Unknown repetitions