Unit 4 - LOOPS

"Doing things again and again... without getting tired!"

What are Loops?

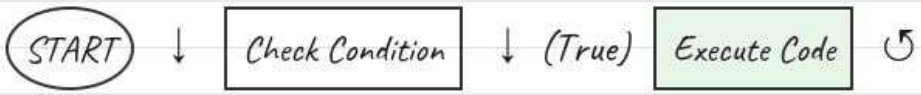
In programming, loops are used to repeat a block of code multiple times until a specific condition is met.

Real-world Analogy:

- ⏰ **Alarm Clock:** Rings every 5 mins until you wake up!
- 💓 **Heartbeat:** Beats continuously as long as you are alive.
- 🏃 **Laps in a Race:** Run around the track 10 times.

Types of Loops in Python:

1. **while loop** (Condition-based)
2. **for loop** (Sequence-based)



1. The **while** Loop

It repeats as long as the condition is True.

```
# Syntax
while condition:
    # code to repeat
    increment/decrement
```

Example: Print numbers 1 to 5

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

```
Output: 1 2 3 4 5
```

Dry Run Table:

Iteration	Condition (i <= 5)	Output	New i
1st	1 <= 5 (T)	1	2
2nd	2 <= 5 (T)	2	3
3rd	3 <= 5 (T)	3	4
4th	4 <= 5 (T)	4	5
5th	5 <= 5 (T)	5	6
6th	6 <= 5 (F)	--	STOP

⚠ **Common Mistake:**

Forgetting to update the variable (i = i + 1) leads to an INFINITE LOOP! Your computer will hang! 😱

2. The `for` Loop

Used to iterate over a sequence (list, tuple, string) or a range.

```
for item in sequence:
    # code to repeat
```

The `range()` Function

It generates a sequence of numbers.

range(start, stop, step)
- start: inclusive (default 0)
- stop: exclusive (mandatory)
- step: increment value (default 1)

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

```
Output: 1 2 3 4 5
```

> 💡 ***Memory Trick:***
> *Range is like a bus stop. It starts at 'start', but stops just before the 'stop' sign!*

Iterating over a String:

```
for char in "PYTHON":
    print(char)
```

```
Output: P Y T H O N
```

Loop Control Statements

Sometimes we need to change the normal flow of a loop.

A. **break** (The Emergency Exit)

Terminates the loop immediately.

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

```
Output: 1 2 3 4
```

(Loop stops as soon as i becomes 5)

B. **continue** (The Skip Button)

Skips the current iteration and moves to the next one.

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

```
Output: 1 2 4 5
```

(3 is skipped!)

Break: 🛑 STOP

Continue: ⏭ SKIP

C. **pass** (The Placeholder)

It does nothing! Used when a statement is required syntactically but you don't want to execute any code.

```
for i in range(5):
    pass # I will write logic later!
```

D. **else** with Loops

The `else` *block executes ONLY if the loop finishes* naturally *(without a break).*

```
for i in range(3):
    print(i)
else:
    print("Loop finished successfully!")
```

```
Output: 0 1 2 Loop finished successfully!
```

Interview Question:

Q: What happens to the `else` *block if the loop is terminated by* `break`*?*

A: The `else` *block is* SKIPPED.

3. Nested Loops

A loop inside another loop. The "Inner Loop" completes all its iterations for every single iteration of the "Outer Loop".

Analogy: A Clock
- Outer Loop: Hours (1 to 12)
- Inner Loop: Minutes (0 to 59)

For every 1 hour, the minute hand goes around 60 times!

Example: Multiplication Table (1 to 2)

```
for i in range(1, 3):
    for j in range(1, 11):
        print(i * j, end=" ")
    print() # New line
```

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
```

🔥 **Pro Tip:**
Be careful! Nested loops can make your code slow if the data is huge.

Pattern Programs (Logic Building)

Patterns help in mastering nested loops. Think in terms of Rows (i) and Columns (j).

1. Square Pattern

```
for i in range(4):
    for j in range(4):
        print("*", end=" ")
    print()
```

2. Right Triangle

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

Logic: In row 1, print 1 star. In row 2, print 2 stars...

Important Number Programs

1. Sum of Digits

```
num = 123, sum = 0
while num > 0:
    digit = num % 10
    sum += digit
    num //= 10
print(sum)
```

```
Output: 6
```

2. Reverse a Number

```
n = 456, rev = 0
while n > 0:
    rem = n % 10
    rev = rev * 10 + rem
    n //= 10
```

```
Output: 654
```

Quick Revision:

- `%  10` *gives the LAST digit.*
- `//  10` *REMOVES the last digit.*

Real-world Example:

Checking if a Credit Card number is valid using the Luhn Algorithm (uses loops to process digits).

Practice & Interview Prep

📝 **Mini Exercises:**

1. Write a loop to print the first 10 even numbers.
2. Find the factorial of a number using a `while` loop.
3. Print the Fibonacci sequence up to 10 terms.
4. Check if a number is Prime or not.
5. Create a diamond pattern using nested loops.

Top Interview Questions:

1. Difference between `for` and `while` loop?
2. What is an infinite loop? How to stop it?
3. How does the `else` clause work with loops?

CONGRATULATIONS!

You have mastered Unit 4: Loops!

Next Unit: Functions 🚀