**📘 Recursion in DSA (Python)**

**🔹 What is Recursion?**

**Recursion** is a method where a function **calls itself** to solve a smaller subproblem of the original problem.

It continues calling itself until a **base condition** is met.

**🔹 Real-Life Analogy**

Imagine placing mirrors facing each other — reflections go deeper and deeper until they can't anymore.  
In recursion, the function **keeps calling itself** until it hits a **base case**, then **unwinds** back.

**🔹 Structure of Recursive Function**

def recursive\_function(params):

if base\_condition:

return result # Stop condition (important!)

else:

return recursive\_function(smaller\_problem)

**🔹 Example 1: Factorial using Recursion**

def factorial(n):

if n == 0 or n == 1:

return 1 # Base case

return n \* factorial(n - 1) # Recursive call

**🔹 Dry Run: factorial(4)**

factorial(4)

→ 4 \* factorial(3)

→ 4 \* 3 \* factorial(2)

→ 4 \* 3 \* 2 \* factorial(1)

→ 4 \* 3 \* 2 \* 1 = 24

**🔹 Example 2: Fibonacci using Recursion**

def fibonacci(n):

if n <= 1:

return n

return fibonacci(n - 1) + fibonacci(n - 2)

**🔹 Output:**

for i in range(6):

print(fibonacci(i), end=" ")

# Output: 0 1 1 2 3 5

**🔹 Important Terms**

| **Term** | **Meaning** |
| --- | --- |
| Base Case | Condition where recursion stops |
| Recursive Case | Function calls itself with smaller input |
| Stack | Recursion uses **call stack** to store function calls |

**🔹 Pros and Cons**

| **✅ Pros** | **❌ Cons** |
| --- | --- |
| Simpler & cleaner code | Can be slower due to repeated calls |
| Good for divide-and-conquer | Risk of **stack overflow** if not handled |
| Matches mathematical logic | Harder to debug than loops |

**🔹 When to Use Recursion in DSA?**

* Tree traversals (inorder, preorder, postorder)
* Divide & conquer algorithms (Merge Sort, Quick Sort)
* Backtracking (Sudoku, Maze, N-Queens)
* Graph traversals (DFS)
* Dynamic Programming (Memoization)

**🔹 Convert Recursion to Iteration**

Many recursive problems can also be solved using **loops + stack** to avoid stack overflow.

Let me know if you want:

* Recursive Tree Traversals
* Backtracking examples (like N-Queens)
* Tail recursion optimization
* How recursion works under the hood