Recursion – Base Case & Recursive Case

1. Introduction

Every recursive function is built using **two essential components**:

- **Base Case**
- **Recursive Case**

These two parts work together to ensure that recursion **solves the problem correctly** and **terminates safely**.

Without understanding these two concepts, recursion becomes confusing and error-prone.

---

2. What is a Base Case?

The **base case** is the condition where the recursive function **stops calling itself**.

It represents the **simplest form of the problem** that can be solved directly without further recursion.

Key Points:

- Prevents infinite recursion
- Returns a direct result
- Must always be reachable

---

3. Why is the Base Case Important?

The base case is important because:

- It stops the recursive calls
- Prevents stack overflow
- Defines when the problem is fully solved

- Allows results to return back through the call stack

Without a base case, recursion will run **forever**.

---

4. Example of a Base Case (Conceptual)

To find factorial of a number:

- Factorial of 0 or 1 is 1

This condition becomes the **base case**.

When the function reaches this input, it **stops recursion**.

---

5. What is a Recursive Case?

The **recursive case** is the part of the function where it **calls itself** with a **smaller or simpler input**.

This step gradually moves the problem **closer to the base case**.

---

6. Why is the Recursive Case Important?

The recursive case is important because:

- It breaks the problem into smaller sub-problems
- Defines how recursion progresses
- Ensures the problem is reduced step by step
- Connects the large problem to the base case

---

7. Relationship Between Base Case and Recursive Case

Base Case	Recursive Case
Stops recursion	Continues recursion
Solves smallest problem	Breaks problem further
Returns result	Calls itself

Base Case	Recursive Case
Prevents infinite loop	Moves toward base case

Both must work together for recursion to succeed.

---

8. Logic of Recursion Using Base & Recursive Case (Plain English)

1. Check if the input satisfies the base condition
2. If yes, return the result directly
3. If not, perform a recursive call with reduced input
4. Continue reducing input
5. Reach base case
6. Return results step by step

---

9. Example Walkthrough (Conceptual)

Problem: Print numbers from 1 to 5

- Base Case:

  Stop when number becomes greater than 5

- Recursive Case:

  Print current number and call function with next number

This ensures:

- Numbers are printed in order
- Recursion stops correctly

---

10. Common Mistakes Related to Base & Recursive Case

- Forgetting to define base case

- Base case never reached
- Recursive call not reducing input
- Multiple base cases without clarity
- Wrong condition causing infinite recursion

---

11. How to Identify Base & Recursive Case in Problems

Base Case:

- Smallest input
- Simplest scenario
- No further division possible

Recursive Case:

- Input reduces
- Same function logic applies
- Moves toward base case

---

12. Real-World Analogy

Think of climbing down stairs:

- **Recursive Case:**

  Take one step down
- **Base Case:**

  When you reach the ground floor, stop

---

13. Summary

- Base case stops recursion
- Recursive case continues recursion

- Both are mandatory
- Recursive case must move toward base case
- Proper design avoids infinite calls
