What is Recursion?

- **Definition:** A function that calls itself.
- **Key idea:** Breaking a problem into smaller, similar subproblems.
- Uses the **call stack** to keep track of function calls.

- **Structure of a Recursive Function**
  - A base case is the condition in a recursive function that stops the recursion. It's the simplest, smallest instance of the problem that can be solved directly without making another recursive call.
  - A recursive case is the part of a recursive function where the function calls itself with a reduced version of the original problem, moving it closer to the base case. It's where the actual "problem-solving" happens by breaking down a complex problem into smaller, more manageable subproblems.

Activation record

- **Activation record**
  - An **activation record** is the *data structure* created **each time a function (or procedure) is called**.
    It stores everything needed for the program to execute that function and later resume the caller.
- **Call stack** = a *stack* of **activation records**.
  - A **call stack** is a **runtime data structure** that manages *function calls* in a program.
  - It keeps track of **which function is currently running** and **where to return after it finishes**.
  - Operates in **Last-In, First-Out (LIFO)** order.

Example: Factorial

```python
def fact(n):
    #Calculates n!
    if n == 0:
        return 1
    else:
        return n * fact(n-1)
```

- Call chain for `factorial(3)` :

```
factorial(3)
   -> 3 * factorial(2)
      -> 2 * factorial(1)
         -> 1 * factorial(0)
            -> returns 1
```

- **Common Pitfalls**
  - **No base case** → infinite recursion → program crash.
  - **Base case never reached** due to wrong step size.

Recursion vs Iteration

```python
# Recursive factorial
def fact(n):
    if n == 0 or n == 1:
        return 1
    return n * fact(n - 1)
```

```python
# Iterative version
def fact(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

Advantages and Disadvantages of Recursion

- **Advantages of Recursion**

  ✅ Mirrors mathematical definitions.
  ✅ Clean and concise for problems naturally recursive.
  ✅ Easy to implement algorithms that work on hierarchical/nested data.

- **Disadvantages of Recursion**

  ⚠️ Higher memory usage (each call consumes stack space).
  ⚠️ Slightly slower due to function call overhead.
  ⚠️ Risk of **stack overflow** if recursion depth is too large (Python has a limit ~1000 calls).
  ⚠️ Sometimes harder to debug if logic is wrong.