(25/8/25: Rapid Introduction to Procedural Programming).

1. What is Procedural Programming?

Procedural Programming is a **programming paradigm** (style of programming) that is based on the concept of **procedures** (also called functions, routines, or subroutines).

- A program is divided into **small blocks of code** (procedures) that perform specific tasks.
- These procedures can be reused throughout the program.
- It focuses on **step-by-step execution** (sequence of instructions).

Python, although it supports multiple paradigms (procedural, object-oriented, functional), is commonly introduced through **procedural programming**.

2. Key Features of Procedural Programming

1. **Top-Down Approach** → Programs are written step by step in a logical order.
2. **Use of Functions** → Code is organized into reusable blocks (functions).
3. **Modularity** → The program is divided into smaller, manageable parts.
4. **Variables** → Used to store data for processing.
5. **Control Structures** → If-else, loops, etc. are used to control flow.

3. Steps in Procedural Programming

1. Define the **problem** clearly.
2. Break it into **smaller tasks** (procedures).
3. Write functions for each task.
4. Execute them in a **step-by-step sequence**.

4. Advantages

- Easy to learn and use.
- Code is **readable** and structured.
- Functions promote **code reuse**.
- Debugging is simpler (smaller chunks of code).

5. Limitations

- Not suitable for **very large projects** (hard to manage functions).
- Code can become **less flexible** compared to OOP.

Example Program: Sum of Two Numbers

```
# Example: Procedural approach in Python

# Step 1: Define a procedure (function) for input
def get_numbers():
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    return a, b

# Step 2: Define a procedure for calculation
def calculate_sum(x, y):
    return x + y

# Step 3: Define a procedure for output
def display_result(result):
    print("The sum is:", result)

# Main program (step-by-step execution)
num1, num2 = get_numbers()   # Input
result = calculate_sum(num1, num2)   # Processing
display_result(result)   # Output
```

6.**Sample Output**

```
Enter first number: 10
Enter second number: 20
The sum is: 30
```