Introduction to Procedural Programming in Python

1. What is Procedural Programming?

- A **programming paradigm** focused on writing **step-by-step instructions** that the program executes in sequence.
- Code is structured into **procedures**, also known as:
  - **Subroutines**: Blocks of code for specific tasks.
  - **Functions**: Reusable and independent logic sections.

2. Key Features of Procedural Programming

- **Modular Code**: Divides the program into smaller sections for specific tasks.
- **Reusability**: Functions can be reused across the program.
- **DRY Principle**: *Don't Repeat Yourself*—reduces code duplication.
- **Readability**: Breaking code into logical procedures makes it easier to understand.

3. Example 1: Adding Two Numbers

Inefficient Approach

```
# Adding specific numbers
result1 = 5 + 10
print(result1)

result2 = 8 + 4
print(result2)
```

- **Problem**: Duplication. Separate code is needed for every new pair of numbers.

Improved Approach: Using a Function

```
# Function for adding numbers
def add_numbers(a, b):
    return a + b

# Reusing the function
print(add_numbers(5, 10))  # Output: 15
print(add_numbers(8, 4))   # Output: 12
```

- **Solution**: The `add_numbers` function is reusable, avoiding repetitive code.

4. Example 2: Calculating a Bill with Tax

Breaking Down the Problem

1. **Calculate Total Bill**: Add up all items on the bill.
2. **Calculate Tax**: Determine the tax for the total bill.
3. **Display Results**: Show the total bill, tax, and overall amount.

Step-by-Step Code

1. Define `bill_total` Function

```
def bill_total(bill):
    """Calculates the total of a given bill."""
    total = 0
    for item in bill:
        total += item
    return total
```

2. Define `calculate_tax` Function

```
def calculate_tax(total, tax_percentage):
    """Calculates tax based on the total and tax percentage."""
    tax = round(total * (tax_percentage / 100), 2)
    return tax
```

3. Create a Sample Bill

```
food_bill = [10.50, 20.30, 5.20]  # Example: List of item prices
```

4. Combine Procedures

```
# Main code
total = bill_total(food_bill)
tax = calculate_tax(total, 5)  # Assuming 5% tax

print(f"Total Bill: ${total:.2f}")
print(f"Tax: ${tax:.2f}")
print(f"Overall Total: ${total + tax:.2f}")
```

5. Advantages of Procedural Programming

1. **Easy to Learn**: Simple, straightforward, and beginner-friendly.
2. **Reusable Procedures**: Functions can be reused in multiple places.
3. **Logical Flow**: Steps are clear and executed in order.
4. **Easier Debugging**: Smaller, isolated tasks are easier to test and debug.

6. Disadvantages of Procedural Programming

1. **Harder Maintenance**:
   - Extending functionality can lead to complex interdependencies.
2. **Global Data Exposure**:
   - Shared data can be inadvertently modified by different parts of the program.
3. **Limited Real-World Modeling**:
   - Doesn't map well to real-world objects compared to Object-Oriented Programming (OOP).

7. Summary

Procedural programming provides an essential foundation for beginners, helping them:

- Break problems into logical steps.
- Write reusable, modular code.
- Understand the DRY principle.

While it has limitations, procedural programming remains a powerful tool for simple, structured, and linear applications. As developers progress, they can combine it with other paradigms like **Object-Oriented Programming (OOP)** and **Functional Programming** for greater flexibility and scalability.