Project Report: Python Calculator Application

Project Name: Simple Calculator **Language:** Python 3.x **Author:** [Your Name] **Date:** [Current Date]

---

1. Introduction

The **Simple Calculator** is a command-line interface (CLI) application developed using the Python programming language. Its primary purpose is to perform fundamental arithmetic operations—Addition, Subtraction, Multiplication, and Division—quickly and accurately.

This project demonstrates core programming concepts such as modular function design, conditional logic, user input handling, and basic error management. It serves as a foundational tool for understanding how software processes numerical data.

2. Objectives

The main objectives of this project are:

- To create a user-friendly interface for mathematical calculations.
- To implement modular programming by separating logic into distinct functions.
- To ensure the program is robust by handling edge cases, such as **division by zero**.
- To provide a clean, readable codebase suitable for educational or portfolio purposes.

3. System Requirements

- **Operating System:** Windows, macOS, or Linux.
- **Runtime Environment:** Python 3.6 or higher.
- **Development Tools:** Visual Studio Code (VS Code), Jupyter Notebook.

4. Methodology & Implementation

The project is built using a **procedural programming paradigm**. The code is structured into specific functions, each handling a single responsibility.

4.1 Functional Modules

The application is divided into the following user-defined functions:

1. **add(a, b)**: Accepts two floating-point numbers and returns their sum.
2. **subtract(a, b)**: Accepts two floating-point numbers and returns the difference.
3. **multiply(a, b)**: Accepts two floating-point numbers and returns the product.
4. **divide(a, b)**: Accepts two numbers. It includes a logical check:
    - If the divisor (b) is 0, it returns an error message ("Error: Cannot divide by zero").

Mahima Gupta

- Otherwise, it returns the quotient.

4.2 Main Execution Logic

The program follows a linear execution flow:

1. **Display Menu**: The user is presented with available operations (+, -, *, /).
2. **Input Acquisition**: The program prompts the user to enter two numbers (float type) and an operator string.
3. **Conditional Routing**: An if-elif-else control structure determines which function to call based on the operator entered.
4. **Result Display**: The final calculated value (or error message) is printed to the console.

5. Testing and Validation

The application was tested with various inputs to ensure accuracy and stability.

Test Case	**Input 1**	**Input 2**	**Operator**	**Expected Output**	**Actual Output**	**Status**
Addition	10.5	5.5	+	16.0	16.0	Pass
Subtraction	20	8	-	12.0	12.0	Pass
Multiplication	4	2.5	*	10.0	10.0	Pass
Division	100	20	/	5.0	5.0	Pass
Div by Zero	50	0	/	Error Message	Error: Cannot divide by zero	Pass
Invalid Op	10	10	%	Invalid operator!	Invalid operator!	Pass

Mahima Gupta

6. Conclusion and Future Scope

6.1 Conclusion

The Simple Calculator project successfully meets all its objectives. It provides a reliable way to perform calculations and demonstrates clean coding practices. The separation of concerns (using functions) makes the code easy to read, debug, and maintain.

6.2 Future Scope

To enhance this project in the future, the following features could be added:

- **Scientific Operations:** Adding support for square roots, exponents, and trigonometric functions (sin, cos, tan).

- **Graphical User Interface (GUI):** Developing a visual calculator window with clickable buttons to replace the text-based interface.
- **History Feature:** Saving the last 5 calculations to a log file so users can review their work.

Mahima Gupta