TITLE: PYTHON CALCULATOR WITH UNIT TESTING

Team Members:

1.	**Name:**	Bhanu Priya R
	CAN_ID:	CAN_35273311
2.	**Name:**	Muppalla Mohitha
	CAN_ID:	CAN_35272392
3.	**Name:**	M Chandra Shekara
	CAN_ID:	CAN_35269198
4.	**Name:**	Nikhila S
	CAN_ID:	CAN_35266392
5.	**Name:**	Narendra Babu G
	CAN_ID:	CAN_35277509

PHASE 1: RESEARCH

1.1 INTRODUCTION

In today’s software-driven world, calculators serve as fundamental tools across almost every field, from basic arithmetic to complex engineering computations. This project, titled "Python Calculator with Unit Testing", focuses on building a simple yet structured arithmetic calculator using the Python programming language. The core aim is to perform basic mathematical operations—addition, subtraction, multiplication, and division—while demonstrating key concepts of Object-Oriented Programming (OOP) and software testing practices.

Unlike traditional calculators, this one emphasizes code quality, maintainability, and reliability. The application is designed using a class-based structure (Calculator class), where each operation is encapsulated within its own method. To ensure the correctness of each function, a comprehensive unit testing module is implemented using Python's built-in unittest framework. This makes the project a great example of integrating both development and testing environments within a small-scale application.

Furthermore, the project incorporates exception handling to manage common runtime errors, such as division by zero or invalid data types. This ensures that the application behaves predictably and robustly, even in the presence of user or developer errors.

The Python Calculator project not only highlights the fundamental principles of programming but also introduces best practices such as type annotations, error handling, and test-driven development (TDD). It serves as an ideal foundation for beginners aiming to understand how real-world applications are designed, developed, and tested. Additionally, the modular nature of the code allows for future expansion, such as integrating a graphical user interface (GUI) or adding more advanced mathematical functions.

1.2 PROBLEM STATEMENT

In most beginner-level software projects, calculators are implemented in a procedural way with minimal attention to software quality, testing, or extensibility. This results in applications that are difficult to maintain or scale. Additionally, many such calculators do not gracefully handle exceptions like invalid input types or division by zero, leading to runtime crashes.

This project aims to solve these limitations by building a structured, class-based calculator that:

- Follows OOP principles.
- Includes input validation and exception handling.
- Implements unit testing to verify functionality and robustness.

1.3 SCOPE

The scope of this project is intentionally kept limited to maintain clarity and focus while demonstrating essential programming practices. This project covers:

- Implementation of a class-based calculator that performs four arithmetic operations.
- Handling of errors such as division by zero and invalid input types.
- Inclusion of Python type hints for better code understanding and type safety.
- Development of a testing module using Python’s built-in unittest framework.

- Use of standard development and testing environments.

The project is designed as a console-based application, but its modular structure allows for future extension into graphical or web-based calculators.

1.4 OBJECTIVES

The primary objectives of this project are:

1. To design and implement a calculator using Python with a focus on clean, modular, and maintainable code.
2. To handle runtime errors and invalid user inputs gracefully using exception handling.
3. To introduce Python type annotations for enhanced clarity and reliability.
4. To implement comprehensive unit tests for each calculator function using the unit test module.
5. To simulate a basic development + testing workflow, thereby preparing for real-world software development scenarios.

1.5 SUMMARY

This project is a foundational software application that demonstrates how to design and test a basic calculator in Python using object-oriented principles. Beyond simple arithmetic operations, the project highlights best practices such as modular design, type safety, exception handling, and automated testing.