

Assignment–04

Tools for Programming, Learning, and Collaboration

Mini Project: Calculator in Python using VS Code, Git & GitHub

Made By: Manas Bhasker

Submitted to: Mr. Rajesh Badrana Sir

1. Introduction

Modern software development is not limited to writing code; it requires a strong understanding of integrated development environments, version control, documentation, and collaborative workflows. In this assignment, a mini-project was created using **Visual Studio Code**, **Git**, and **GitHub** to simulate real-world development practices.

The goal of this assignment was to build a simple yet functional **Python Calculator** capable of performing basic arithmetic operations. The project was developed using VS Code, version-controlled using Git commands, and hosted in a public GitHub repository. Proper documentation and repository structure were also maintained as per professional standards.

2. Project Selection and Description

2.1 Selected Project: Python Calculator

The selected mini-project is a **console-based Calculator** written in Python. It performs:

- Addition
- Subtraction
- Multiplication
- Division
- Error handling for invalid inputs and division by zero

This project is ideal for understanding basic programming concepts, modular code design, and Git workflow.

3. Development Tools and Environment

3.1 Visual Studio Code

VS Code was used to:

- Write and test the Python script
- Organize project files
- Use integrated terminal for Git commands
- View Git changes visually

3.2 Git Version Control

Git was used for:

- Creating a repository
- Tracking changes
- Maintaining commit history
- Pushing the project to GitHub

3.3 GitHub

GitHub acted as the remote repository to:

- Host the project
- Store documentation
- Display README with screenshots
- Demonstrate proper repository structure

4. Project Implementation

4.1 Program Code

```
def add(a, b):
```

```
    return a + b
```

```
def subtract(a, b):
```

```
    return a - b
```

```
def multiply(a, b):
```

```
    return a * b
```

```
def divide(a, b):
```

```
    if b == 0:
```

```
        return "Error! Division by zero is not allowed."
    return a / b
```

```
print("=== Simple Calculator ===")
print("Select an operation:")
print("1. Addition")
print("2. Subtraction")
print("3. Multiplication")
print("4. Division")
```

```
choice = input("Enter choice (1/2/3/4): ")
```

```
if choice in ['1', '2', '3', '4']:
```

```
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
```

```
    if choice == '1':
```

```
        print("Result:", add(num1, num2))
```

```
    elif choice == '2':
```

```
        print("Result:", subtract(num1, num2))
```

```
    elif choice == '3':
```

```
        print("Result:", multiply(num1, num2))
```

```
    elif choice == '4':
```

```
        print("Result:", divide(num1, num2))
```

```
else:
```

```
    print("Invalid input! Please choose 1, 2, 3 or 4.")
```

5. Git Commands Used:

1. `git init`
2. `git add .`
3. `git commit -m "Initial commit: Added calculator program"`
`git commit -m "Added README and documentation"`
`git commit -m "Improved error handling"`
`git commit -m "Added screenshots folder"`
`git commit -m "Final project structure updated"`
4. `git push -u origin main`

6. Challenges Faced

During the development of the Python Calculator project, several challenges were encountered. These challenges played an important role in enhancing understanding of programming concepts, Git workflows, and documentation practices.

1. Understanding Git Workflow

Initially, working with Git commands such as `git init`, `git add`, `git commit`, and `git push` was slightly confusing. Managing multiple commits and ensuring that only relevant files were tracked took some time to learn. Configuring the remote repository and resolving minor push errors were also part of the learning curve.

2. Structuring the Project Repository

Organizing the repository with a proper folder structure (including `docs/`, `screenshots/`, and configuration files like `.gitignore`) required careful planning. Ensuring that unnecessary files were excluded from Git tracking was an unexpected challenge for a small project.

3. Writing Clean and Modular Code

Although the calculator is simple, designing it using functions for each operation required a shift from basic sequential programming to modular design. Ensuring that each function worked independently while maintaining readability demanded attention.

7. Learning Outcomes

The development of the Python Calculator project using VS Code, Git, and GitHub resulted in several important learning outcomes that contributed to both technical and professional growth.

1. Improved Programming Skills

Building the calculator enhanced understanding of Python fundamentals, including:

- Functions
- Conditional statements
- User input handling
- Error handling (e.g., division by zero)

This reinforced logical thinking and modular code development.

2. Hands-on Experience with VS Code

Using Visual Studio Code provided exposure to:

- Writing and editing code efficiently
- Running scripts using the integrated terminal
- Using built-in extensions for Python support
- Navigating files and managing project structure

This made the coding workflow faster and more organized.

3. Practical Knowledge of Git Version Control

The assignment provided real experience with:

- Initializing repositories
- Staging and committing changes
- Writing meaningful commit messages
- Managing multiple versions of the project

This improved understanding of how professional developers track and manage code changes.

LINK FOR REPOSITORY:

<https://github.com/StillnWater/Miniproject>