Demonstrating Coding Best Practices

Group Members:

1) Shreem Dixit

2) Tanishka Shankar Sonawane

3) Harsh Makadia

---

1. Program Overview

Program Name: Calculator

Purpose of the Program: This program allows the user to carry out the basic arithmetic functions of addition, subtraction, multiplication, and division on two numbers. The calculator project will show some programming practices that are not good as well as demonstrate some programming practices that are good by way of having two different implementations of the calculator.

How the Program Works: The user chooses an action and types in two numbers. In turn, the program executes the appropriate function to calculate and show the result of that operation. Good code uses only good coding habits whereas poor code uses bad coding habits.

Number of Functions Used: The program uses five functions:

- add()
- subtract()
- multiply()
- divide()
- main()

---

2. Selected Coding Principles

Principle 1

Name of Principle: DRY - Acronym for Don't Repeat Yourself.

Definition: DRY indicates code should never have duplicate coding. Instead, if the same logic has been applied several times, you must create reusable functions so that you can avoid redundancy in your code, making it easier for you to maintain your program.

Principle 2

Name of Principle: Sole Responsibility Principle

Definition: A method of functionality should be dedicated only to one activity. Consequently, a method's single it's business to accomplish its goal; not engage in any other task. Because we focus on one goal at a time, code becomes easier to read and maintain.

Principle 3

Name of Principle: Clean Code

Definition: Clean code is readable, understandable, and properly structured. It uses meaningful variable names, consistent formatting, and clear logic so that other developers can easily understand it.

---

3. Good Version – good_version.py

Overall Structure

Principle 1 – Implementation

How It Is Applied: Instead of repeating calculation logic multiple times, each operation is written once inside a function and reused whenever needed.

Functions Demonstrating This:

- **add()**
- **subtract()**
- **multiply()**
- **divide()**

Principle 2 – Implementation

How It Is Applied: Each function does just one thing. An example of this is add() which adds. All any function does is perform its task and the same is true for the main() function, which simply manages user input and user output.

Examples of Functions Doing One Thing:

- Add Function - only performs addition
- Subtract Function - only performs subtraction
- Multiply Function - only performs multiplication
- Divide Function - only performs division
- Main Function - just manages user interaction.

Principle 3 – Implementation

How It Is Applied: The program utilizes meaningful variable names such as 'num1', 'num2' and 'Choice'. The Indentation is uniform, and the logic within has an easy to understand flow.

Functions Demonstrating This: All functions are formatted and named consistently according to clean standards.

Why This Version Reflects Clean Code

---

4. Bad Version – bad_version.py

Overall Structure

Principle 1 – DRY Violation

How It Is Violated: The exact same lines of code asking for user input are copied and pasted four separate times inside the if/elif blocks.

Where It Occurs in Code: Inside the main conditional block. If we wanted to change the prompt from "Enter first number:" to "Type a number:", we would have to rewrite it in four different places.

Principle 2 – SRP Violation

How It Is Violated: The program mixes the menu display, user input gathering, mathematical calculations, and screen printing all into a single, massive function.

Where It Occurs in Code: The entire script operates in one continuous flow rather than separating the math logic from the user interface.

Principle 3 – Clean Code Violation

How It Is Violated: The code uses terrible, single letter variable names that provide no context. It uses global variables unnecessarily. It includes useless comments and leaves commented out "dead code" in the script. Finally, it imports modules (random) that are never used (a YAGNI violation).

Where It Occurs in Code: Throughout the entire file (using x, y, c, and r instead of descriptive names; using global r).

Why This Version Reflects Poor Design

---

5. Comparison Between Good and Bad Code

Key Structural Differences: The good code is built like building blocks (modular), where each function is an independent piece that can be swapped or tested on its own. The bad code is built like a tangled spaghetti (monolithic), where everything happens linearly in one giant block.

Impact on Readability and Maintainability: If a new developer joins the team, they can immediately understand good_version.py because the variable names tell a story. In contrast, bad_version.py forces the developer to decode the logic line-by-line to understand what r = x + y means in context, making updates dangerous and time consuming.

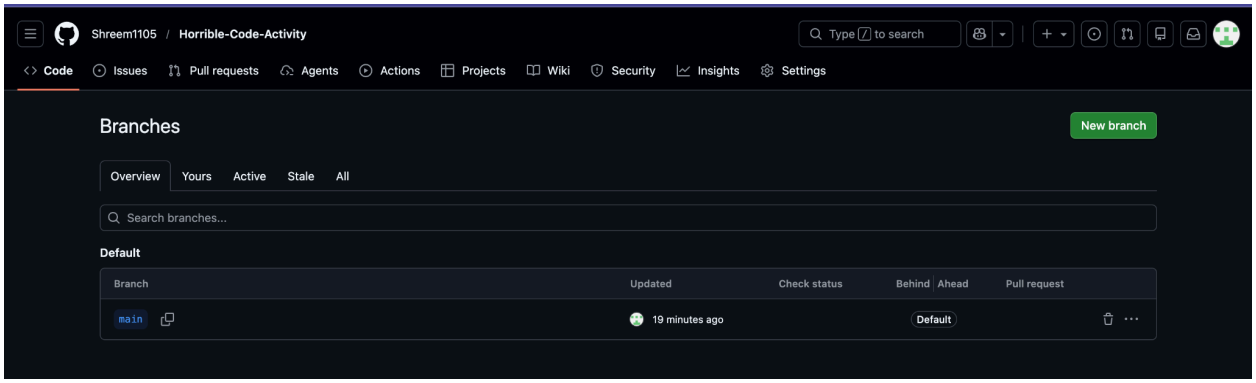
---

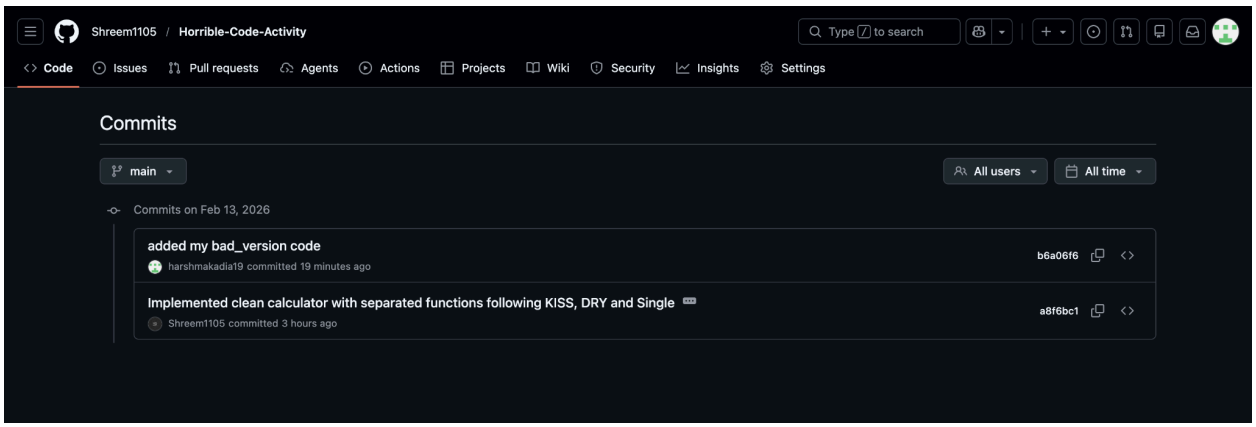
6. Conclusion

Through this exercise, we demonstrated that writing code that simply "works" is not enough. The bad version of our calculator successfully outputs correct math, but it is a nightmare to maintain. By applying the DRY Principle, the Single Responsibility Principle, and Clean Code standards to our good version, we ensured that the software is scalable, readable, and professional. Good coding practices are an investment in the future of the codebase.

---

7. Bonus


Shreem1105 / Horrible-Code-Activity
Type / to search
Code
Issues
Pull requests
Agents
Actions
Projects
Wiki
Security
Insights
Settings
Branches
New branch
Overview
Yours
Active
Stale
All
Search branches...
Default
Branch
Updated
Check status
Behind | Ahead
Pull request
main
19 minutes ago
Default


Shreem1105 / Horrible-Code-Activity
Type / to search
Code
Issues
Pull requests
Agents
Actions
Projects
Wiki
Security
Insights
Settings
Commits
main
All users
All time
Commits on Feb 13, 2026
added my bad_version code
harshmakadia19 committed 19 minutes ago
b6a06f6
Implemented clean calculator with separated functions following KISS, DRY and Single
Shreem1105 committed 3 hours ago
a8f6bc1