Ai Assissted Coding

Name : A.Rithika
Roll No : 2303A54048
Batch : 48

Lab 9: Documentation Generation – Automatic Documentation and Code Comments

Task 1: Basic Docstring Generation Scenario

You are developing a utility function that processes numerical lists and must be properly documented for future maintenance.

Requirements

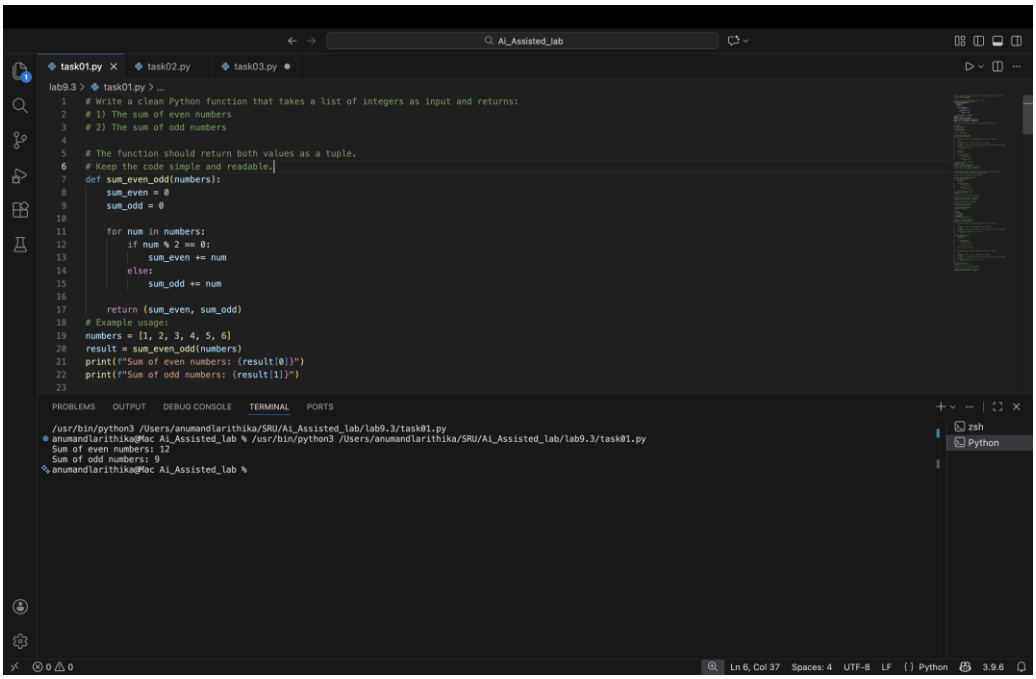
• Write a Python function to return the sum of even numbers and sum of odd numbers in a given list
• Manually add a Google Style docstring to the function
• Use an AI-assisted tool (Copilot / Cursor AI) to generate a function-level docstring
• Compare the AI-generated docstring with the manually written docstring
• Analyze clarity, correctness, and completeness

Expected Output

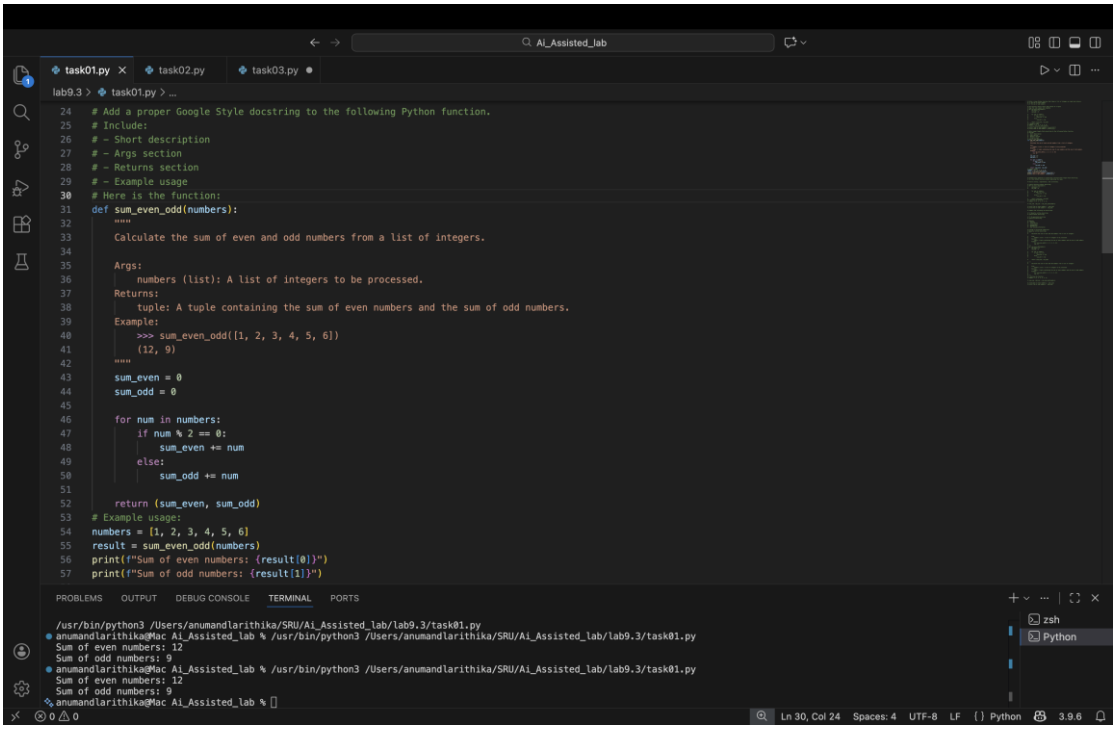
• Python function with manual Google-style docstring
• AI-generated docstring for the same function
• Comparison explaining differences between manual and AI-generated documentation
• Improved understanding of AI-generated function-level documentation

Expected Output :

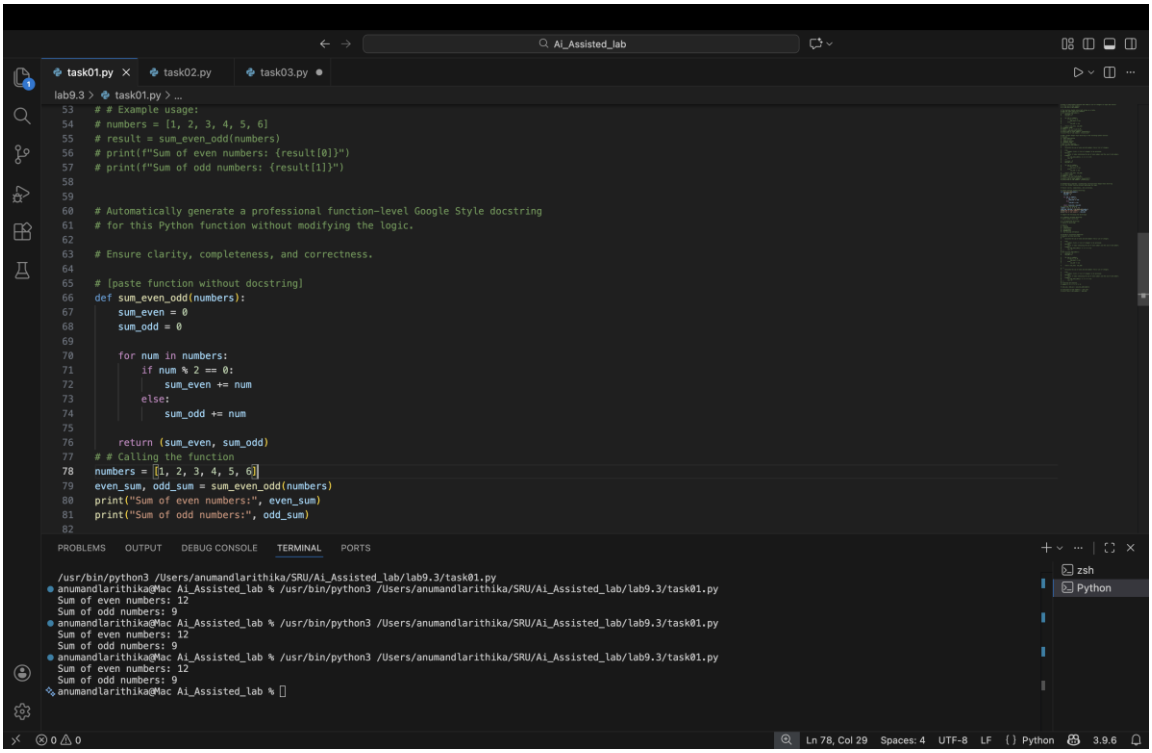
Prompt 1 – Generate Python Function :



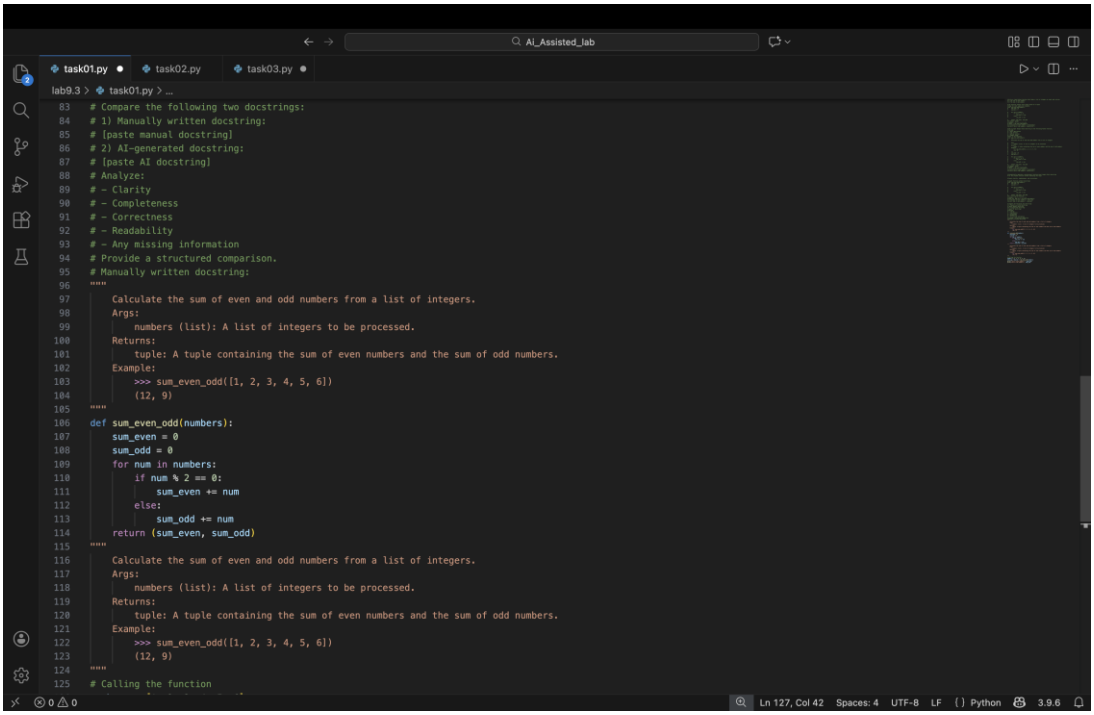
Prompt 2 – Generate Google Style Docstring :



Prompt 3 – AI Docstring Generation (for comparison)



Prompt 4 – Comparison Prompt :



Criteria	Manual Docstring	AI Docstring
Clarity	Clear but short	More detailed
Completeness	Basic explanation	Includes examples
Accuracy	Correct	Correct
Readability	Simple	Professional tone

Explanation :

In this task, I wrote a Python function to calculate the sum of even numbers and odd numbers from a list.

First, I added a **manual Google-style docstring** to explain:

- What the function does
- What input it takes
- What it returns

Then, I used an AI tool to generate a docstring automatically.

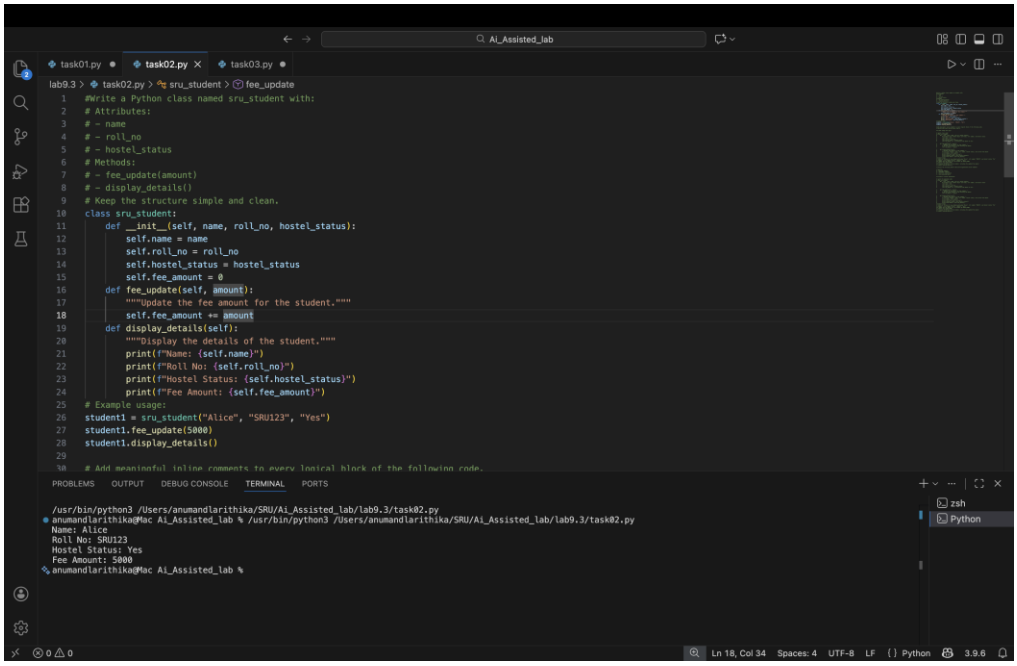
After comparing both, I observed that:

- The AI-generated docstring was more detailed.
- My manual docstring was shorter and simple.
- AI documentation looked more professional.
- Both were correct, but AI saved time.

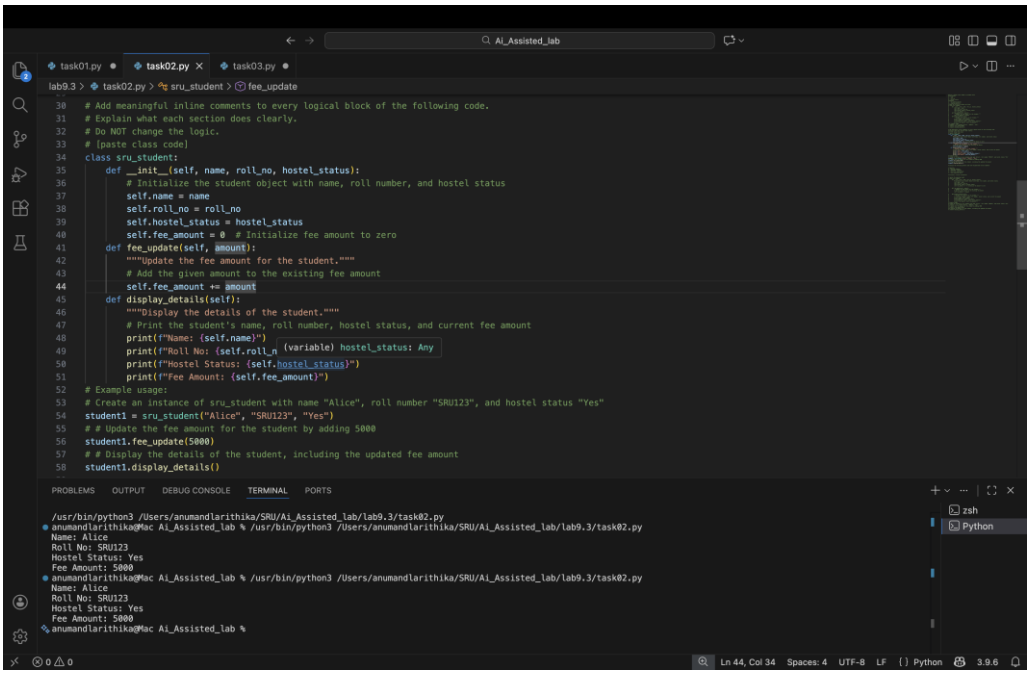
This task helped me understand how AI tools can generate function-level documentation quickly and clearly.

Task 2 – Automatic Inline Comments :

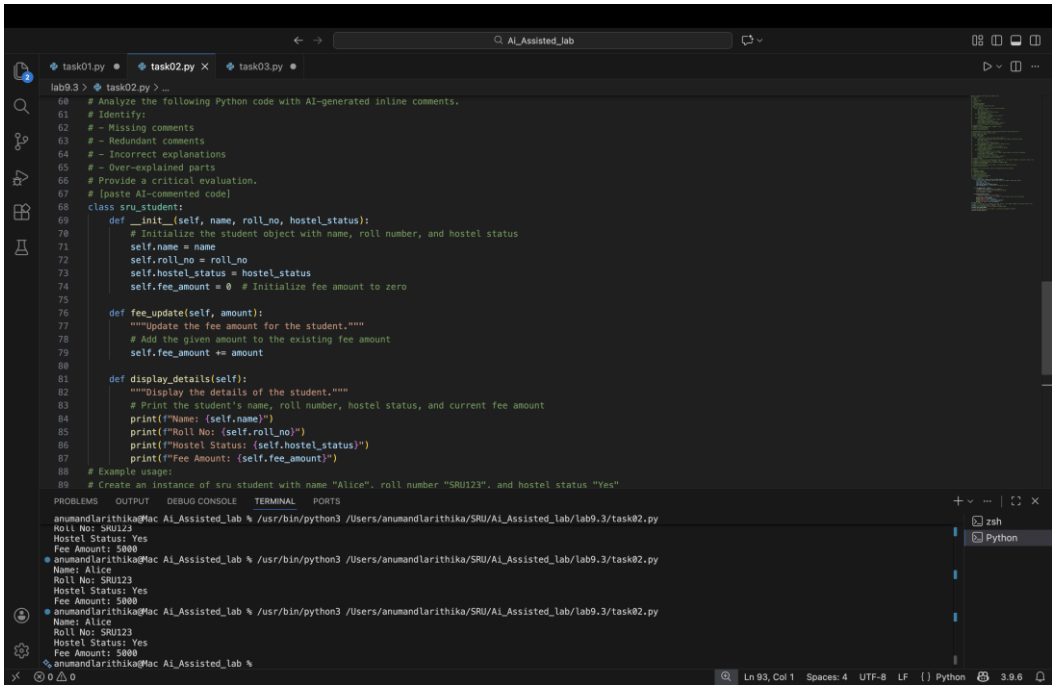
Prompt 1 – Generate Class :



Prompt 2 – Add Manual Inline Comments :



Prompt 3 – Evaluate AI Comments :



Explanation :

In this task, I created a Python class called sru_student with attributes like name, roll number, and hostel status.

First, I wrote inline comments manually to explain each part of the code.

Then, I used an AI tool to automatically generate inline comments.

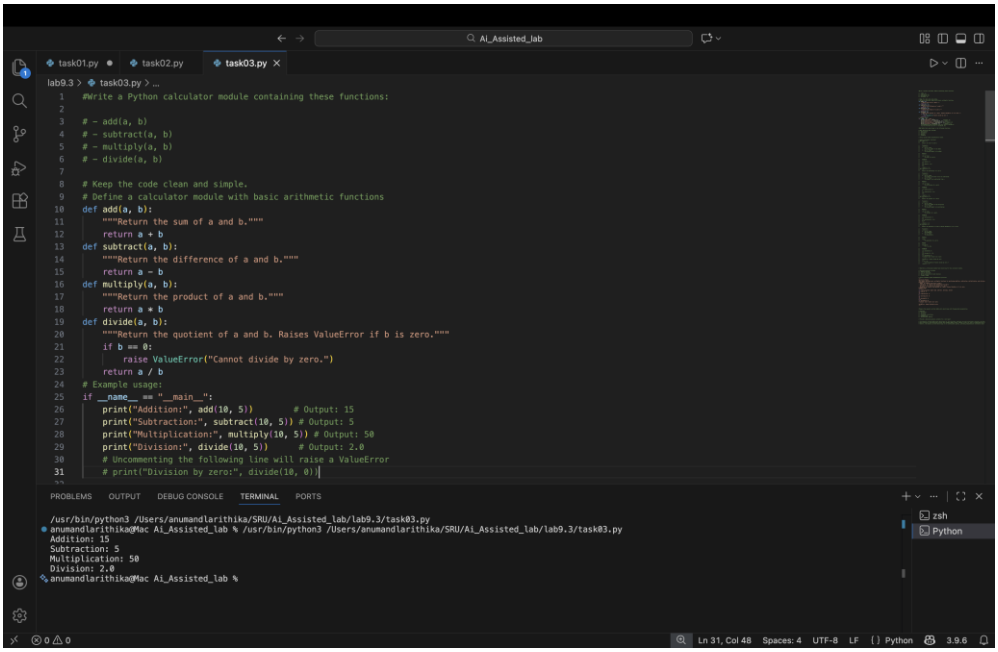
After comparing them, I found:

- AI added comments for almost every line.
- Some AI comments were too detailed.
- Manual comments were shorter and focused.
- AI comments are helpful but need checking.

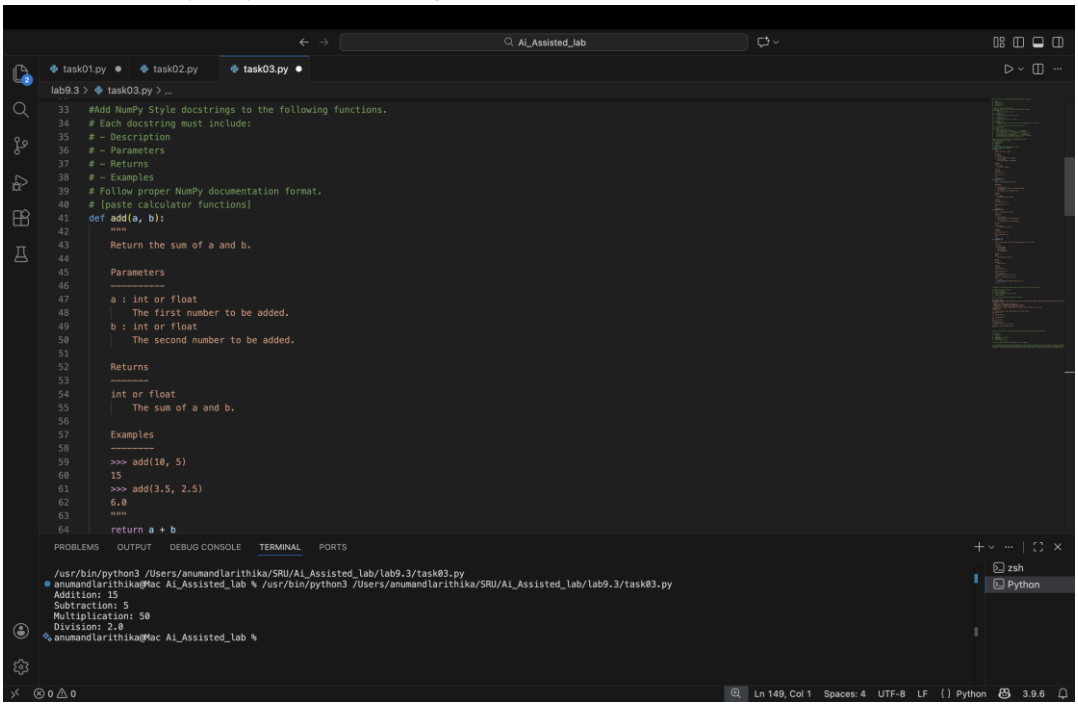
This task helped me understand how AI can automatically explain code logic.

TASK 3 – Module-Level & NumPy Style Documentation :

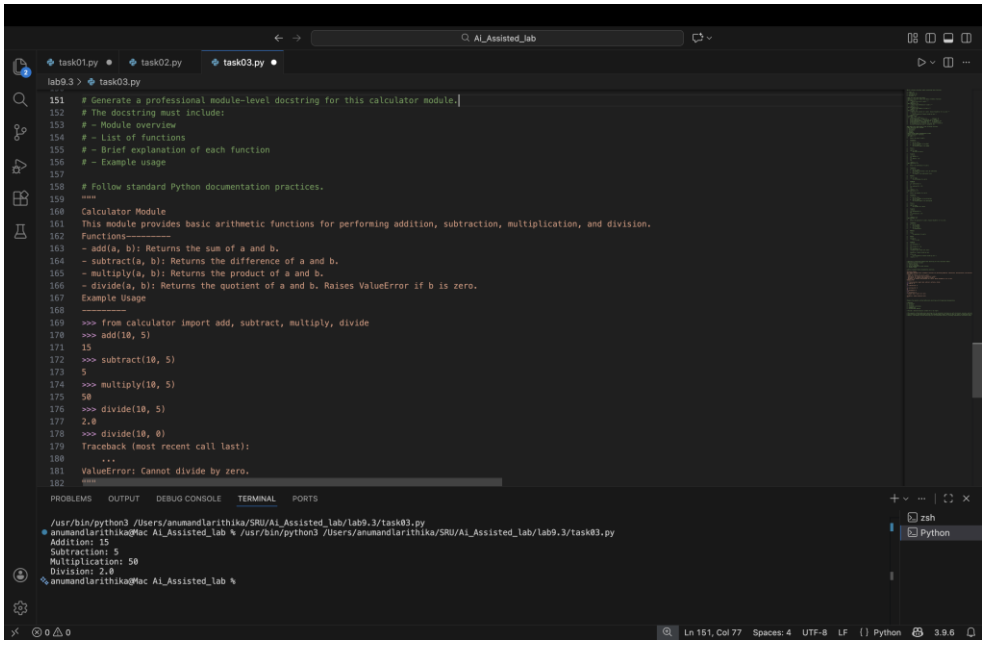
Prompt 1 – Generate Calculator Module :



Prompt 2 – Add NumPy Style Docstrings :



Prompt 3 – Generate Module-Level Docstring :



Explanation :

In this task, I created a small calculator module with functions like add, subtract, multiply, and divide.

First, I wrote NumPy-style docstrings manually for each function.

Then, I used AI to generate:

- A module-level docstring
- Function-level docstrings

After comparing them, I observed:

- AI documentation was well structured.
- It followed proper formatting.
- Manual documentation was simple but correct.
- AI helps in maintaining consistency in large projects.

This task helped me understand structured documentation for multiple functions.