

Ai Assisted Coding

Name : A.Navaneeth

Roll No : 2303A54056

Batch : 47-b

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

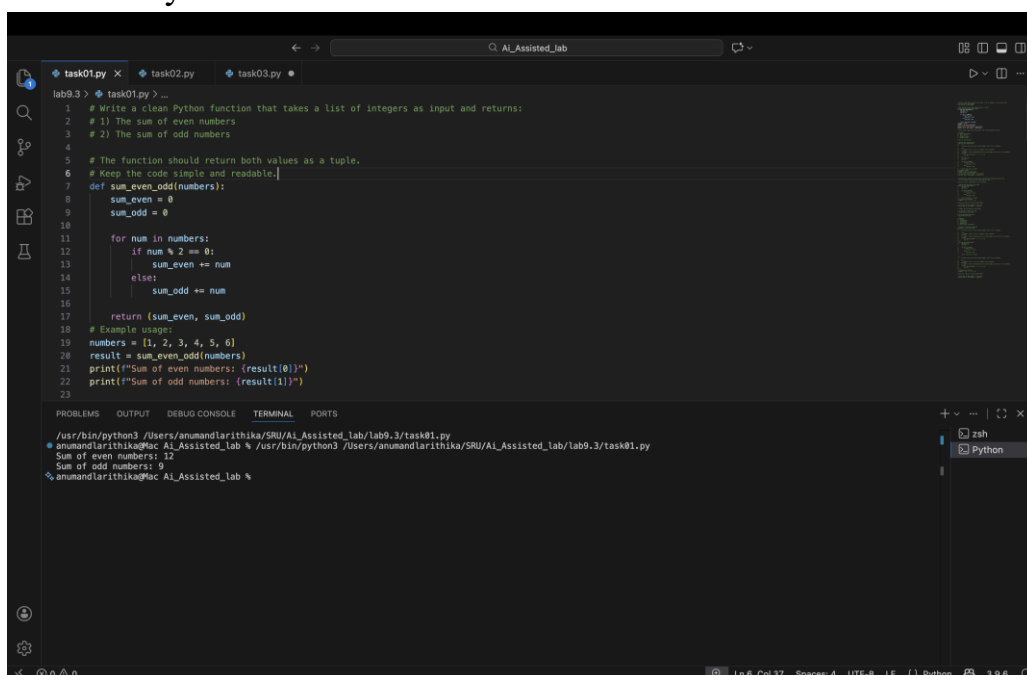
- Write a Python function to return the sum of even numbers and sum of odd numbers in a given list
- Manually add a Google Style docstring to the function
- Use an AI-assisted tool (Copilot / Cursor AI) to generate a function-level docstring
- Compare the AI-generated docstring with the manually written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



```
lab9.3 > task01.py > ...
1 # Write a clean Python function that takes a list of integers and returns:
2 # 1) The sum of even numbers
3 # 2) The sum of odd numbers
4
5 # The function should return both values as a tuple.
6 # Keep the code simple and readable.
7 def sum_even_odd(numbers):
8     sum_even = 0
9     sum_odd = 0
10
11     for num in numbers:
12         if num % 2 == 0:
13             sum_even += num
14         else:
15             sum_odd += num
16
17     return (sum_even, sum_odd)
18
19 # Example usage:
20 numbers = [1, 2, 3, 4, 5, 6]
21 result = sum_even_odd(numbers)
22 print(f"Sum of even numbers: {result[0]}")
23 print(f"Sum of odd numbers: {result[1]}")
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/bin/python3 /Users/anumandlarithika/SRU/AI_Assisted_lab/lab9.3/task01.py
anumandlarithika@Mac AI_Assisted_lab % /usr/bin/python3 /Users/anumandlarithika/SRU/AI_Assisted_lab/lab9.3/task01.py
Sum of even numbers: 12
Sum of odd numbers: 9
anumandlarithika@Mac AI_Assisted_lab %
```

Prompt 2 – Generate Google Style Docstring :

```
24 # Add a proper Google Style docstring to the following Python function.
25 # Include:
26 # - Short description
27 # - Args section
28 # - Returns section
29 # - Example usage
30 # Here is the function:
31 def sum_even_odd(numbers):
32     """
33     Calculate the sum of even and odd numbers from a list of integers.
34
35     Args:
36         numbers (List): A List of integers to be processed.
37     Returns:
38         tuple: A tuple containing the sum of even numbers and the sum of odd numbers.
39     Example:
40         >>> sum_even_odd([1, 2, 3, 4, 5, 6])
41         (12, 9)
42     """
43     sum_even = 0
44     sum_odd = 0
45
46     for num in numbers:
47         if num % 2 == 0:
48             sum_even += num
49         else:
50             sum_odd += num
51
52     return (sum_even, sum_odd)
53
54 # Example usage:
55 numbers = [1, 2, 3, 4, 5, 6]
56 result = sum_even_odd(numbers)
57 print("Sum of even numbers: (result[0])")
58 print("Sum of odd numbers: (result[1])")
```

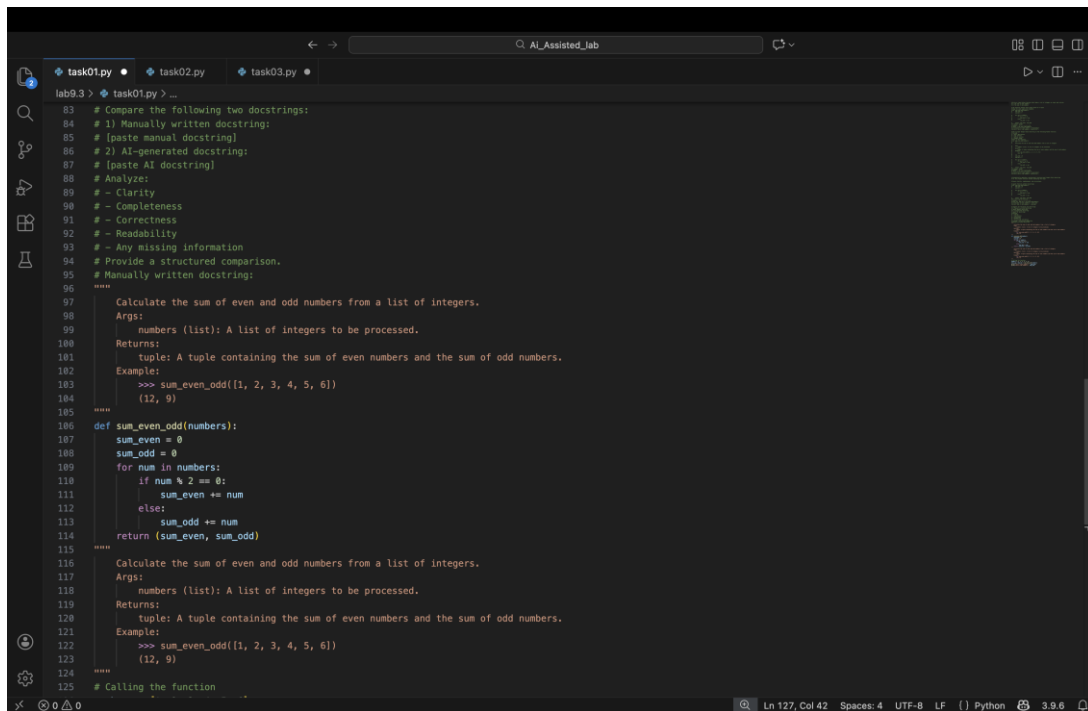
```
Ln 30, Col 24 Spaces: 4 UTF-8 LF Python 3.9.6
```

Prompt 3 – AI Docstring Generation (for comparison)

```
53 # Example usage:
54 numbers = [1, 2, 3, 4, 5, 6]
55 result = sum_even_odd(numbers)
56 print("Sum of even numbers: (result[0])")
57 print("Sum of odd numbers: (result[1])")
58
59 # Automatically generate a professional function-level Google Style docstring
60 # for this Python function without modifying the logic.
61
62 # Ensure clarity, completeness, and correctness.
63
64 # [paste function without docstring]
65
66 def sum_even_odd(numbers):
67     sum_even = 0
68     sum_odd = 0
69
70     for num in numbers:
71         if num % 2 == 0:
72             sum_even += num
73         else:
74             sum_odd += num
75
76     return (sum_even, sum_odd)
77
78 # Calling the function
79 numbers = [1, 2, 3, 4, 5, 6]
80 even_sum, odd_sum = sum_even_odd(numbers)
81 print("Sum of even numbers:", even_sum)
82 print("Sum of odd numbers:", odd_sum)
```

```
Ln 78, Col 29 Spaces: 4 UTF-8 LF Python 3.9.6
```

Prompt 4 – Comparison Prompt :

A screenshot of a code editor window titled 'AI_Assisted_lab'. It shows a Python file 'task01.py' with a function 'sum_even_odd' and its docstrings. The code includes comments comparing manually written and AI-generated docstrings. The manually written docstring is concise, while the AI-generated one is more detailed, including an example and a return value description. The status bar at the bottom indicates 'Ln 127, Col 42', 'Spaces: 4', 'UTF-8', 'LF', 'Python', and '3.9.6'.

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.

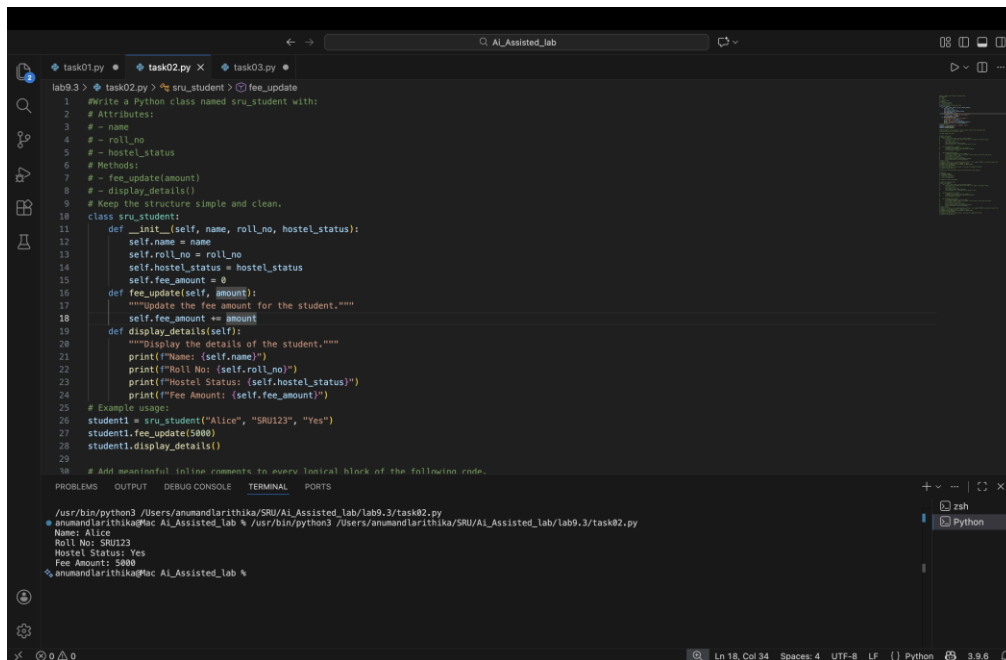
After comparing both, I observed that:

- 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 :

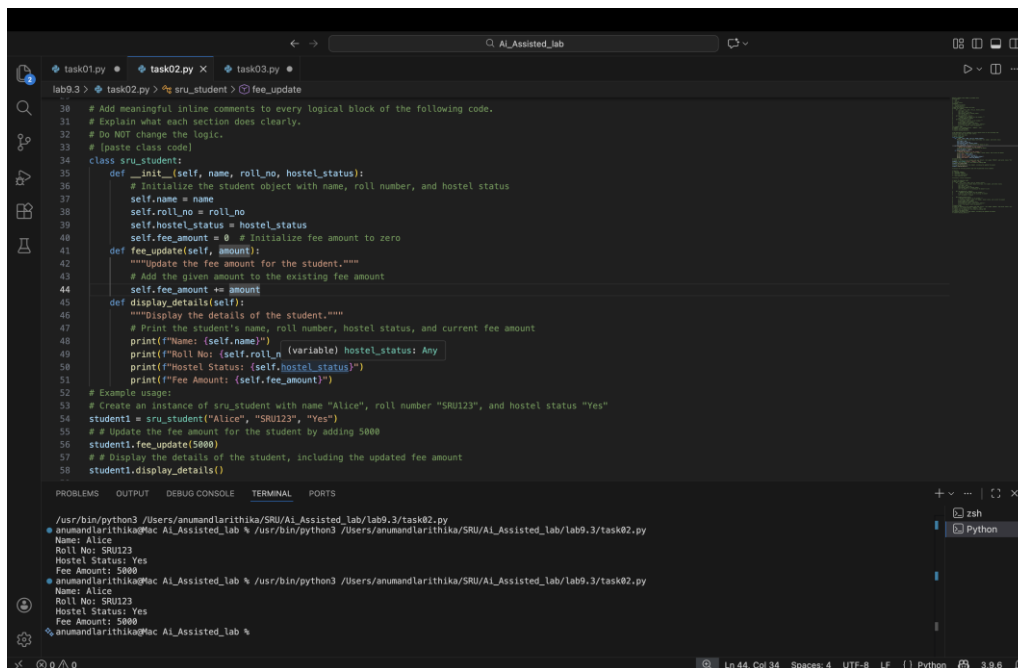
Prompt 1 – Generate Class :



```
lab9.3 > task02.py > % sru_student > fee_update
1 #Write a Python class named sru_student with:
2 # Attributes:
3 # - name
4 # - roll_no
5 # - hostel_status
6 # Methods:
7 # - fee_update(amount)
8 # - display_details()
9 # Keep the structure simple and clean.
10 class sru_student:
11     def __init__(self, name, roll_no, hostel_status):
12         self.name = name
13         self.roll_no = roll_no
14         self.hostel_status = hostel_status
15         self.fee_amount = 0
16     def fee_update(self, amount):
17         """Update the fee amount for the student."""
18         self.fee_amount += amount
19     def display_details(self):
20         """Display the details of the student."""
21         print(f"Name: {self.name}")
22         print(f"Roll No: {self.roll_no}")
23         print(f"Hostel Status: {self.hostel_status}")
24         print(f"Fee Amount: {self.fee_amount}")
25 # Example usage:
26 student1 = sru_student("Alice", "SRU123", "Yes")
27 student1.fee_update(5000)
28 student1.display_details()
29
30 # Add meaningful inline comments to every logical block of the following code.
```

```
/usr/bin/python3 /Users/anumandlarithika/SRU/AI_Assisted_lab/lab9.3/task02.py
Name: Alice
Roll No: SRU123
Hostel Status: Yes
Fee Amount: 5000
anumandlarithika@Mac AI_Assisted_lab %
```

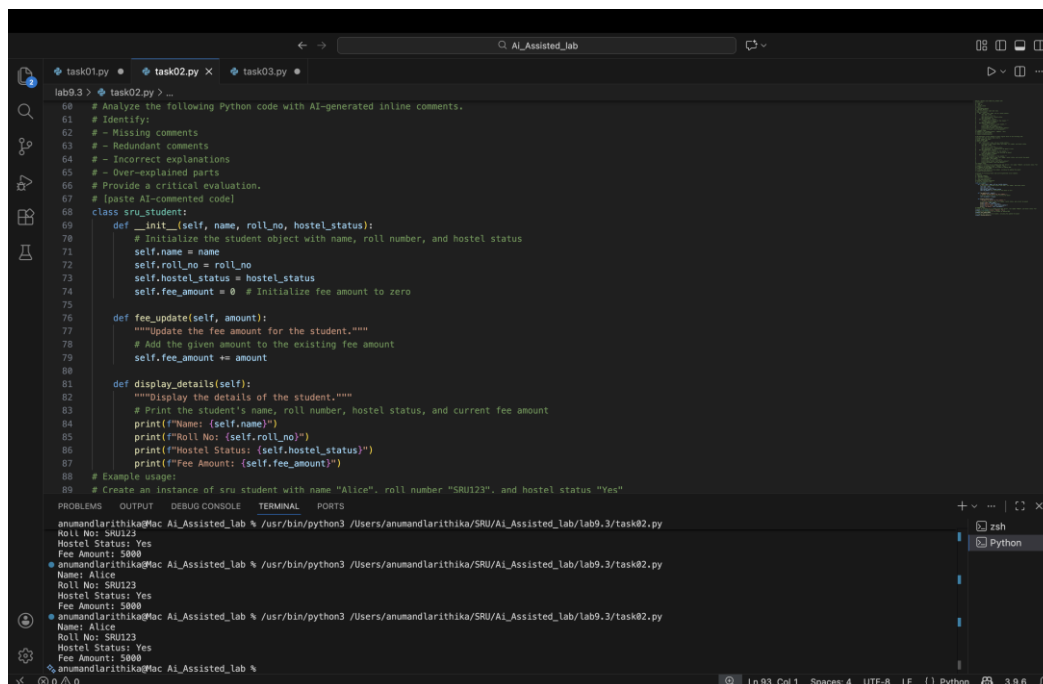
Prompt 2 – Add Manual Inline Comments :



```
30 # Add meaningful inline comments to every logical block of the following code.
31 # Explain what each section does clearly.
32 # Do NOT change the logic.
33 # (paste class code)
34 class sru_student:
35     def __init__(self, name, roll_no, hostel_status):
36         # Initialize the student object with name, roll number, and hostel status
37         self.name = name
38         self.roll_no = roll_no
39         self.hostel_status = hostel_status
40         self.fee_amount = 0 # Initialize fee amount to zero
41     def fee_update(self, amount):
42         """Update the fee amount for the student."""
43         # Add the given amount to the existing fee amount
44         self.fee_amount += amount
45     def display_details(self):
46         """Display the details of the student."""
47         # Print the student's name, roll number, hostel status, and current fee amount
48         print(f"Name: {self.name}")
49         print(f"Roll No: {self.roll_no} (variable) hostel_status: Any")
50         print(f"Hostel Status: {self.hostel_status}")
51         print(f"Fee Amount: {self.fee_amount}")
52 # Example usage:
53 # Create an instance of sru_student with name "Alice", roll number "SRU123", and hostel status "Yes"
54 student1 = sru_student("Alice", "SRU123", "Yes")
55 # Update the fee amount for the student by adding 5000
56 student1.fee_update(5000)
57 # Display the details of the student, including the updated fee amount
58 student1.display_details()
59
```

```
/usr/bin/python3 /Users/anumandlarithika/SRU/AI_Assisted_lab/lab9.3/task02.py
Name: Alice
Roll No: SRU123
Hostel Status: Yes
Fee Amount: 5000
anumandlarithika@Mac AI_Assisted_lab %
```

Prompt 3 – Evaluate AI Comments :



```
lab09.3 > task02.py ...
61 # Analyze the following Python code with AI-generated inline comments.
62 # Identify:
63 # - Missing comments
64 # - Redundant comments
65 # - Incorrect explanations
66 # - Over-explained parts
67 # Provide a critical evaluation.
68 # [paste AI-commented code]
69
70 class sru_student:
71     def __init__(self, name, roll_no, hostel_status):
72         # Initialize the student object with name, roll number, and hostel status
73         self.name = name
74         self.roll_no = roll_no
75         self.hostel_status = hostel_status
76         self.fee_amount = 0 # Initialize fee amount to zero
77
78     def fee_update(self, amount):
79         """Update the fee amount for the student."""
80         # Add the given amount to the existing fee amount
81         self.fee_amount += amount
82
83     def display_details(self):
84         """Display the details of the student."""
85         # Print the student's name, roll number, hostel status, and current fee amount
86         print(f"Name: {self.name}")
87         print(f"Roll No: {self.roll_no}")
88         print(f"Hostel Status: {self.hostel_status}")
89         print(f"Fee Amount: {self.fee_amount}")
90
91 # Example usage:
92 # Create an instance of sru student with name "Alice", roll number "SRU123", and hostel status "Yes"
93
94 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
95 anumandlarithika@AI_Assisted_lab % /usr/bin/python3 /Users/anumandlarithika/SRU/AI_Assisted_lab/lab09.3/task02.py
96 Roll No: SRU123
97 Hostel Status: Yes
98 Fee Amount: 5000
99 anumandlarithika@AI_Assisted_lab % /usr/bin/python3 /Users/anumandlarithika/SRU/AI_Assisted_lab/lab09.3/task02.py
100 Name: Alice
101 Roll No: SRU123
102 Hostel Status: Yes
103 Fee Amount: 5000
104 anumandlarithika@AI_Assisted_lab % /usr/bin/python3 /Users/anumandlarithika/SRU/AI_Assisted_lab/lab09.3/task02.py
105 Name: Alice
106 Roll No: SRU123
107 Hostel Status: Yes
108 Fee Amount: 5000
109 anumandlarithika@AI_Assisted_lab %
```

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.

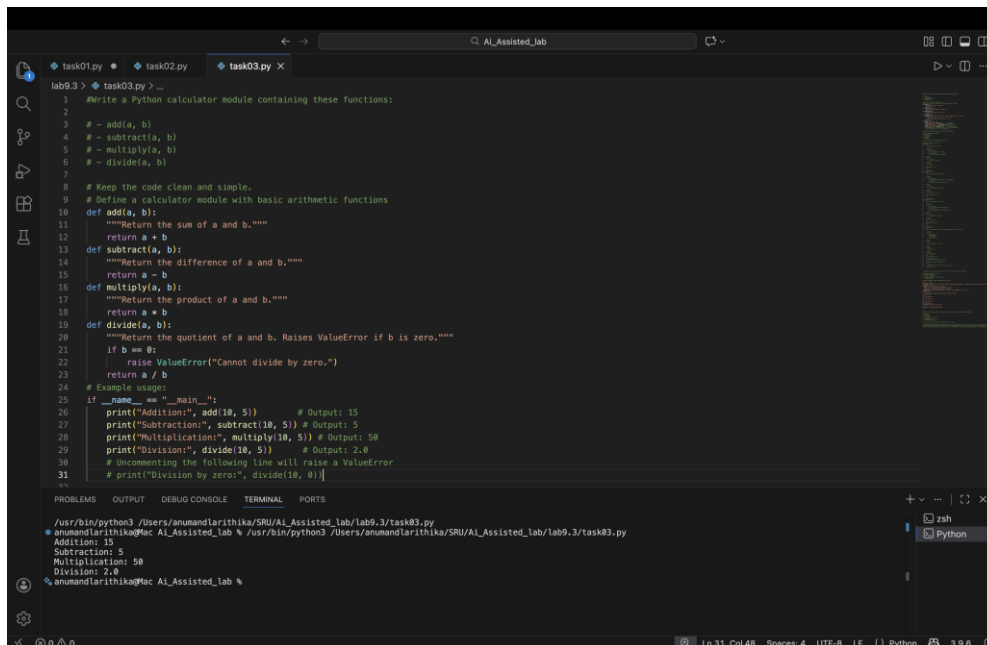
After comparing them, I found:

- 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 :

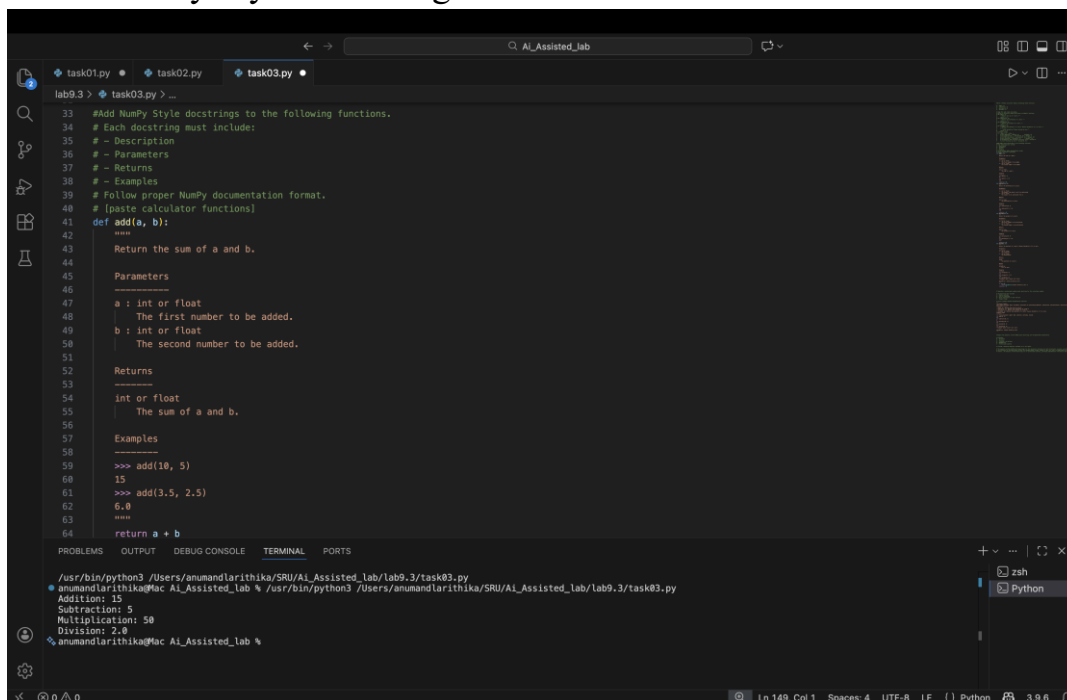


```
lab03 > task03.py > ...
1 #Write a Python calculator module containing these functions:
2
3 # - add(a, b)
4 # - subtract(a, b)
5 # - multiply(a, b)
6 # - divide(a, b)
7
8 # Keep the code clean and simple.
9 # Define a calculator module with basic arithmetic functions
10 def add(a, b):
11     """Return the sum of a and b."""
12     return a + b
13 def subtract(a, b):
14     """Return the difference of a and b."""
15     return a - b
16 def multiply(a, b):
17     """Return the product of a and b."""
18     return a * b
19 def divide(a, b):
20     """Return the quotient of a and b. Raises ValueError if b is zero."""
21     if b == 0:
22         raise ValueError("Cannot divide by zero.")
23     return a / b
24 # Example usage:
25 if __name__ == "__main__":
26     print("Addition:", add(10, 5)) # Output: 15
27     print("Subtraction:", subtract(10, 5)) # Output: 5
28     print("Multiplication:", multiply(10, 5)) # Output: 50
29     print("Division:", divide(10, 5)) # Output: 2.0
30     # Uncommenting the following line will raise a ValueError
31     # print("Division by zero:", divide(10, 0))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/bin/python3 /Users/anumandlarithika/SRU/Al_Assisted_lab/lab9.3/task03.py
anumandlarithika@Mac AI_Assisted_lab % /usr/bin/python3 /Users/anumandlarithika/SRU/Al_Assisted_lab/lab9.3/task03.py
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
anumandlarithika@Mac AI_Assisted_lab %
```

Prompt 2 – Add NumPy Style Docstrings :

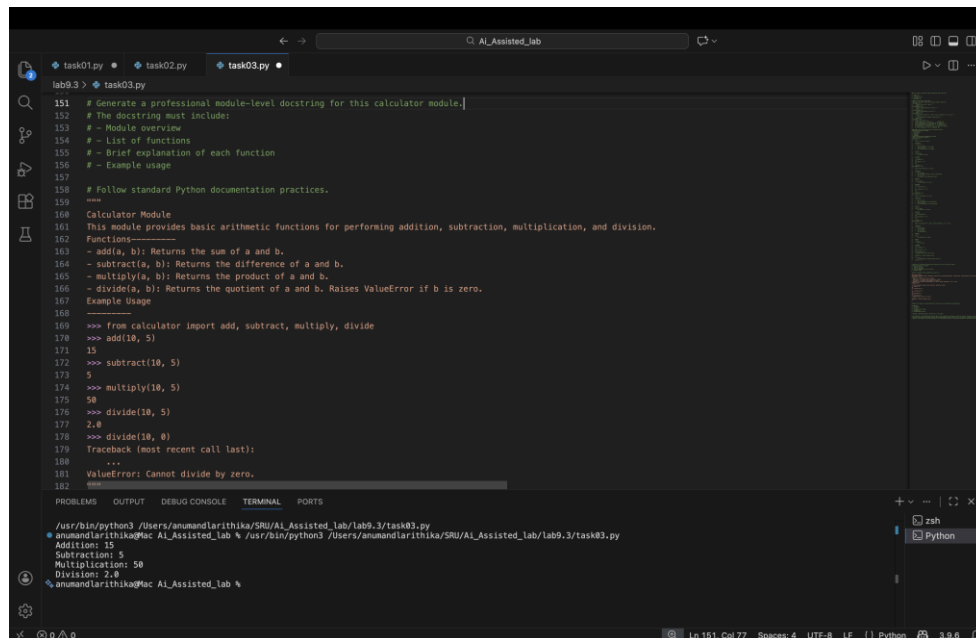


```
lab03 > task03.py > ...
33 Add NumPy Style docstrings to the following functions.
34 # Each docstring must include:
35 # - Description
36 # - Parameters
37 # - Returns
38 # - Examples
39 # Follow proper NumPy documentation format.
40 # [paste calculator functions]
41 def add(a, b):
42     """
43     Return the sum of a and b.
44
45     Parameters
46     -----
47     a : int or float
48         The first number to be added.
49     b : int or float
50         The second number to be added.
51
52     Returns
53     -----
54     int or float
55         The sum of a and b.
56
57     Examples
58     -----
59     >>> add(10, 5)
60     15
61     >>> add(3.5, 2.5)
62     6.0
63     """
64     return a + b
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/bin/python3 /Users/anumandlarithika/SRU/Al_Assisted_lab/lab9.3/task03.py
anumandlarithika@Mac AI_Assisted_lab % /usr/bin/python3 /Users/anumandlarithika/SRU/Al_Assisted_lab/lab9.3/task03.py
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
anumandlarithika@Mac AI_Assisted_lab %
```

Prompt 3 – Generate Module-Level Docstring :



```
lab9.3 > task03.py
151 # Generate a professional module-level docstring for this calculator module.
152 # The docstring must include:
153 # - Module overview
154 # - List of functions
155 # - Brief explanation of each function
156 # - Example usage
157
158 # Follow standard Python documentation practices.
159 """
160 Calculator Module
161 This module provides basic arithmetic functions for performing addition, subtraction, multiplication, and division.
162 Functions-----
163 ~ add(a, b): Returns the sum of a and b.
164 ~ subtract(a, b): Returns the difference of a and b.
165 ~ multiply(a, b): Returns the product of a and b.
166 ~ divide(a, b): Returns the quotient of a and b. Raises ValueError if b is zero.
167 Example Usage
168 -----
169 """
170
171 from calculator import add, subtract, multiply, divide
172
173 >>> add(10, 5)
174 15
175 >>> subtract(10, 5)
176 5
177 >>> multiply(10, 5)
178 50
179 >>> divide(10, 5)
180 2.0
181 >>> divide(10, 0)
182 Traceback (most recent call last):
183 ...
184 ValueError: Cannot divide by zero.
185 """
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/bin/python3 /Users/anumandlarithika/SRU/Al_Assisted_lab/lab9.3/task03.py
anumandlarithika@Mac AI_Assisted_lab % /usr/bin/python3 /Users/anumandlarithika/SRU/Al_Assisted_lab/lab9.3/task03.py
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
anumandlarithika@Mac AI_Assisted_lab %
```

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.