# AI Assisted Coding

Assignment 9.3

Name: R.Bharadwaj

Hall ticket no: 2303A51610

Batch no: 19

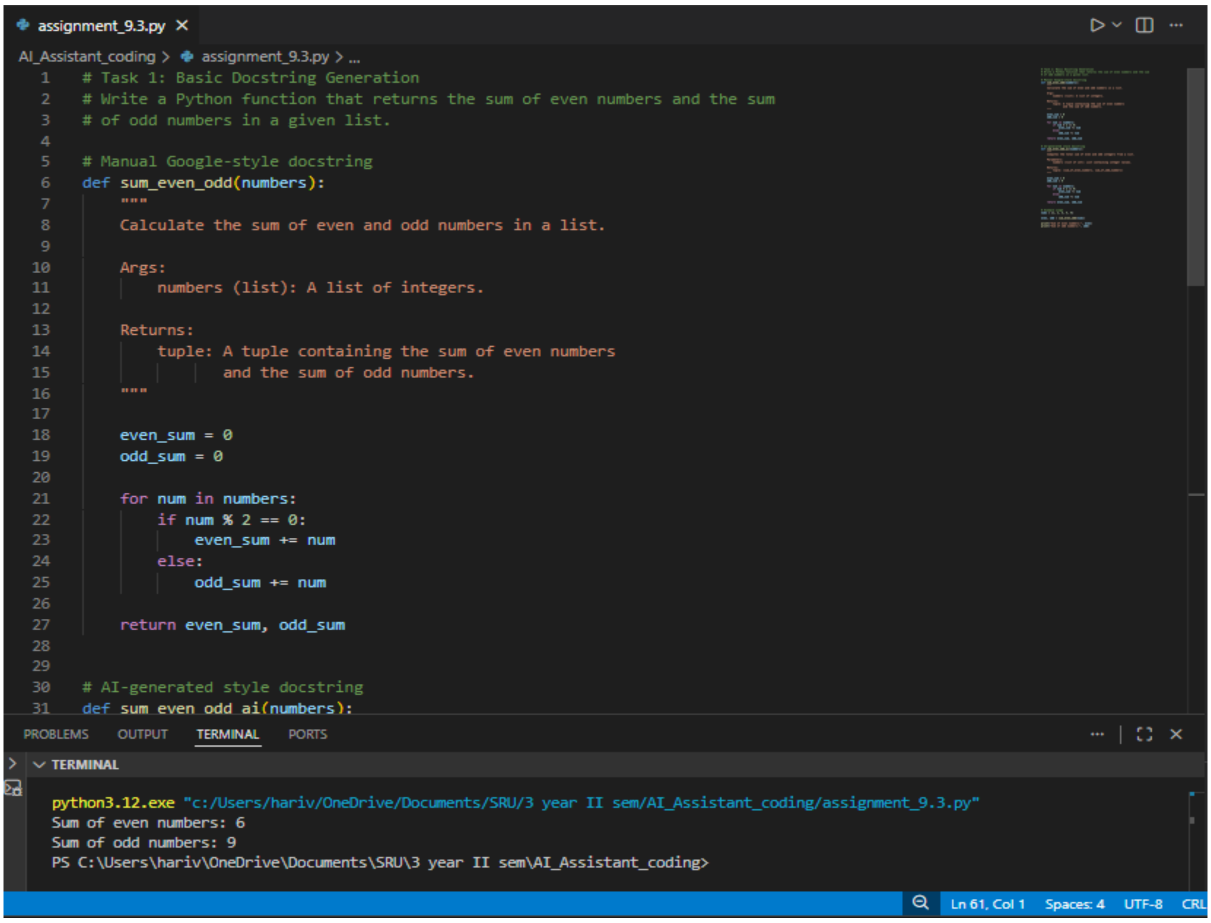
## Task 1: Basic Docstring Generation

## Prompt:

Write a Python function that returns the sum of even numbers and the sum of odd numbers in a given list.

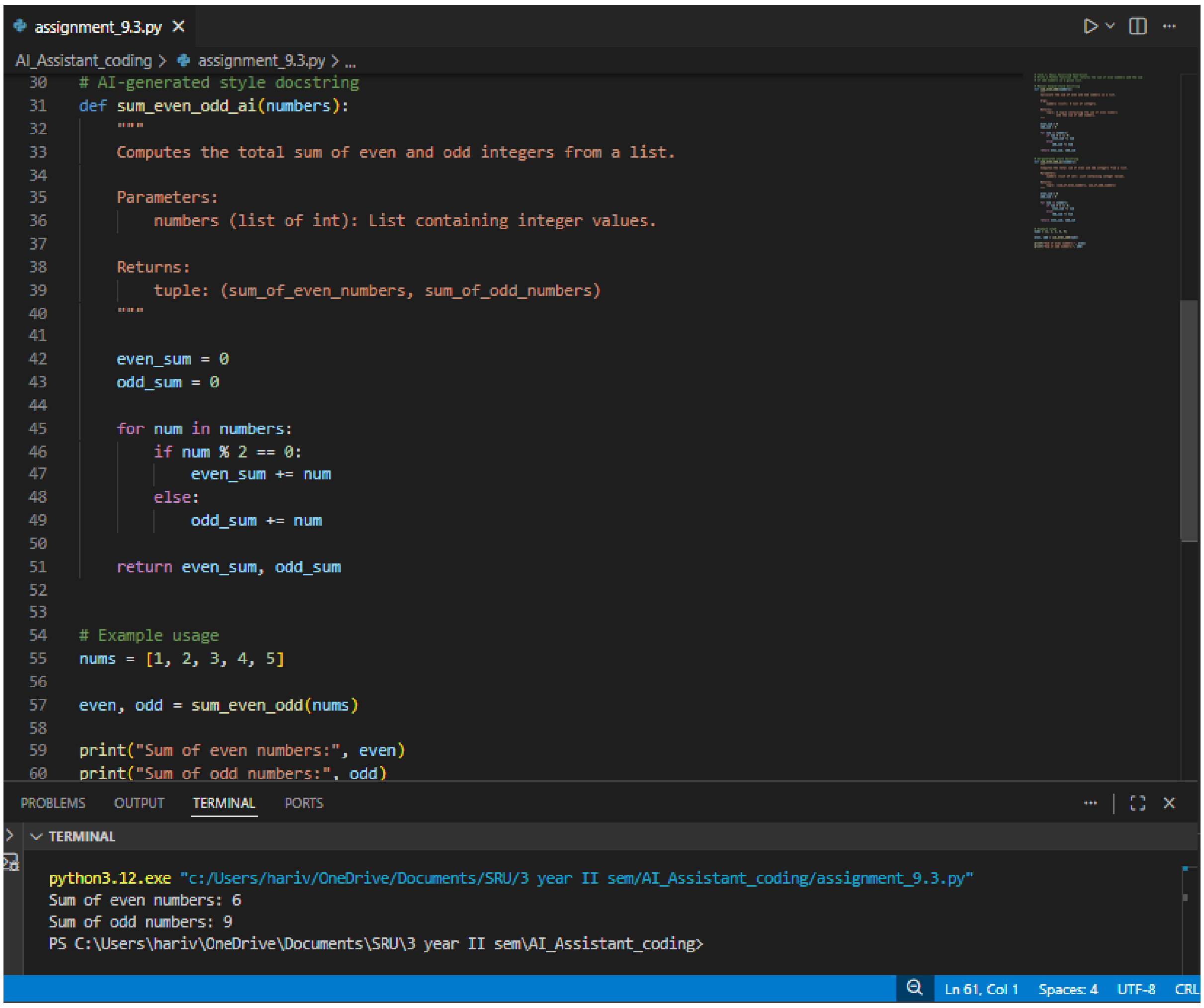
Add a Google Style docstring manually and then generate a docstring using AI assistance for the same function.

## Code & Output:



```python
# Task 1: Basic Docstring Generation
# Write a Python function that returns the sum of even numbers and the sum
# of odd numbers in a given list.

# Manual Google-style docstring
def sum_even_odd(numbers):
    """
    Calculate the sum of even and odd numbers in a list.

    Args:
        numbers (list): A list of integers.

    Returns:
        tuple: A tuple containing the sum of even numbers
            and the sum of odd numbers.
    """

    even_sum = 0
    odd_sum = 0

    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum


# AI-generated style docstring
def sum_even_odd_ai(numbers):
```

```
python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
Sum of even numbers: 6
Sum of odd numbers: 9
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```



```python
# AI-generated style docstring
def sum_even_odd_ai(numbers):
    """
    Computes the total sum of even and odd integers from a list.

    Parameters:
        numbers (list of int): List containing integer values.

    Returns:
        tuple: (sum_of_even_numbers, sum_of_odd_numbers)
    """

    even_sum = 0
    odd_sum = 0

    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum


# Example usage
nums = [1, 2, 3, 4, 5]

even, odd = sum_even_odd(nums)

print("Sum of even numbers:", even)
print("Sum of odd numbers:", odd)
```

```
python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
Sum of even numbers: 6
Sum of odd numbers: 9
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

## Explanation:

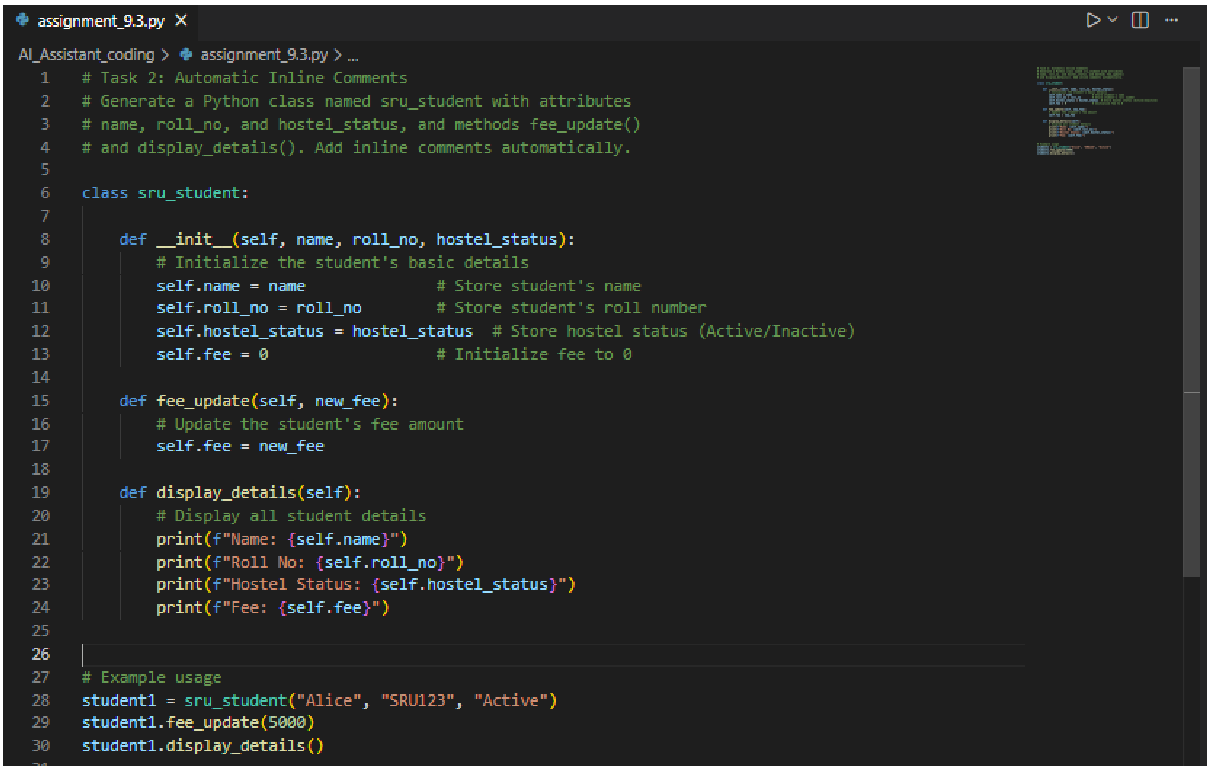
The manual docstring gives a clear and detailed explanation of the function's purpose and return value. The AI-generated version is shorter and accurate but less descriptive. This comparison shows that AI can create correct documentation, yet human refinement is needed to make it more complete and clear.

## Task 2: Automatic Inline Comments

## Prompt:

Generate a Python class named sru_student with attributes name, roll_no, and hostel_status, and methods fee_update() and display_details(). Add inline comments automatically.

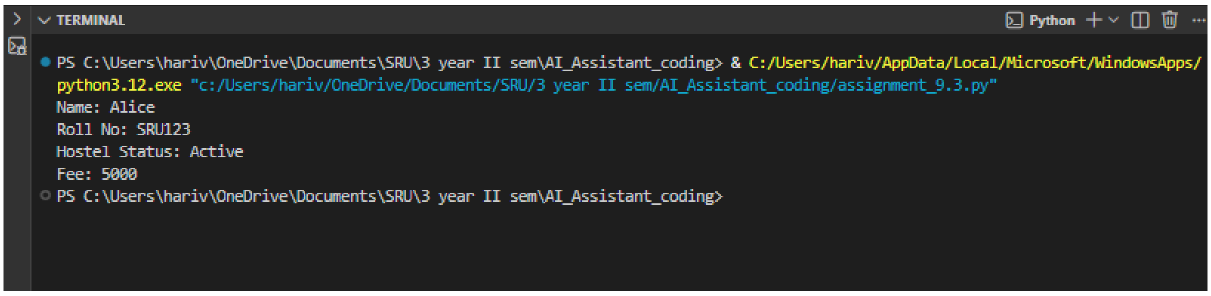
## Code & Output:



assignment_9.3.py

AI_Assistant_coding > assignment_9.3.py > ...

```python
# Task 2: Automatic Inline Comments
# Generate a Python class named sru_student with attributes
# name, roll_no, and hostel_status, and methods fee_update()
# and display_details(). Add inline comments automatically.

class sru_student:

    def __init__(self, name, roll_no, hostel_status):
        # Initialize the student's basic details
        self.name = name               # Store student's name
        self.roll_no = roll_no         # Store student's roll number
        self.hostel_status = hostel_status  # Store hostel status (Active/Inactive)
        self.fee = 0                   # Initialize fee to 0

    def fee_update(self, new_fee):
        # Update the student's fee amount
        self.fee = new_fee

    def display_details(self):
        # Display all student details
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee: {self.fee}")


# Example usage
student1 = sru_student("Alice", "SRU123", "Active")
student1.fee_update(5000)
student1.display_details()
```



TERMINAL

```
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/
python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
Name: Alice
Roll No: SRU123
Hostel Status: Active
Fee: 5000
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

## Explanation:

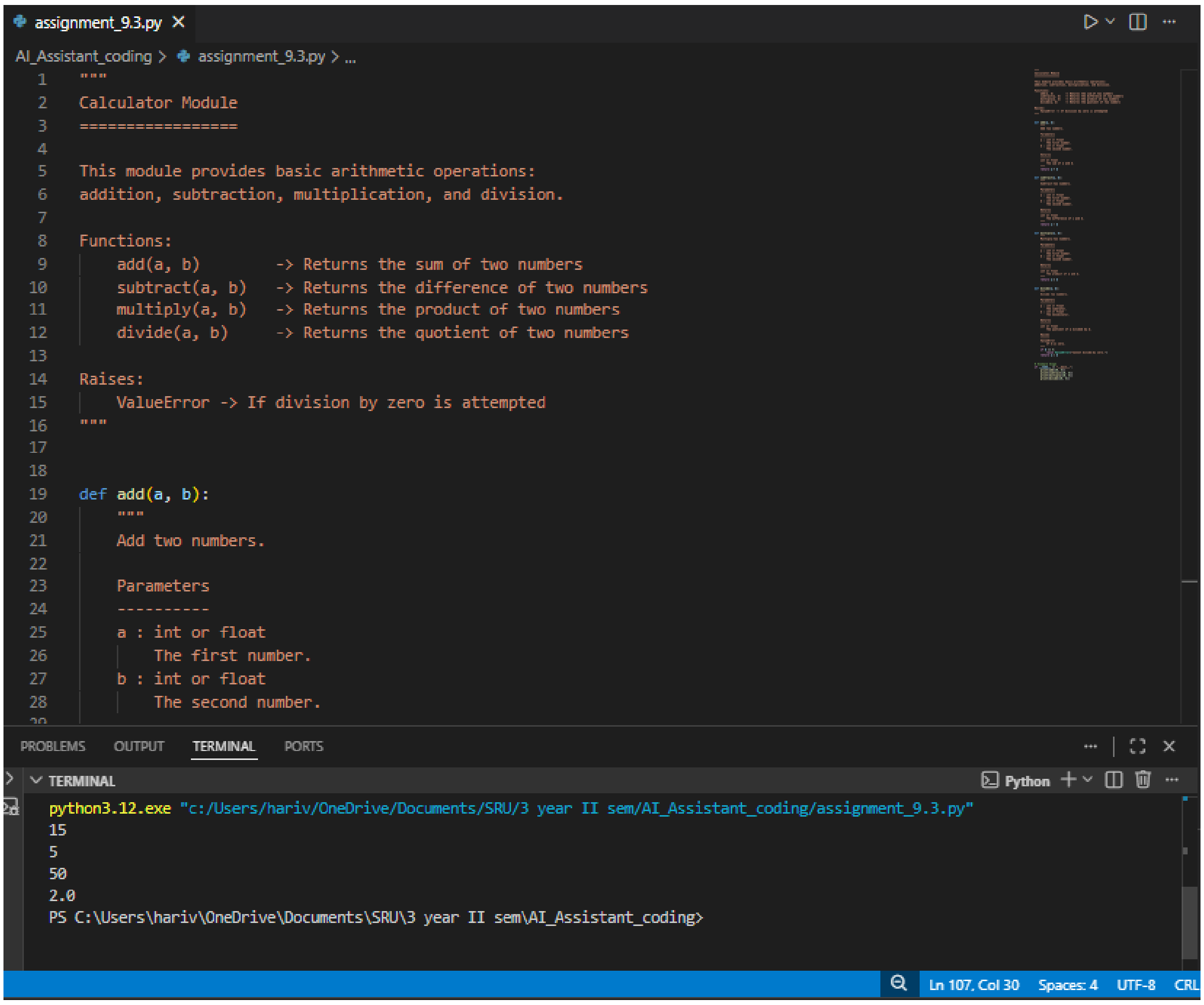
Manual comments are clear, specific, and reflect the developer's intent. AI-generated comments are accurate but tend to be more general and sometimes explain obvious code lines. This shows that while AI can speed up documentation, human review is important to ensure the comments are meaningful and not repetitive.

## Task 3: Module-Level and Function-Level Documentation

### Prompt:

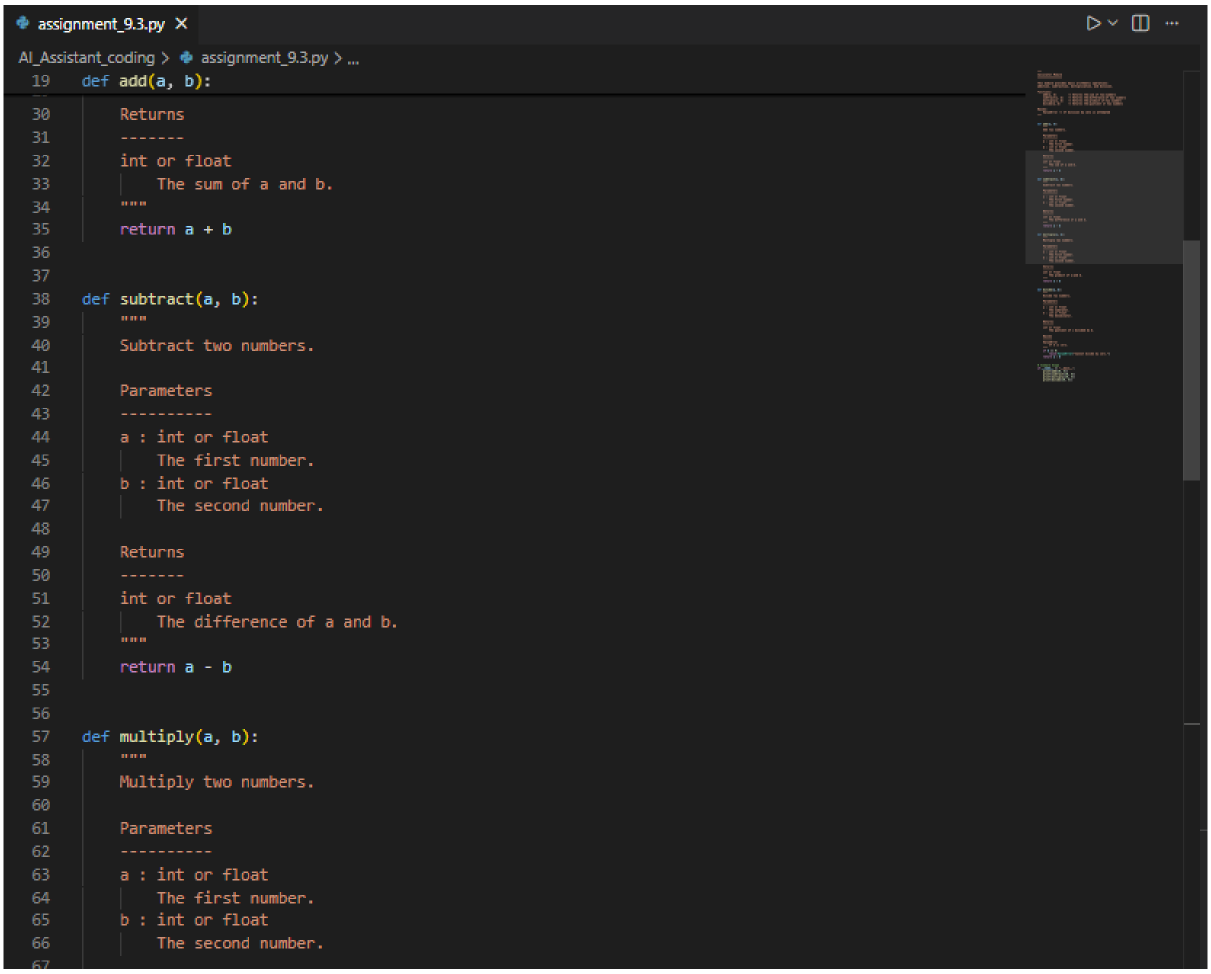
Generate a Python calculator module with functions add, subtract, multiply, and divide. Add NumPy-style docstrings manually and then generate module-level and function-level documentation using AI assistance.

### Code & Output:

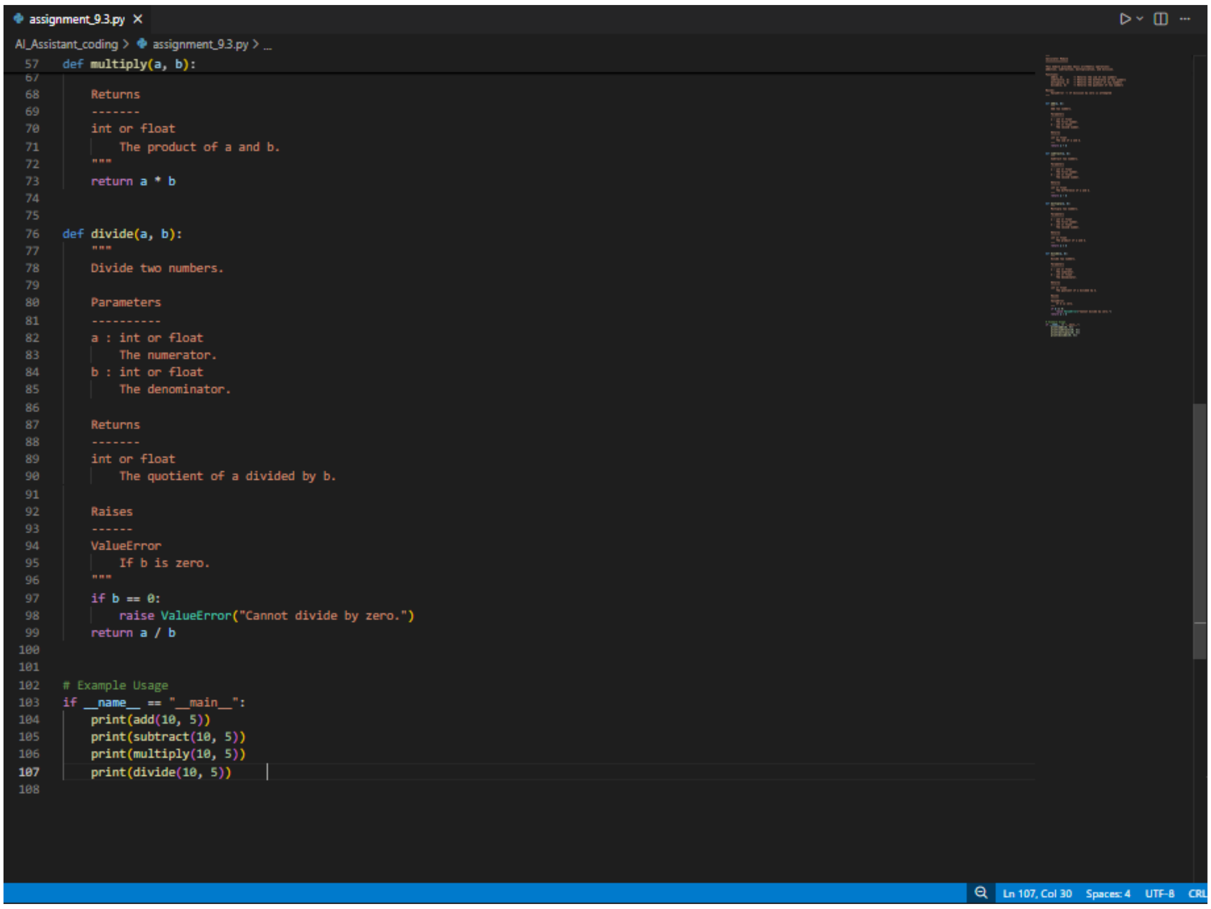


```python
"""
Calculator Module
=================

This module provides basic arithmetic operations:
addition, subtraction, multiplication, and division.

Functions:
    add(a, b)        -> Returns the sum of two numbers
    subtract(a, b)   -> Returns the difference of two numbers
    multiply(a, b)   -> Returns the product of two numbers
    divide(a, b)     -> Returns the quotient of two numbers

Raises:
    ValueError -> If division by zero is attempted
"""


def add(a, b):
    """
    Add two numbers.

    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.
```

```
python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
15
5
50
2.0
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```



```python
def add(a, b):
    Returns
    -------
    int or float
        The sum of a and b.
    """
    return a + b


def subtract(a, b):
    """
    Subtract two numbers.

    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -------
    int or float
        The difference of a and b.
    """
    return a - b


def multiply(a, b):
    """
    Multiply two numbers.

    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.
```



```python
def multiply(a, b):
    Returns
    -------
    int or float
        The product of a and b.
    """
    return a * b


def divide(a, b):
    """
    Divide two numbers.

    Parameters
    ----------
    a : int or float
        The numerator.
    b : int or float
        The denominator.

    Returns
    -------
    int or float
        The quotient of a divided by b.

    Raises
    ------
    ValueError
        If b is zero.
    """
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b


# Example Usage
if __name__ == "__main__":
    print(add(10, 5))
    print(subtract(10, 5))
    print(multiply(10, 5))
    print(divide(10, 5))
```

**Explanation:**

Manual NumPy-style docstrings follow a well-structured scientific format with clear sections for parameters and return values. The AI-generated version is concise and useful for general overviews but does not provide detailed parameter-level explanations. While AI is strong at summarizing, manual documentation offers better technical depth and clarity.