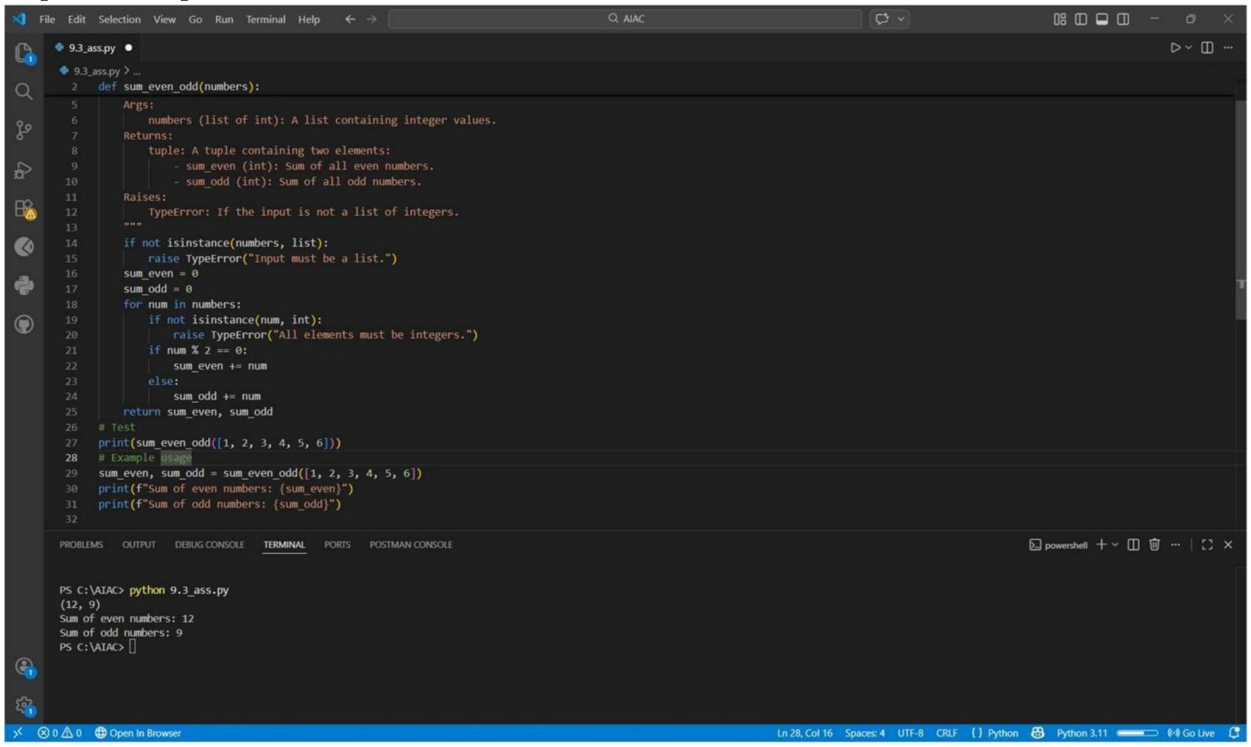
# ASSIGNMENT-9.3

**Name:** V.Pranavith

**HT. No:** 2303A51488

**Batch:** 08
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



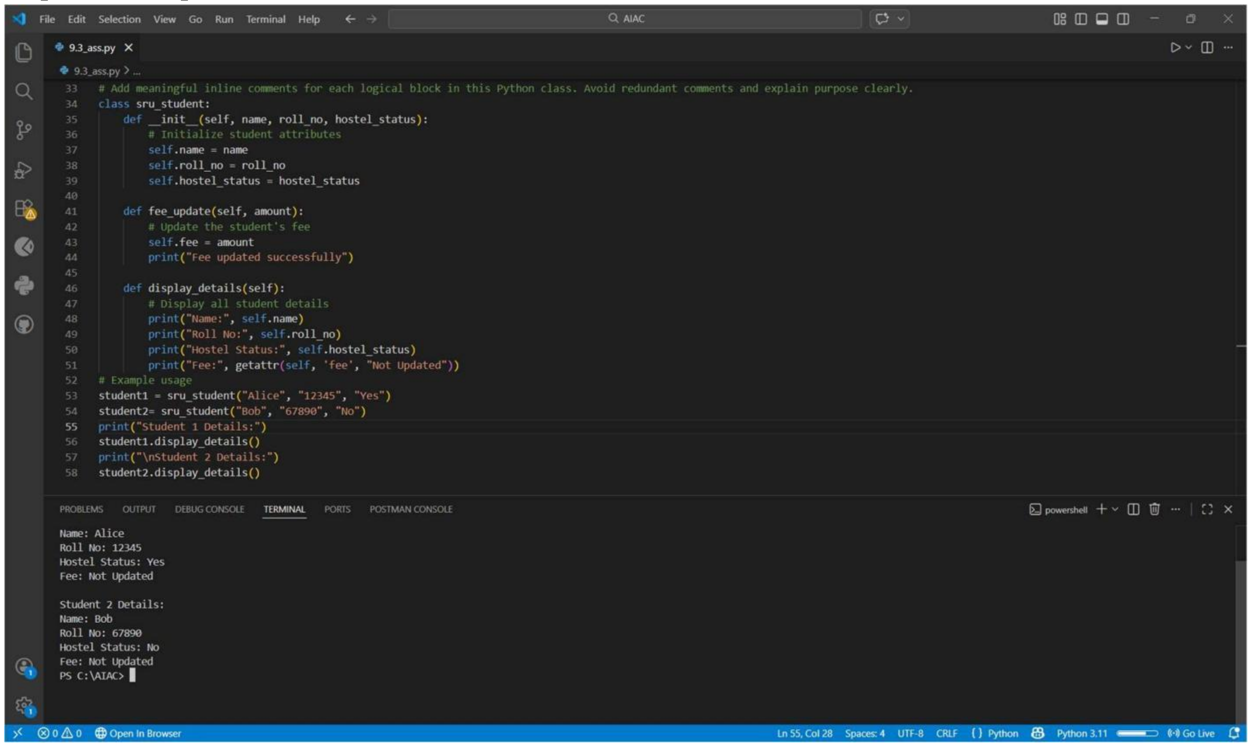
```python
def sum_even_odd(numbers):
    Args:
        numbers (list of int): A list containing integer values.
    Returns:
        tuple: A tuple containing two elements:
            - sum_even (int): Sum of all even numbers.
            - sum_odd (int): Sum of all odd numbers.
    Raises:
        TypeError: If the input is not a list of integers.
    """
    if not isinstance(numbers, list):
        raise TypeError("Input must be a list.")
    sum_even = 0
    sum_odd = 0
    for num in numbers:
        if not isinstance(num, int):
            raise TypeError("All elements must be integers.")
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num
    return sum_even, sum_odd
# Test
print(sum_even_odd([1, 2, 3, 4, 5, 6]))
# Example usage
sum_even, sum_odd = sum_even_odd([1, 2, 3, 4, 5, 6])
print(f"Sum of even numbers: {sum_even}")
print(f"Sum of odd numbers: {sum_odd}")
```

```
PS C:\AIAC> python 9.3_ass.py
(12, 9)
Sum of even numbers: 12
Sum of odd numbers: 9
PS C:\AIAC>
```

Task 2: Automatic Inline Comments
Scenario
You are developing a student management module that must be easy to understand for new developers.
Requirements
• Write a Python program for an sru_student class with the following:
– Attributes: name, roll_no, hostel_status
– Methods: fee_update() and display_details()

• Manually write inline comments for each line or logical block
• Use an AI-assisted tool to automatically add inline comments
• Compare manual comments with AI-generated comments
• Identify missing, redundant, or incorrect AI comments

Expected Output


File Edit Selection View Go Run Terminal Help
AIAC
9.3_ass.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
Open In Browser
Ln 55, Col 28 Spaces: 4 UTF-8 CRLF Python Python 3.11 Go Live

```python
# Add meaningful inline comments for each logical block in this Python class. Avoid redundant comments and explain purpose clearly.
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        # Initialize student attributes
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status

    def fee_update(self, amount):
        # Update the student's fee
        self.fee = amount
        print("Fee updated successfully")

    def display_details(self):
        # Display all student details
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee:", getattr(self, 'fee', "Not Updated"))
# Example usage
student1 = sru_student("Alice", "12345", "Yes")
student2= sru_student("Bob", "67890", "No")
print("Student 1 Details:")
student1.display_details()
print("\nStudent 2 Details:")
student2.display_details()
```

```
Name: Alice
Roll No: 12345
Hostel Status: Yes
Fee: Not Updated

Student 2 Details:
Name: Bob
Roll No: 67890
Hostel Status: No
Fee: Not Updated
PS C:\AIAC>
```

Task 3: Module-Level and Function-Level Documentation

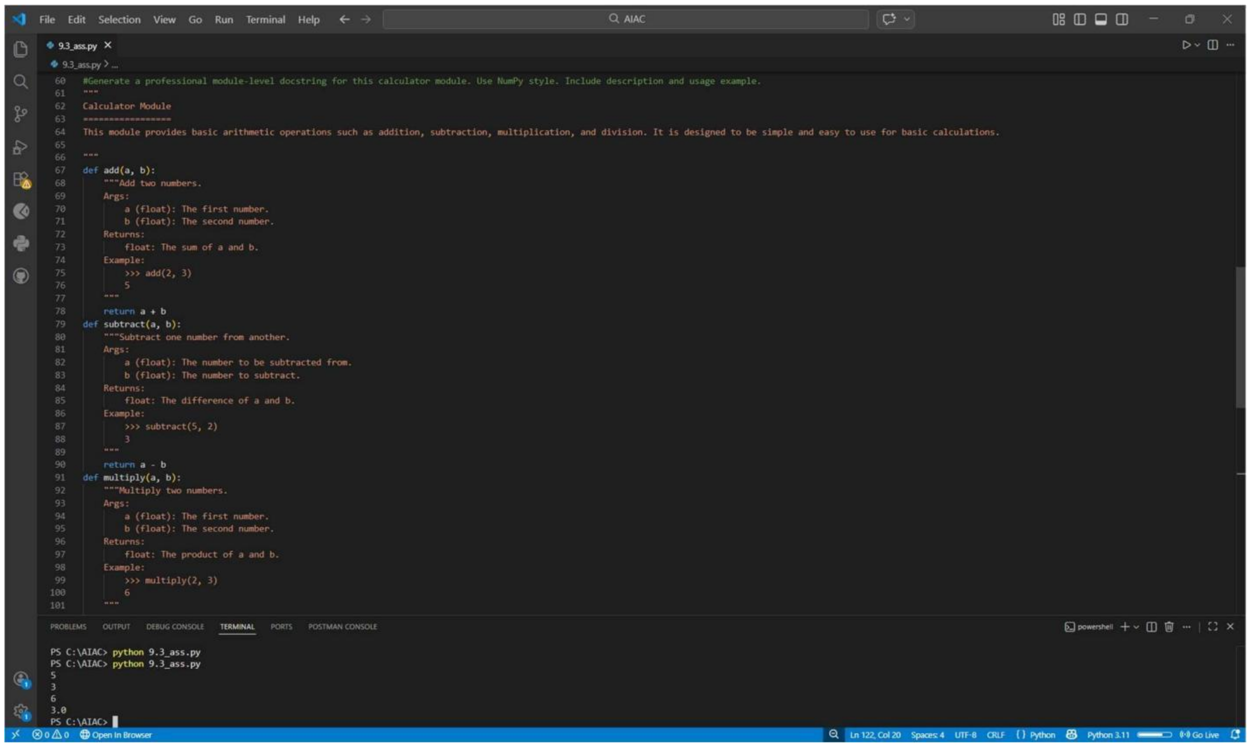
Scenario

You are building a small calculator module that will be shared across multiple projects and requires structured documentation.

Requirements

• Write a Python script containing 3–4 functions (e.g., add, subtract, multiply, divide)
• Manually write NumPy Style docstrings for each function
• Use AI assistance to generate:
– A module-level docstring
– Individual function-level docstrings
• Compare AI-generated docstrings with manually written ones
• Evaluate documentation structure, accuracy, and readability

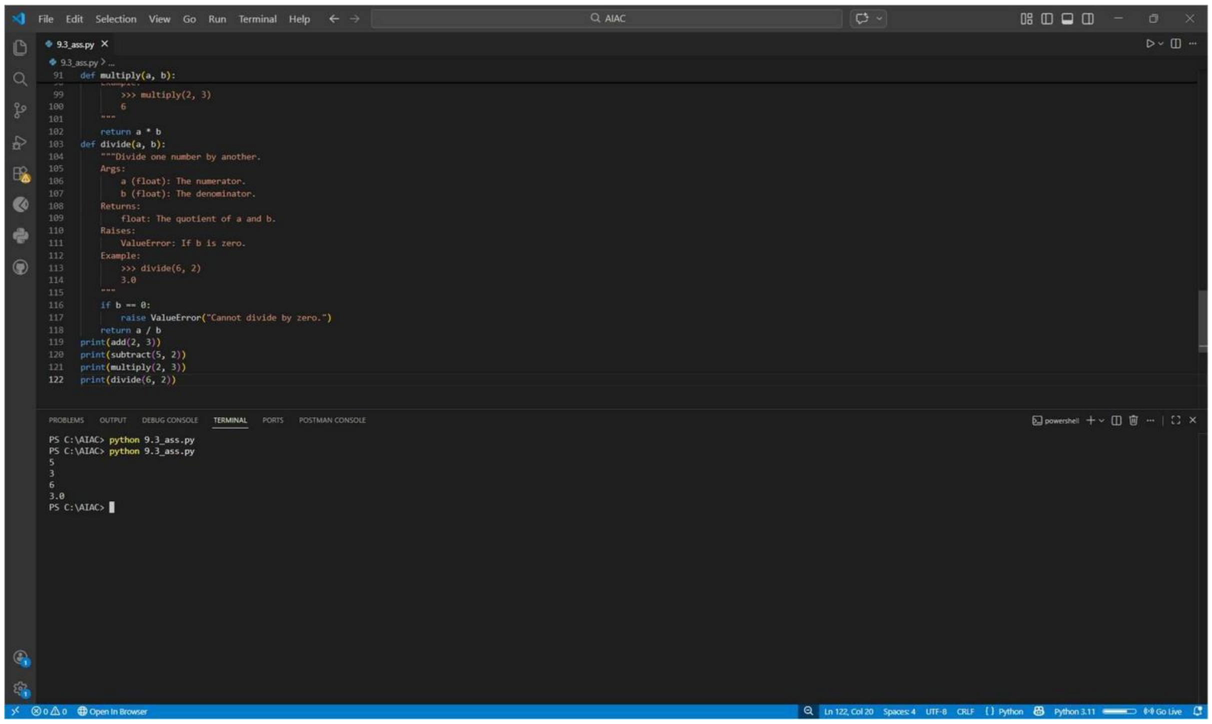
Expected Output



```python
#Generate a professional module-level docstring for this calculator module. Use NumPy style. Include description and usage example.
"""
Calculator Module
=================
This module provides basic arithmetic operations such as addition, subtraction, multiplication, and division. It is designed to be simple and easy to use for basic calculations.

"""
def add(a, b):
    """Add two numbers.
    Args:
        a (float): The first number.
        b (float): The second number.
    Returns:
        float: The sum of a and b.
    Example:
        >>> add(2, 3)
        5
    """
    return a + b
def subtract(a, b):
    """Subtract one number from another.
    Args:
        a (float): The number to be subtracted from.
        b (float): The number to subtract.
    Returns:
        float: The difference of a and b.
    Example:
        >>> subtract(5, 2)
        3
    """
    return a - b
def multiply(a, b):
    """Multiply two numbers.
    Args:
        a (float): The first number.
        b (float): The second number.
    Returns:
        float: The product of a and b.
    Example:
        >>> multiply(2, 3)
        6
    """
```

```
PS C:\AIAC> python 9.3_ass.py
PS C:\AIAC> python 9.3_ass.py
5
3
6
3.0
PS C:\AIAC>
```



```python
def multiply(a, b):
        >>> multiply(2, 3)
        6
    """
    return a * b
def divide(a, b):
    """Divide one number by another.
    Args:
        a (float): The numerator.
        b (float): The denominator.
    Returns:
        float: The quotient of a and b.
    Raises:
        ValueError: If b is zero.
    Example:
        >>> divide(6, 2)
        3.0
    """
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b
print(add(2, 3))
print(subtract(5, 2))
print(multiply(2, 3))
print(divide(6, 2))
```

```
PS C:\AIAC> python 9.3_ass.py
PS C:\AIAC> python 9.3_ass.py
5
3
6
3.0
PS C:\AIAC>
```