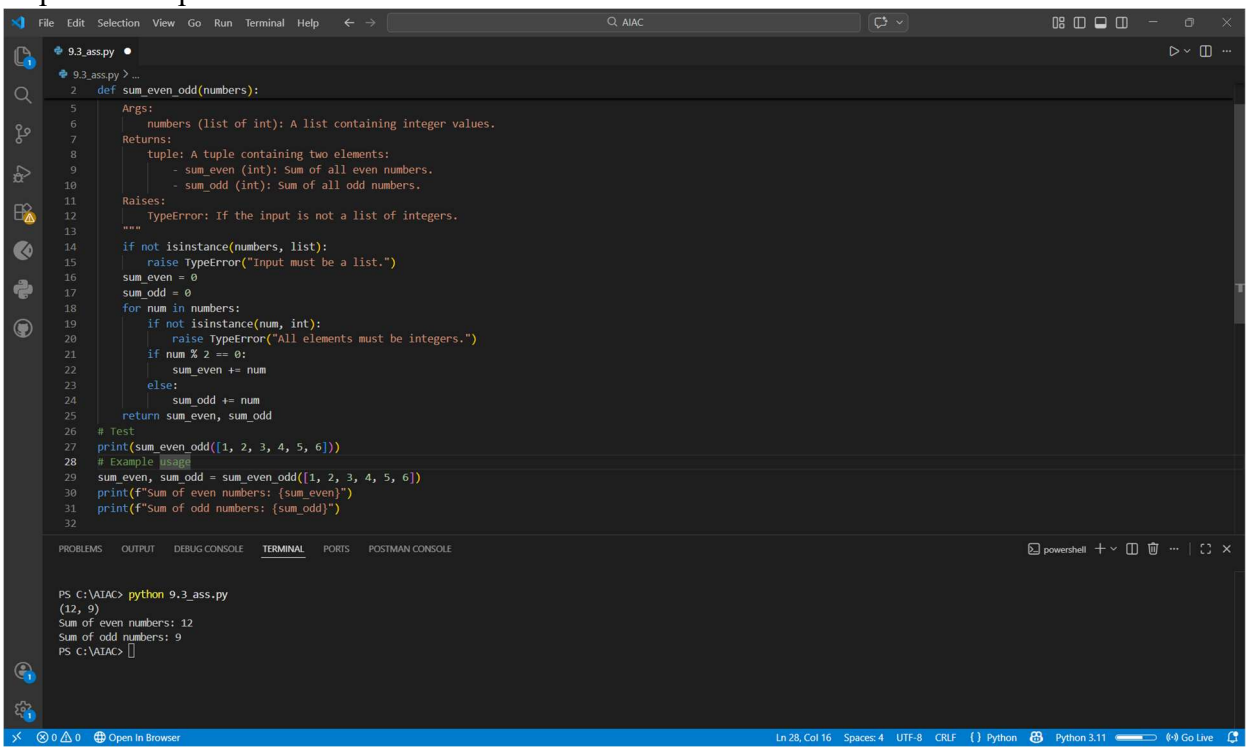
ASSIGNMENT-9.3

Name: J.Vyshnavi

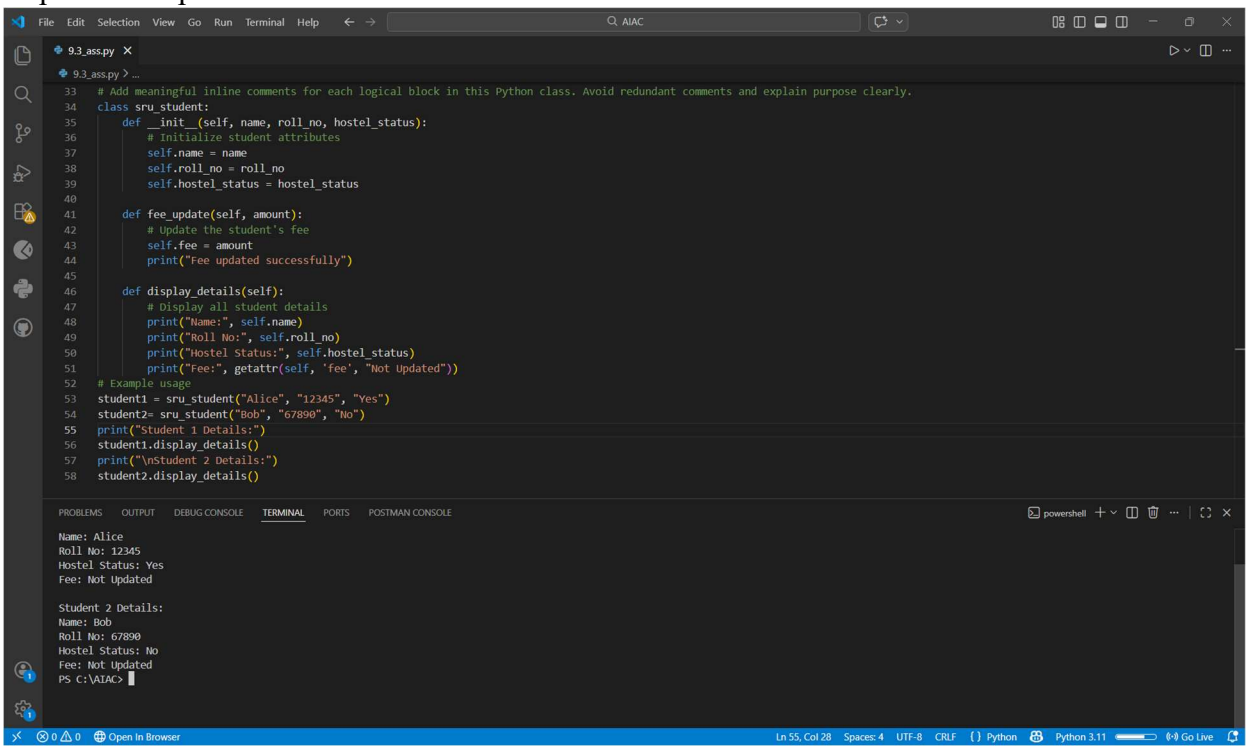
HT. No: 2303A51895

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

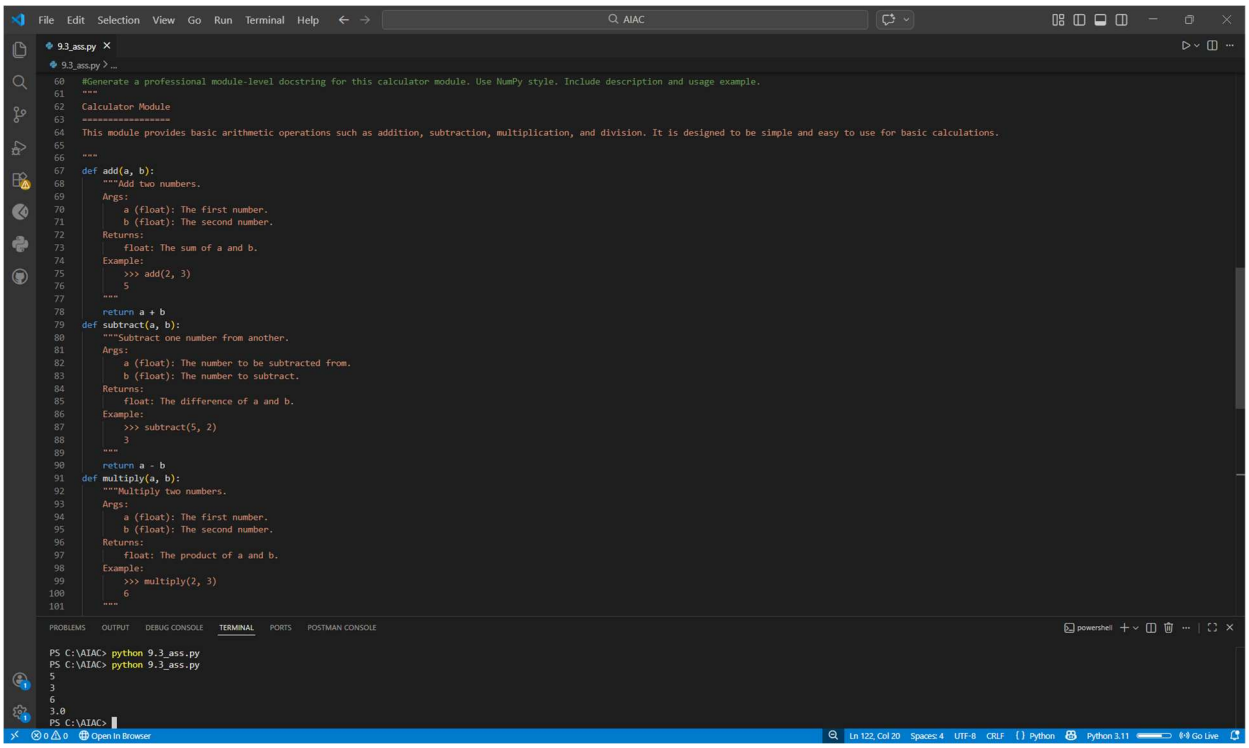


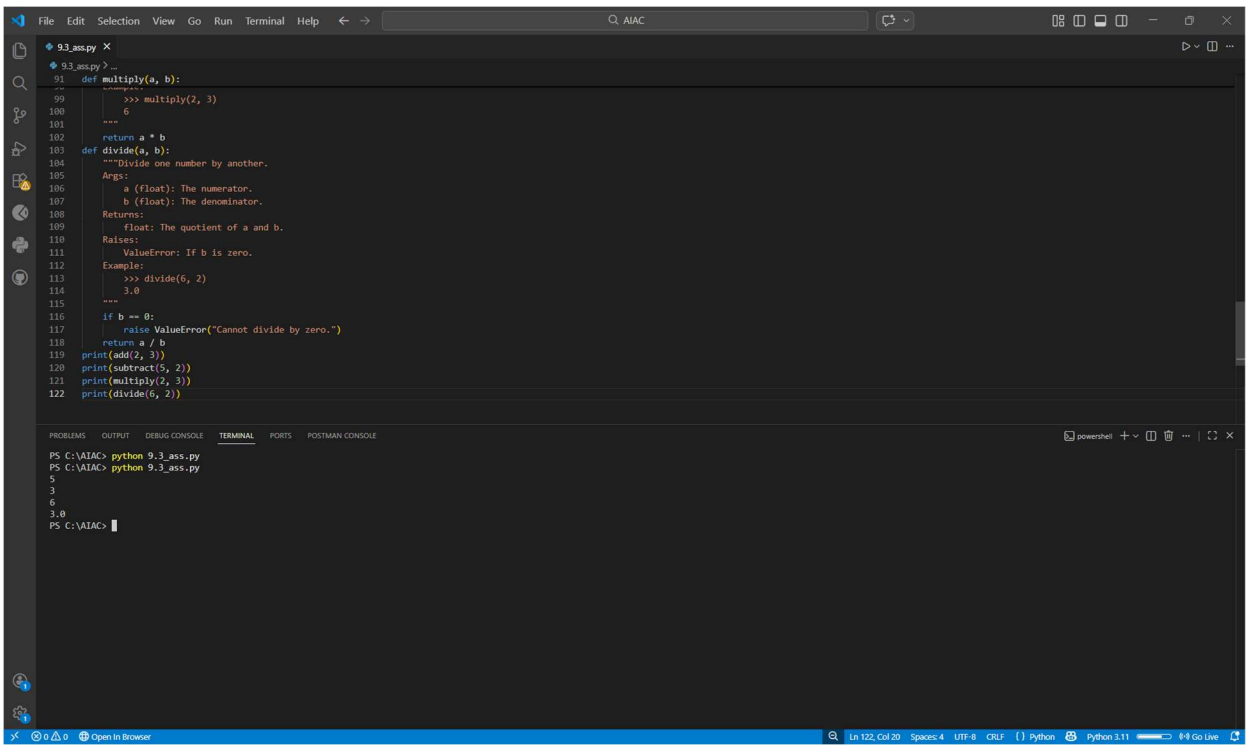
Task 2: Automatic Inline Comments
Scenario
You are developing a student management module that must be easy to understand for new developers.
Requirements
• Write a Python program for an sru_student class with the following:
– Attributes: name, roll_no, hostel_status
– Methods: fee_update() and display_details()

• 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



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


File Edit Selection View Go Run Terminal Help
AIAC
9.3_ass.py
9.3_ass.py > ...
#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
"""
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
PS C:\AIAC> python 9.3_ass.py
PS C:\AIAC> python 9.3_ass.py
5
3
6
3.0
PS C:\AIAC>
0 0 Open in Browser
Ln 122, Col 20 Spaces: 4 UTF-8 CRLF Python Python 3.11 Go Live


File Edit Selection View Go Run Terminal Help
AIAC
9.3_ass.py
9.3_ass.py > ...
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))
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
PS C:\AIAC> python 9.3_ass.py
PS C:\AIAC> python 9.3_ass.py
5
3
6
3.0
PS C:\AIAC>
0 0 Open in Browser
Ln 122, Col 20 Spaces: 4 UTF-8 CRLF Python Python 3.11 Go Live