ASSIGNMENT- 9.3

Name: POOJITHA.EDDE

HT.No: 2303A51356

Batch: 20

Task 1: Basic Docstring Generation

Scenario

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

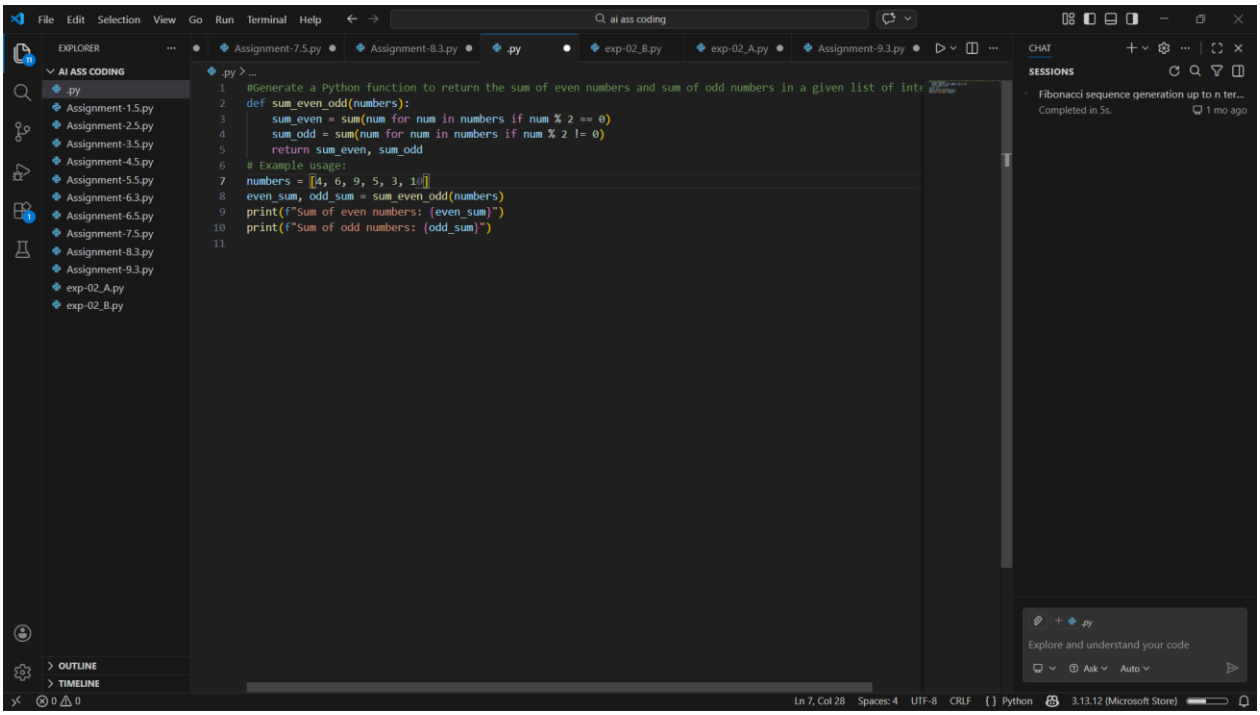
Requirements

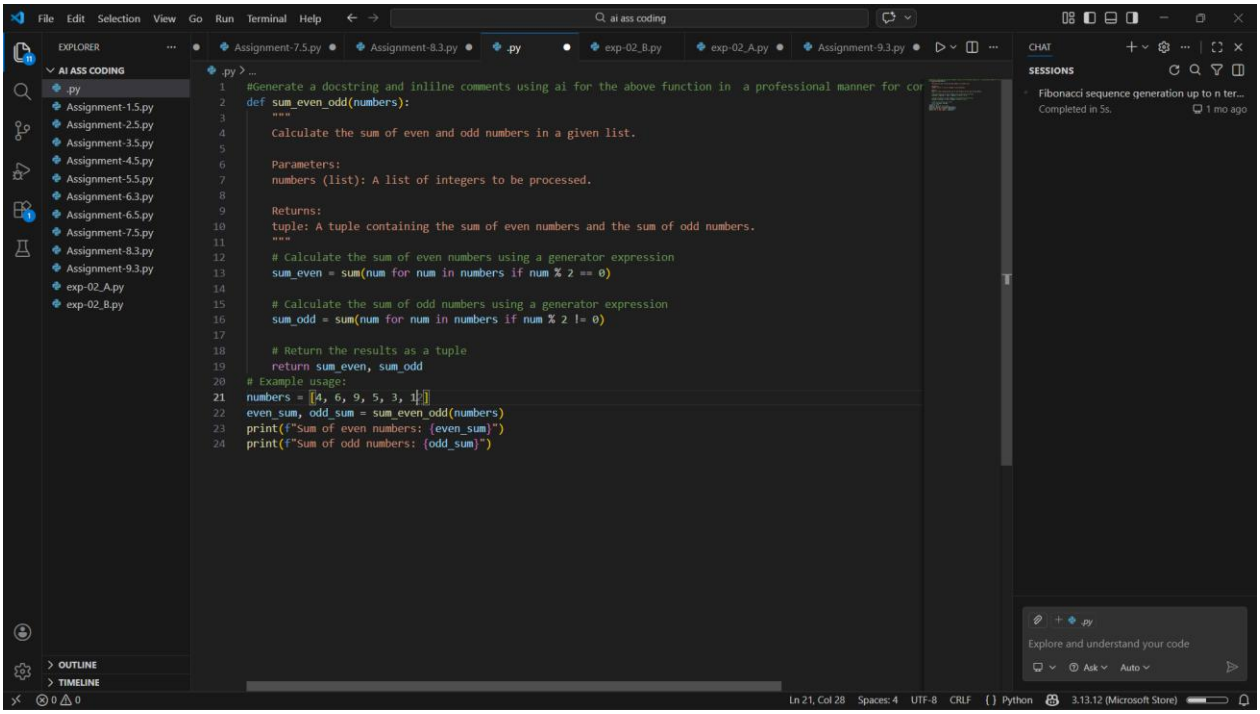
- Write a Python function to return the sum of even numbers and sum of odd numbers in a given list
- Manually add a Google Style docstring to the function
- Use an AI-assisted tool (Copilot / Cursor AI) to generate a function-level docstring
- Compare the AI-generated docstring with the manually written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

Expected Output

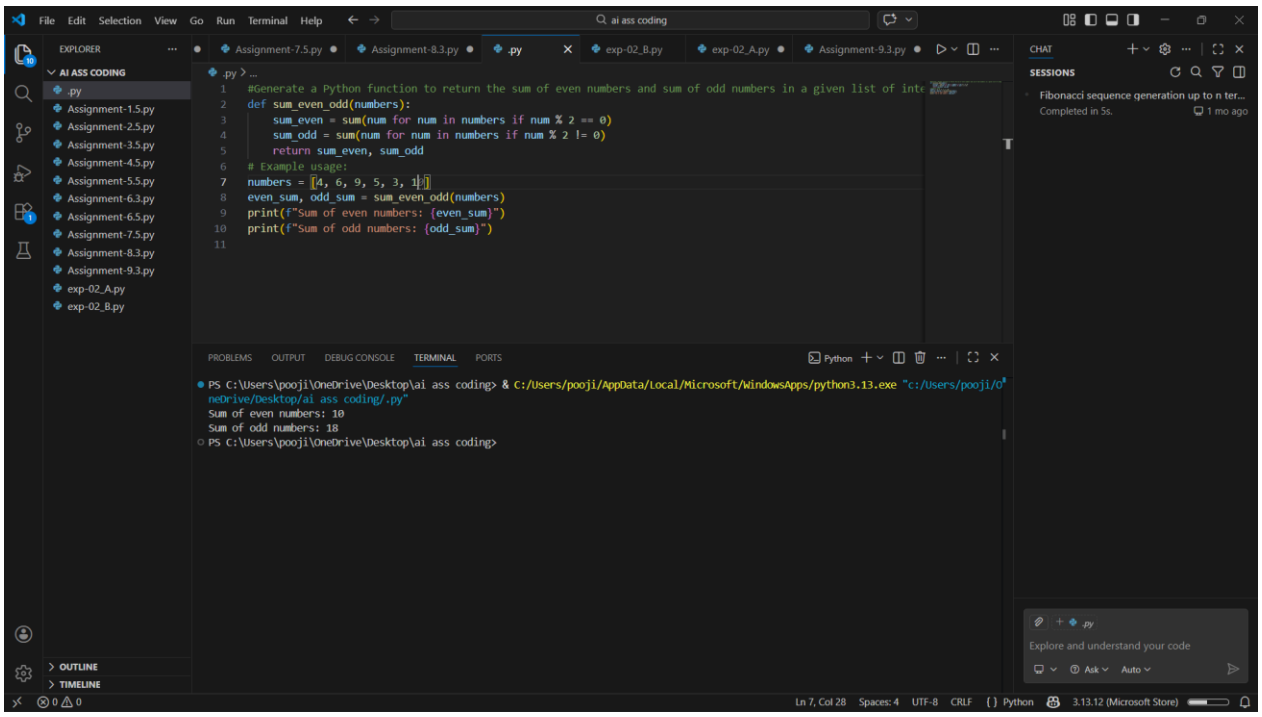
- Python function with manual Google-style docstring
- AI-generated docstring for the same function
- Comparison explaining differences between manual and AI-generated documentation • Improved understanding of AI-generated function-level documentation

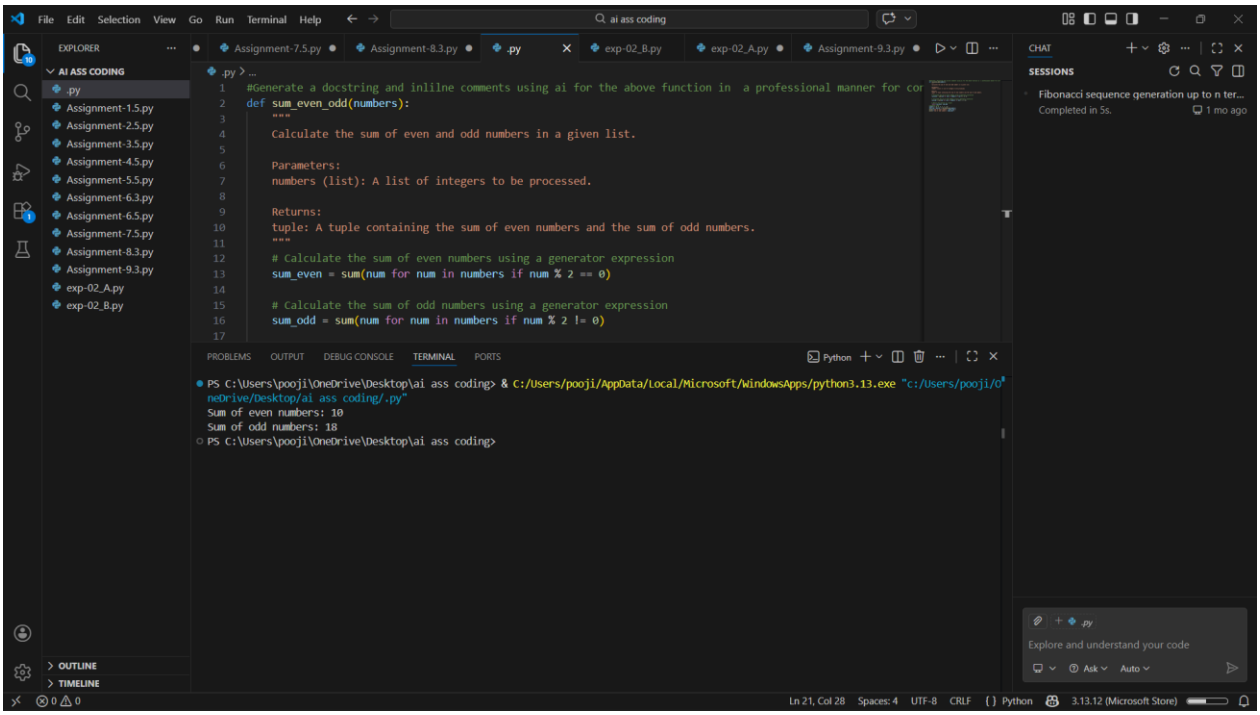
CODE:




File Edit Selection View Go Run Terminal Help
ai ass coding
EXPLORER
AI ASS CODING
.py
Assignment-1.5.py
Assignment-2.5.py
Assignment-3.5.py
Assignment-4.5.py
Assignment-5.5.py
Assignment-6.3.py
Assignment-6.5.py
Assignment-7.5.py
Assignment-8.3.py
Assignment-9.3.py
exp-02_A.py
exp-02_B.py
Assignment-7.5.py
Assignment-8.3.py
.py
exp-02_B.py
exp-02_A.py
Assignment-9.3.py
#Generate a docstring and inlilne comments using ai for the above function in a professional manner for cor
def sum_even_odd(numbers):
"""
Calculate the sum of even and odd numbers in a given list.
Parameters:
numbers (list): A list of integers to be processed.
Returns:
tuple: A tuple containing the sum of even numbers and the sum of odd numbers.
"""
# Calculate the sum of even numbers using a generator expression
sum_even = sum(num for num in numbers if num % 2 == 0)
# Calculate the sum of odd numbers using a generator expression
sum_odd = sum(num for num in numbers if num % 2 != 0)
# Return the results as a tuple
return sum_even, sum_odd
# Example usage:
numbers = [4, 6, 9, 5, 3, 1]
even_sum, odd_sum = sum_even_odd(numbers)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
CHAT
SESSIONS
Fibonacci sequence generation up to n ter...
Completed in 5s.
1 mo ago
Explore and understand your code
Ask
Auto
OUTLINE
TIMELINE
Ln 21, Col 28
Spaces: 4
UTF-8
CRLF
Python
3.13.12 (Microsoft Store)

OUTPUT:


File Edit Selection View Go Run Terminal Help
ai ass coding
EXPLORER
AI ASS CODING
.py
Assignment-1.5.py
Assignment-2.5.py
Assignment-3.5.py
Assignment-4.5.py
Assignment-5.5.py
Assignment-6.3.py
Assignment-6.5.py
Assignment-7.5.py
Assignment-8.3.py
Assignment-9.3.py
exp-02_A.py
exp-02_B.py
Assignment-7.5.py
Assignment-8.3.py
.py
exp-02_B.py
exp-02_A.py
Assignment-9.3.py
#Generate a Python function to return the sum of even numbers and sum of odd numbers in a given list of inte
def sum_even_odd(numbers):
sum_even = sum(num for num in numbers if num % 2 == 0)
sum_odd = sum(num for num in numbers if num % 2 != 0)
return sum_even, sum_odd
# Example usage:
numbers = [4, 6, 9, 5, 3, 1]
even_sum, odd_sum = sum_even_odd(numbers)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/.py"
Sum of even numbers: 10
Sum of odd numbers: 18
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
CHAT
SESSIONS
Fibonacci sequence generation up to n ter...
Completed in 5s.
1 mo ago
Explore and understand your code
Ask
Auto
OUTLINE
TIMELINE
Ln 7, Col 28
Spaces: 4
UTF-8
CRLF
Python
3.13.12 (Microsoft Store)



OBSERVATION:

The function correctly calculates the sum of even and odd numbers in a clean and efficient way. Without documentation, the logic is understandable but not immediately clear to someone new reading the code. The AI-generated docstring improves clarity by clearly explaining the input, output, and purpose of the function. The added inline comments also make the step-by-step logic easier to follow. Overall, AI documentation enhances readability and professionalism, but it should always be reviewed to ensure it fully matches the code.

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

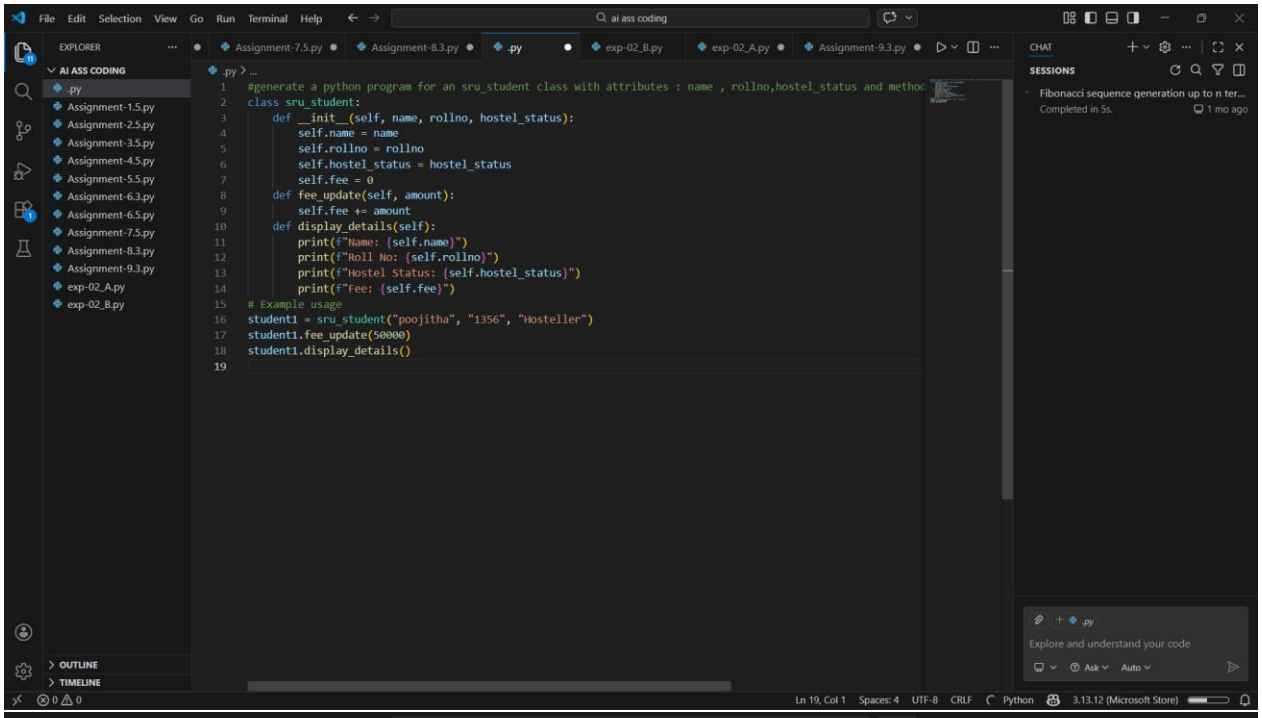
• Python class with manually written inline comments

• AI-generated inline comments added to the same code

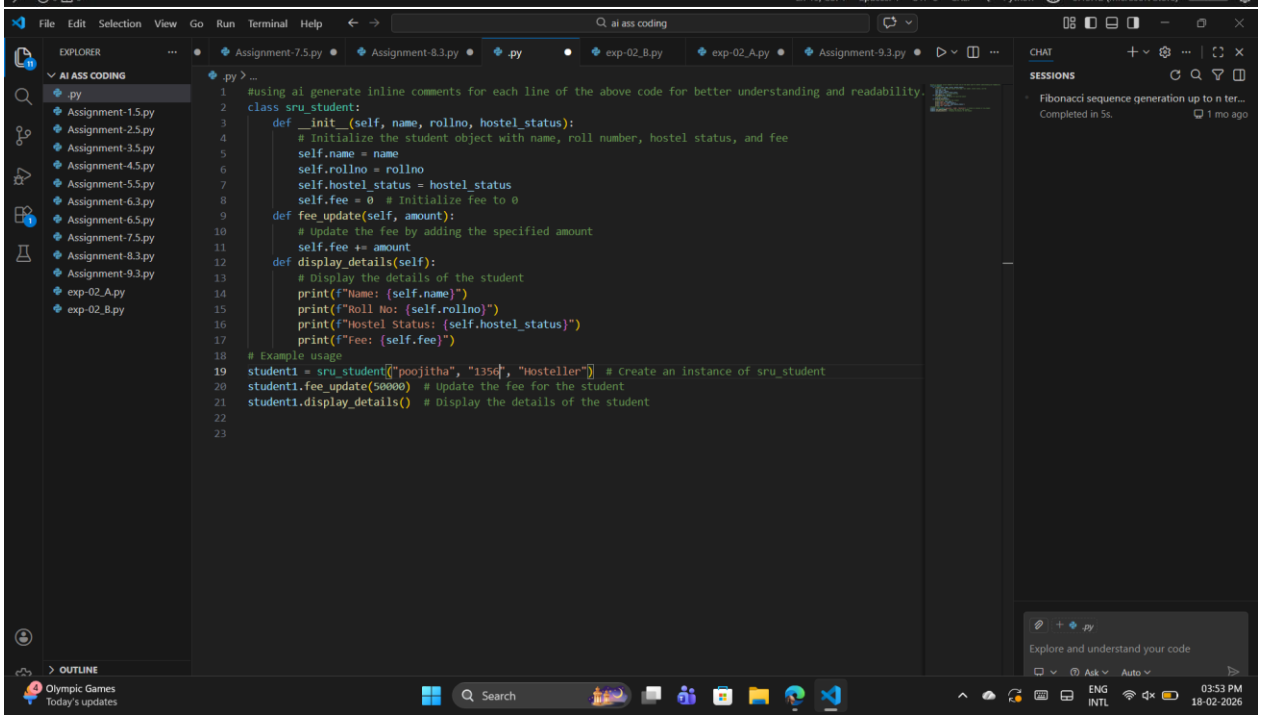
• Comparative analysis of manual vs AI comments

• Critical discussion on strengths and limitations of AI-generated comments

CODE:

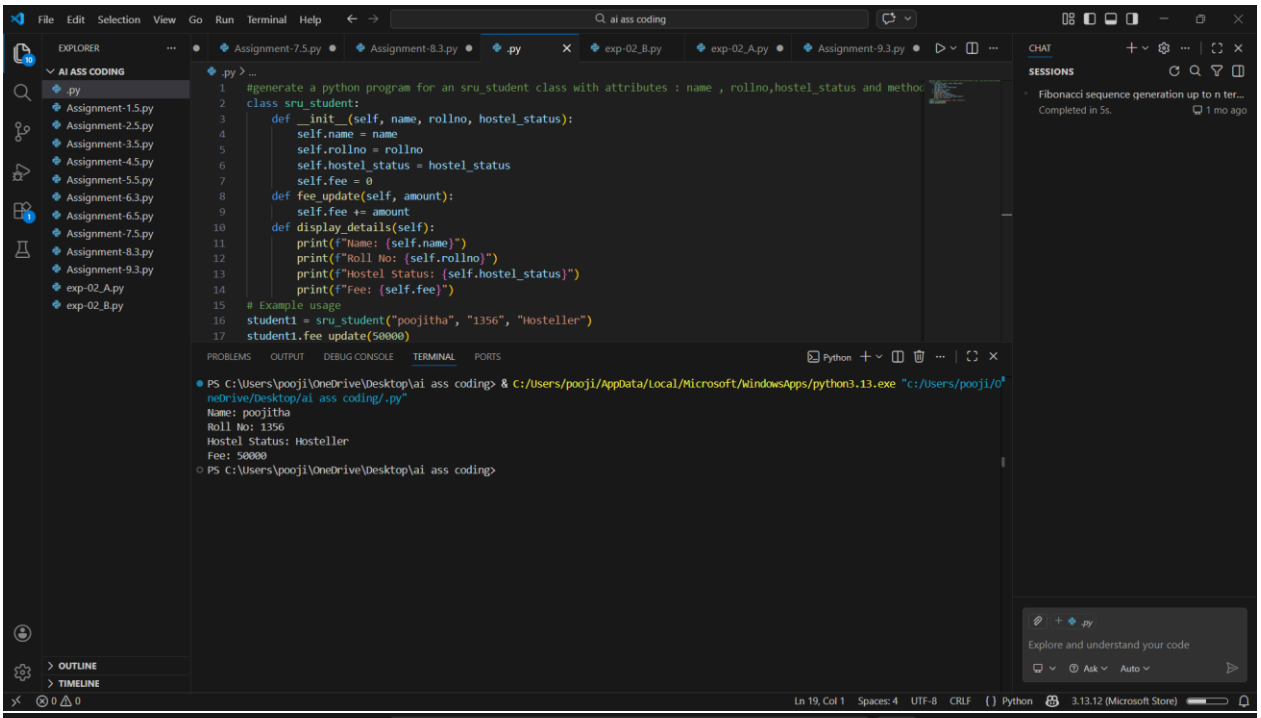


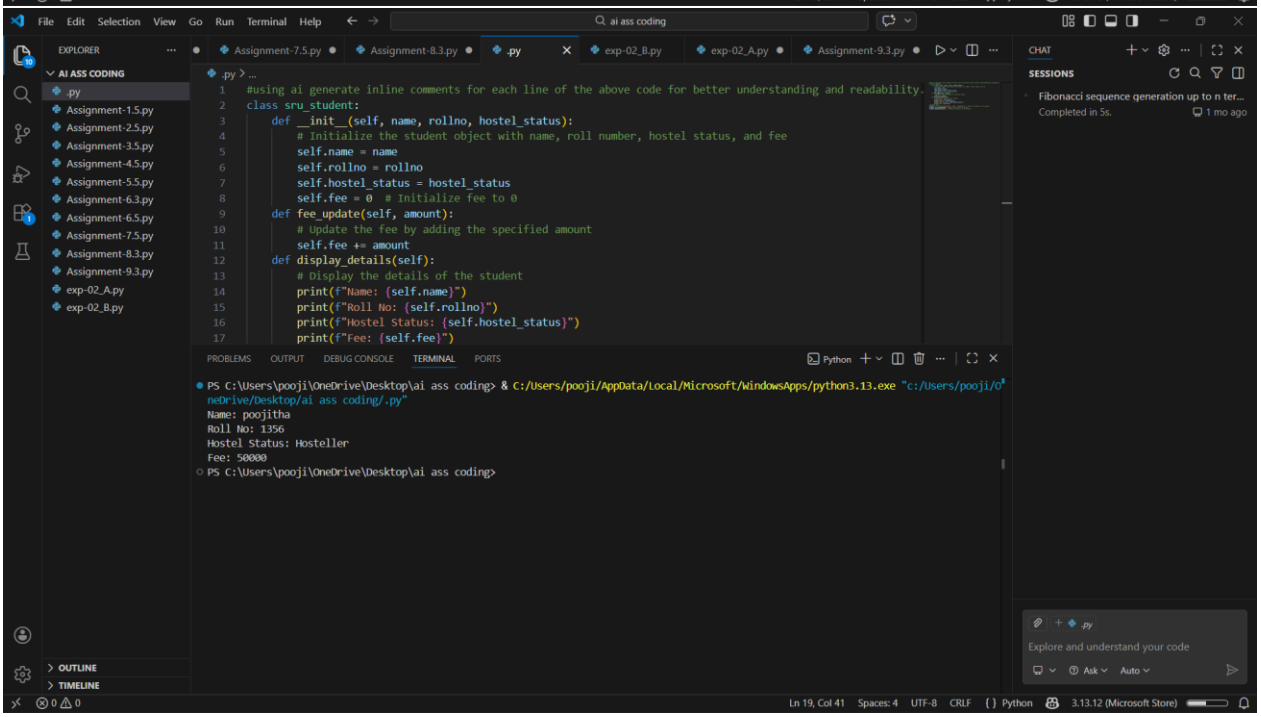
```python
#generate a python program for an sru_student class with attributes : name , rollno,hostel_status and methoc
class sru_student:
    def __init__(self, name, rollno, hostel_status):
        self.name = name
        self.rollno = rollno
        self.hostel_status = hostel_status
        self.fee = 0
    def fee_update(self, amount):
        self.fee += amount
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll No: {self.rollno}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee: {self.fee}")
# Example usage
student1 = sru_student("poojitha", "1356", "Hosteller")
student1.fee_update(50000)
student1.display_details()
```



```python
#using ai generate inline comments for each line of the above code for better understanding and readability.
class sru_student:
    def __init__(self, name, rollno, hostel_status):
        # Initialize the student object with name, roll number, hostel status, and fee
        self.name = name
        self.rollno = rollno
        self.hostel_status = hostel_status
        self.fee = 0  # Initialize fee to 0
    def fee_update(self, amount):
        # Update the fee by adding the specified amount
        self.fee += amount
    def display_details(self):
        # Display the details of the student
        print(f"Name: {self.name}")
        print(f"Roll No: {self.rollno}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee: {self.fee}")
# Example usage
student1 = sru_student("poojitha", "1356", "Hosteller")  # Create an instance of sru_student
student1.fee_update(50000)  # Update the fee for the student
student1.display_details()  # Display the details of the student
```

OUTPUT:





OBSERVATION :

The sru_student class is simple, organized, and easy to understand. The AI-generated inline comments explain the purpose of attributes and methods clearly, which makes the program more beginner-friendly. The comments help readers understand how the fee is updated and how details are displayed. However, some comments describe very obvious actions, which may not be necessary in professional code. Still, AI assistance improves clarity and makes the code easier to maintain.

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

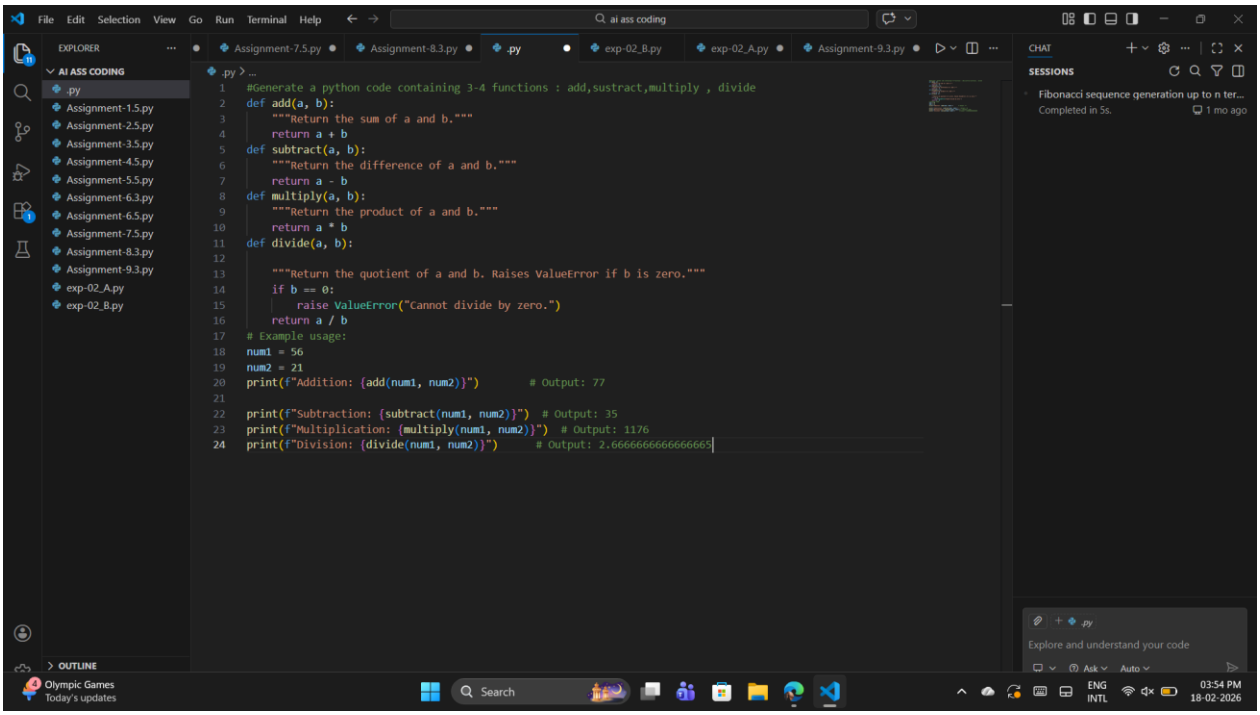
• Python script with manual NumPy-style docstrings

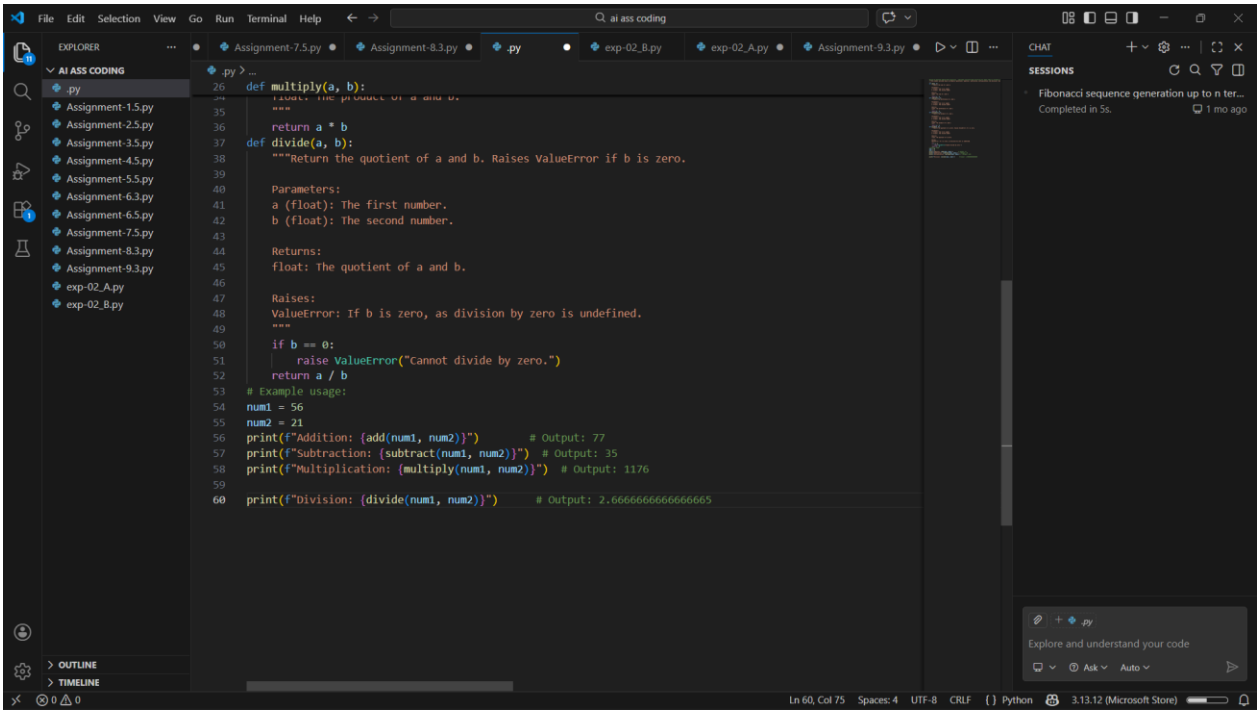
• AI-generated module-level and function-level documentation

• Comparison between AI-generated and manual documentation

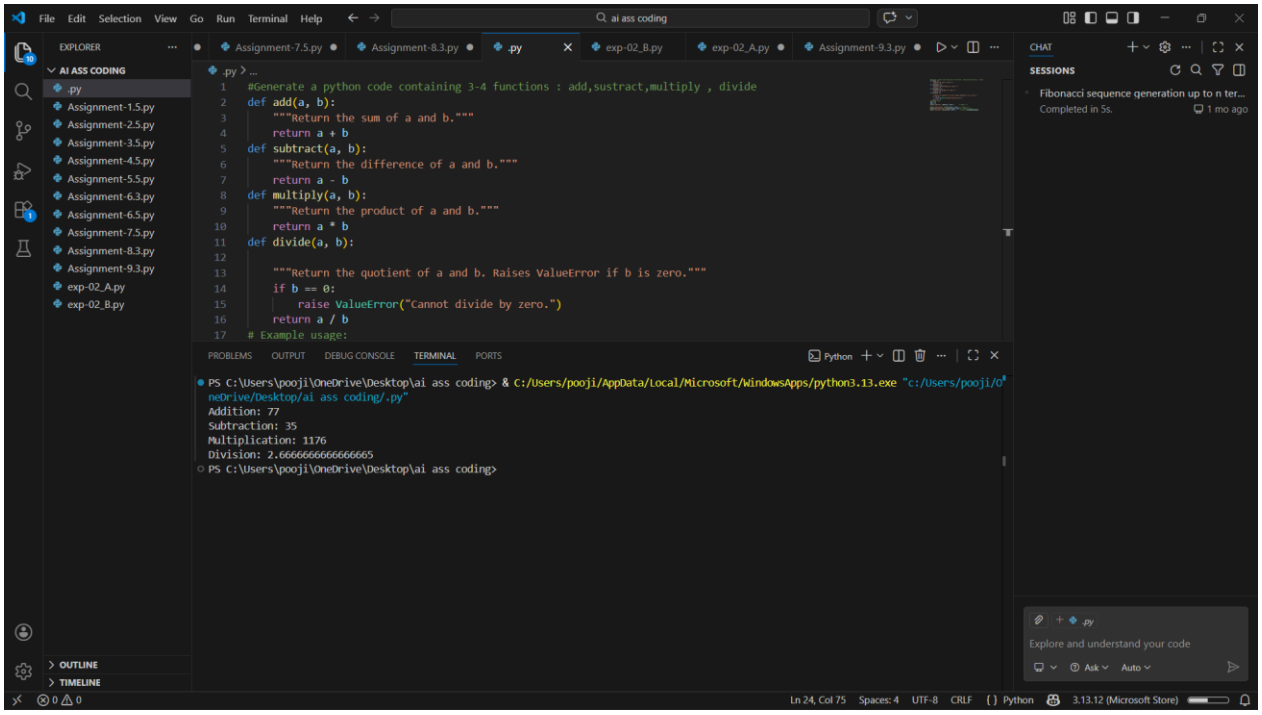
• Clear understanding of structured documentation for multi-function scripts

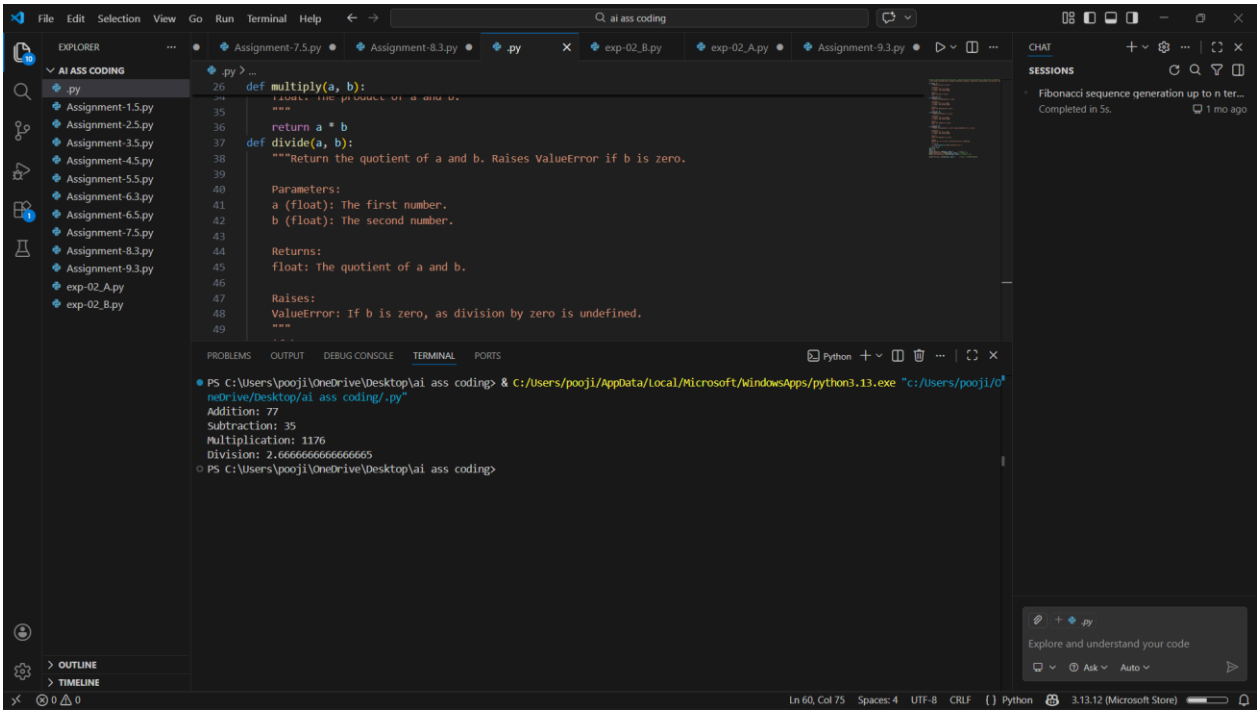
CODE:




File Edit Selection View Go Run Terminal Help
ai ass coding
EXPLORER
AI ASS CODING
.py
Assignment-1.5.py
Assignment-2.5.py
Assignment-3.5.py
Assignment-4.5.py
Assignment-5.5.py
Assignment-6.3.py
Assignment-6.5.py
Assignment-7.5.py
Assignment-8.3.py
Assignment-9.3.py
exp-02_A.py
exp-02_B.py
Assignment-7.5.py
Assignment-8.3.py
.py
exp-02_B.py
exp-02_A.py
Assignment-9.3.py
def multiply(a, b):
"""
return a * b
def divide(a, b):
"""Return the quotient of a and b. Raises ValueError if b is zero.
Parameters:
a (float): The first number.
b (float): The second number.
Returns:
float: The quotient of a and b.
Raises:
ValueError: If b is zero, as division by zero is undefined.
"""
if b == 0:
raise ValueError("Cannot divide by zero.")
return a / b
# Example usage:
num1 = 56
num2 = 21
print(f"Addition: {add(num1, num2)}") # Output: 77
print(f"Subtraction: {subtract(num1, num2)}") # Output: 35
print(f"Multiplication: {multiply(num1, num2)}") # Output: 1176
print(f"Division: {divide(num1, num2)}") # Output: 2.6666666666666665
CHAT
SESSIONS
Fibonacci sequence generation up to n ter...
Completed in 5s.
1 mo ago
Explore and understand your code
Ask
Auto
OUTLINE
TIMELINE
Ln 60, Col 75
Spaces: 4
UTF-8
CRLF
Python
3.13.12 (Microsoft Store)

OUTPUT:


File Edit Selection View Go Run Terminal Help
ai ass coding
EXPLORER
AI ASS CODING
.py
Assignment-1.5.py
Assignment-2.5.py
Assignment-3.5.py
Assignment-4.5.py
Assignment-5.5.py
Assignment-6.3.py
Assignment-6.5.py
Assignment-7.5.py
Assignment-8.3.py
Assignment-9.3.py
exp-02_A.py
exp-02_B.py
Assignment-7.5.py
Assignment-8.3.py
.py
exp-02_B.py
exp-02_A.py
Assignment-9.3.py
#Generate a python code containing 3-4 functions : add,sustract,multiply , divide
def add(a, b):
"""Return the sum of a and b."""
return a + b
def subtract(a, b):
"""Return the difference of a and b."""
return a - b
def multiply(a, b):
"""Return the product of a and b."""
return a * b
def divide(a, b):
"""Return the quotient of a and b. Raises ValueError if b is zero."""
if b == 0:
raise ValueError("Cannot divide by zero.")
return a / b
Example usage:
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/.py"
Addition: 77
Subtraction: 35
Multiplication: 1176
Division: 2.6666666666666665
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
CHAT
SESSIONS
Fibonacci sequence generation up to n ter...
Completed in 5s.
1 mo ago
Explore and understand your code
Ask
Auto
OUTLINE
TIMELINE
Ln 24, Col 75
Spaces: 4
UTF-8
CRLF
Python
3.13.12 (Microsoft Store)



OBSERVATION:

The calculator functions are logically correct and work as expected. In the original version, the documentation is brief and does not provide detailed information about parameters or exceptions. The AI-generated module-level and function-level docstrings add structure and clarity by explaining inputs, outputs, and error handling in a professional format. This makes the script more suitable for collaborative projects and real-world applications. Overall, AI improves documentation quality while keeping the program logic unchanged.