ASSIGNMENT 9.3

K.CHANDRIKAA

2403A51227

TASK 1:

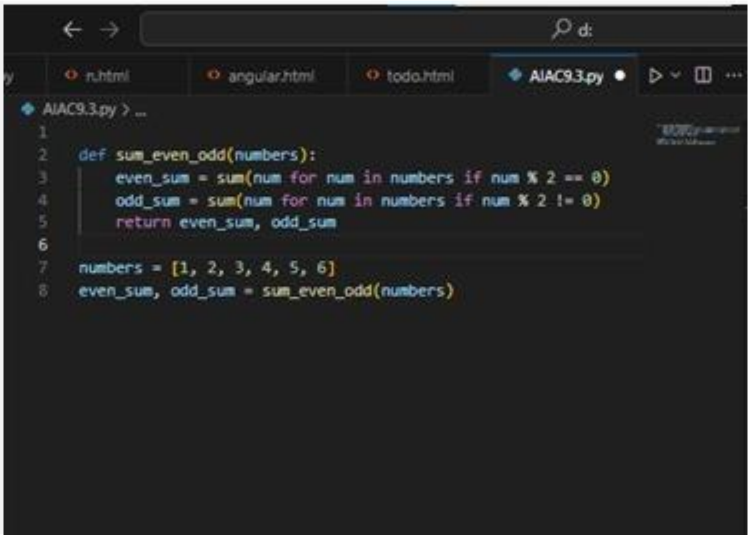
Task Description#1 Basic Docstring Generation

• Write python function to return sum of even and odd numbers in the given list.

• Incorporate manual docstring in code with Google Style

• Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.

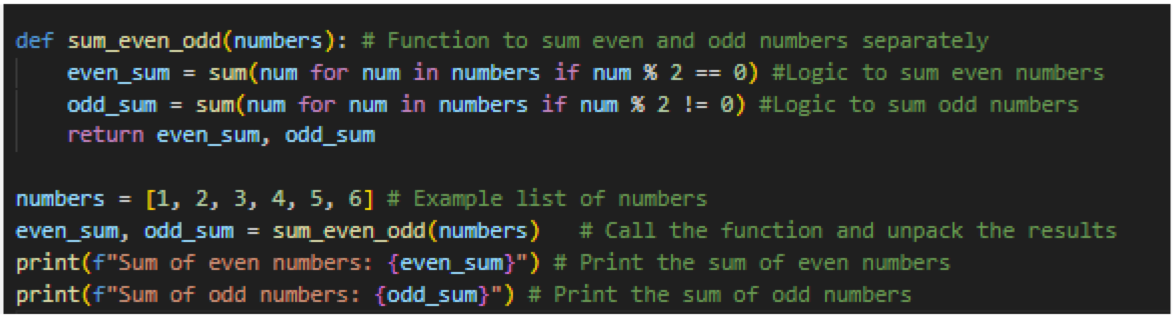
• Compare the AI-generated docstring with your manually written one.



```python

def sum_even_odd(numbers):
    even_sum = sum(num for num in numbers if num % 2 == 0)
    odd_sum = sum(num for num in numbers if num % 2 != 0)
    return even_sum, odd_sum

numbers = [1, 2, 3, 4, 5, 6]
even_sum, odd_sum = sum_even_odd(numbers)
```

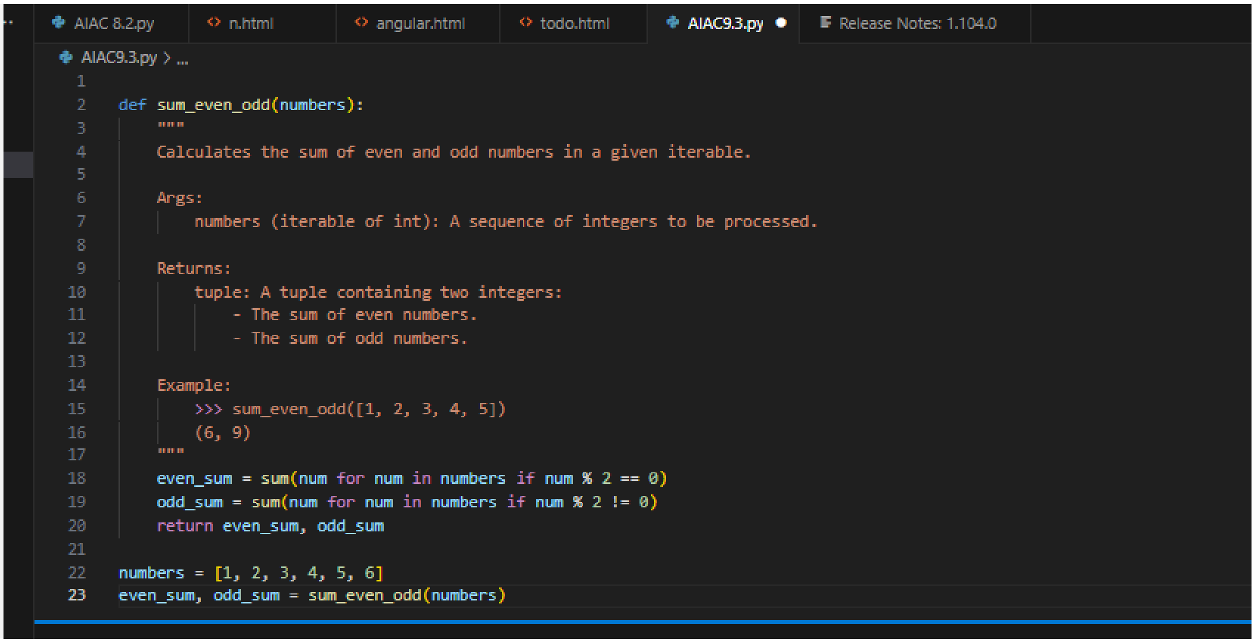
Docstrings:

1. **Manual docstrings:**



```python
def sum_even_odd(numbers): # Function to sum even and odd numbers separately
    even_sum = sum(num for num in numbers if num % 2 == 0) #Logic to sum even numbers
    odd_sum = sum(num for num in numbers if num % 2 != 0) #Logic to sum odd numbers
    return even_sum, odd_sum

numbers = [1, 2, 3, 4, 5, 6] # Example list of numbers
even_sum, odd_sum = sum_even_odd(numbers)   # Call the function and unpack the results
print(f"Sum of even numbers: {even_sum}") # Print the sum of even numbers
print(f"Sum of odd numbers: {odd_sum}") # Print the sum of odd numbers
```

2. AI GENERATED DOCSTRINGS:



```python

def sum_even_odd(numbers):
    """
    Calculates the sum of even and odd numbers in a given iterable.

    Args:
        numbers (iterable of int): A sequence of integers to be processed.

    Returns:
        tuple: A tuple containing two integers:
            - The sum of even numbers.
            - The sum of odd numbers.

    Example:
        >>> sum_even_odd([1, 2, 3, 4, 5])
        (6, 9)
    """
    even_sum = sum(num for num in numbers if num % 2 == 0)
    odd_sum = sum(num for num in numbers if num % 2 != 0)
    return even_sum, odd_sum

numbers = [1, 2, 3, 4, 5, 6]
even_sum, odd_sum = sum_even_odd(numbers)
```

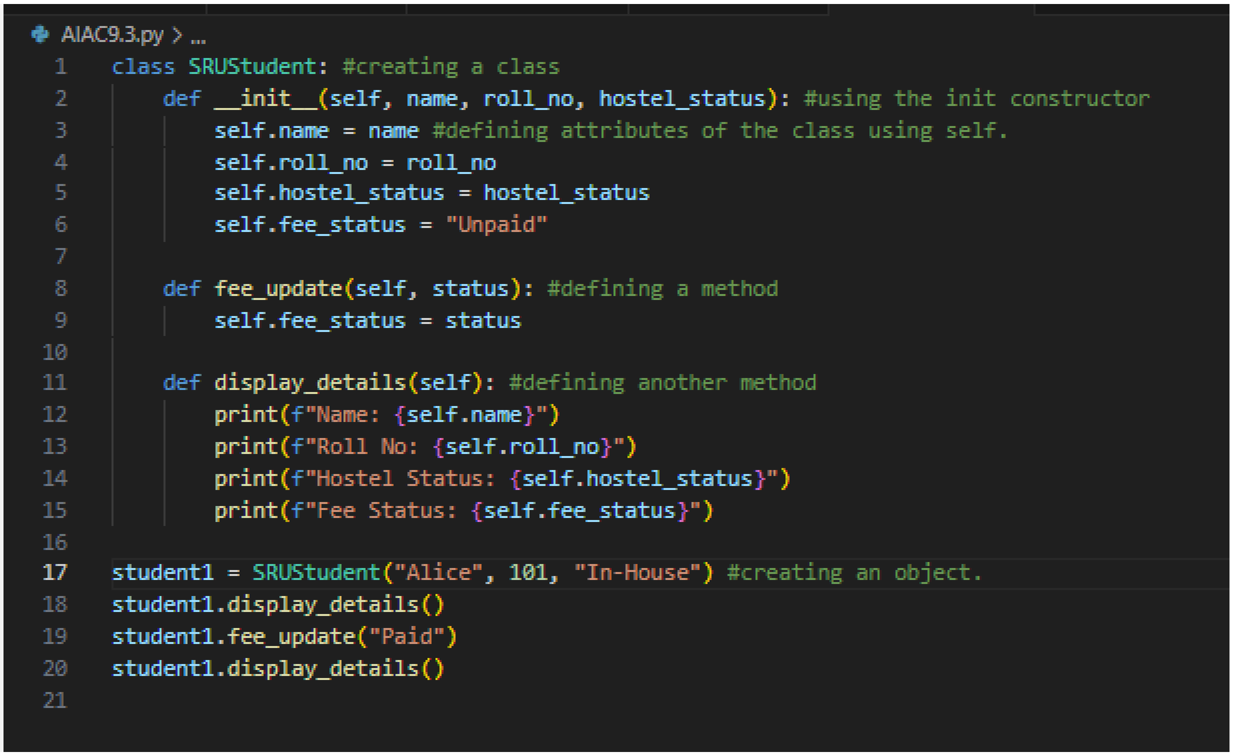
TASK 2:

Task Description#2 Automatic Inline Comments

- Write python program for sru_student class with attributes like name, roll no., hostel_status and fee_update method and display_details method.
- Write comments manually for each line/code block
- Ask an AI tool to add inline comments explaining each line/step.
- Compare the AI-generated comments with your manually written one

DOCSTRINGS:

1. MANUAL DOCSTRING

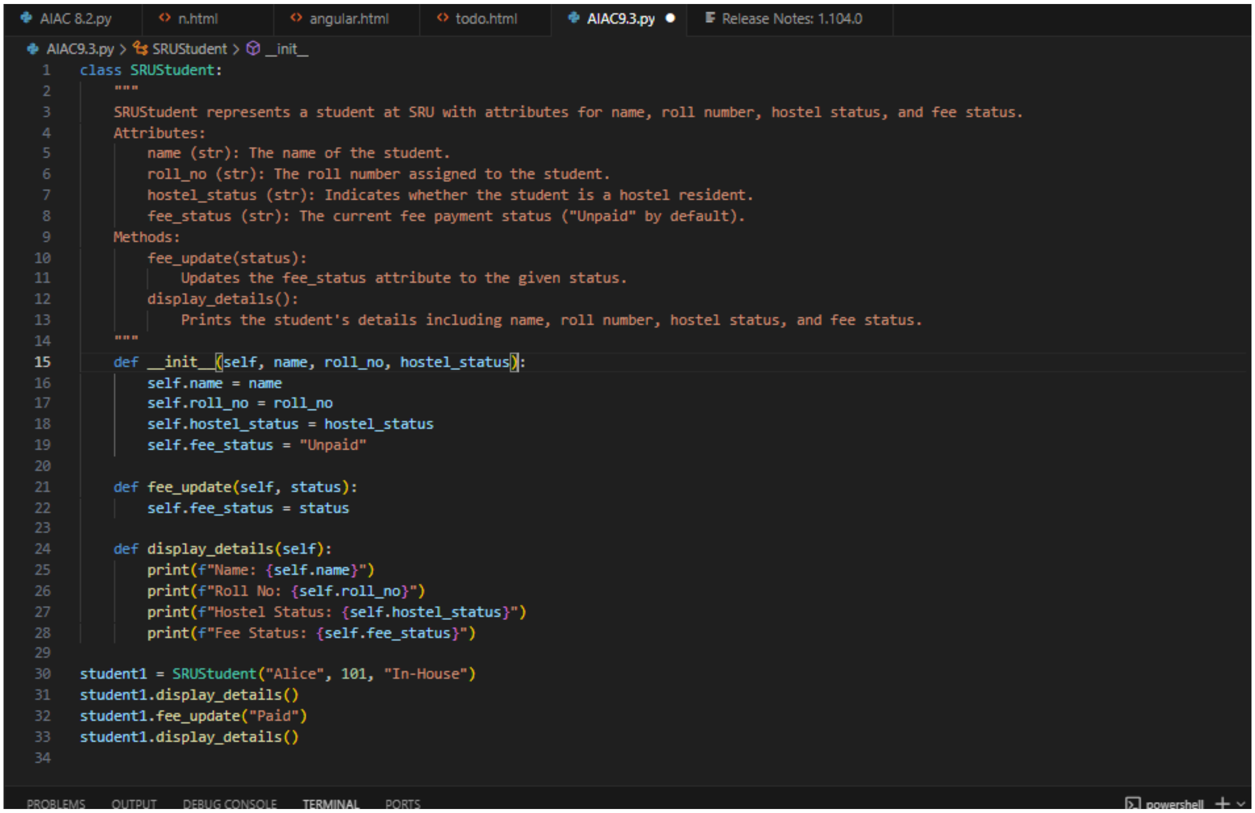


AIAC9.3.py > ...

```python
class SRUStudent: #creating a class
    def __init__(self, name, roll_no, hostel_status): #using the init constructor
        self.name = name #defining attributes of the class using self.
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_status = "Unpaid"

    def fee_update(self, status): #defining a method
        self.fee_status = status

    def display_details(self): #defining another method
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Status: {self.fee_status}")

student1 = SRUStudent("Alice", 101, "In-House") #creating an object.
student1.display_details()
student1.fee_update("Paid")
student1.display_details()
```

The above is manual docstring

2. AI GENERATED DOCSTRING:



```python
class SRUStudent:
    """
    SRUStudent represents a student at SRU with attributes for name, roll number, hostel status, and fee status.
    Attributes:
        name (str): The name of the student.
        roll_no (str): The roll number assigned to the student.
        hostel_status (str): Indicates whether the student is a hostel resident.
        fee_status (str): The current fee payment status ("Unpaid" by default).
    Methods:
        fee_update(status):
            Updates the fee_status attribute to the given status.
        display_details():
            Prints the student's details including name, roll number, hostel status, and fee status.
    """
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_status = "Unpaid"

    def fee_update(self, status):
        self.fee_status = status

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Status: {self.fee_status}")

student1 = SRUStudent("Alice", 101, "In-House")
student1.display_details()
student1.fee_update("Paid")
student1.display_details()
```

This is the AI GENERATED DOCSTRING.

TASK 3:

Task Description#3

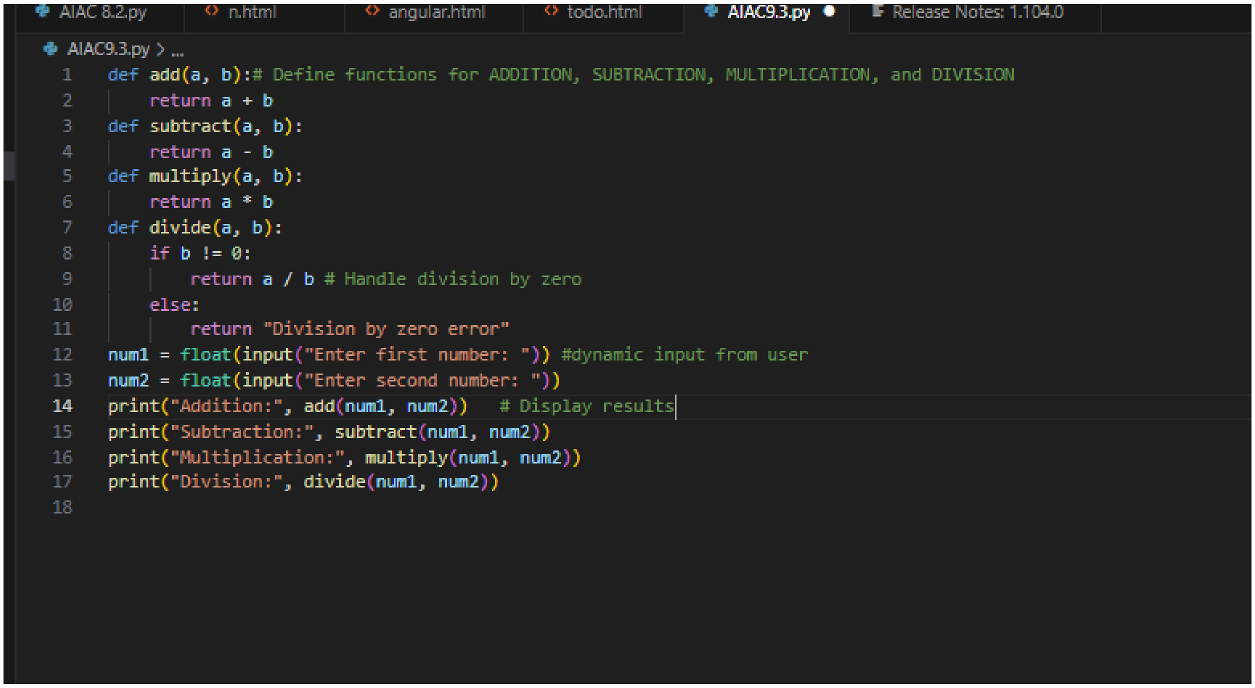
• Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).

• Incorporate manual docstring in code with NumPy Style

• Use AI assistance to generate a module-level docstring + individual function docstrings.

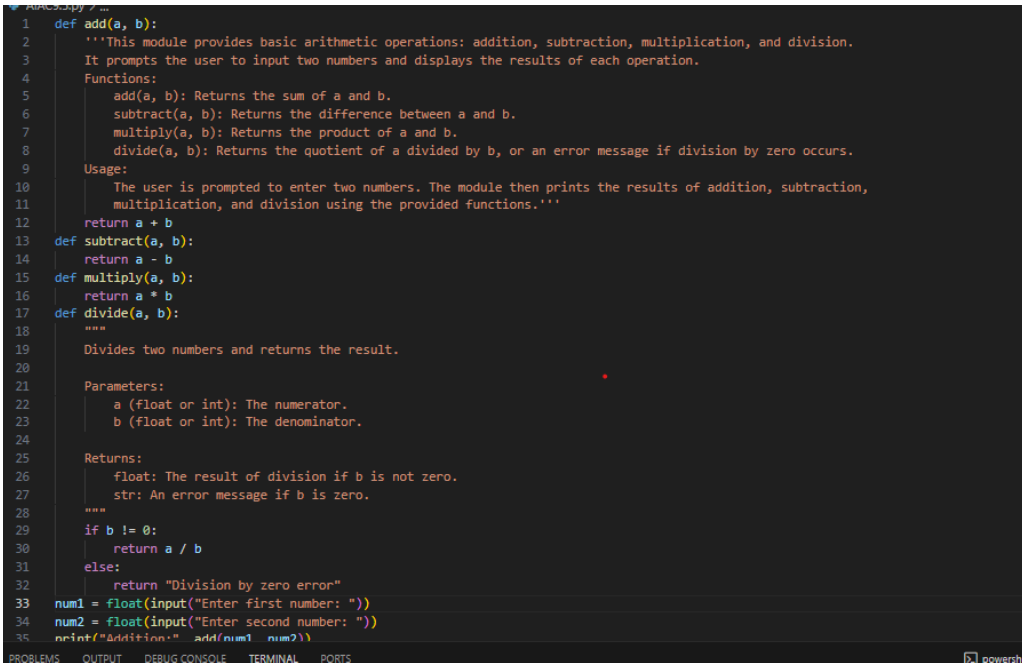
• Compare the AI-generated docstring with your manually written one

Manual docstrings:



```python
def add(a, b):# Define functions for ADDITION, SUBTRACTION, MULTIPLICATION, and DIVISION
    return a + b
def subtract(a, b):
    return a - b
def multiply(a, b):
    return a * b
def divide(a, b):
    if b != 0:
        return a / b # Handle division by zero
    else:
        return "Division by zero error"
num1 = float(input("Enter first number: ")) #dynamic input from user
num2 = float(input("Enter second number: "))
print("Addition:", add(num1, num2))    # Display results
print("Subtraction:", subtract(num1, num2))
print("Multiplication:", multiply(num1, num2))
print("Division:", divide(num1, num2))
```

AI GENERATED DOCSTINGS:



```
def add(a, b):
    '''This module provides basic arithmetic operations: addition, subtraction, multiplication, and division.
    It prompts the user to input two numbers and displays the results of each operation.
    Functions:
        add(a, b): Returns the sum of a and b.
        subtract(a, b): Returns the difference between a and b.
        multiply(a, b): Returns the product of a and b.
        divide(a, b): Returns the quotient of a divided by b, or an error message if division by zero occurs.
    Usage:
        The user is prompted to enter two numbers. The module then prints the results of addition, subtraction,
        multiplication, and division using the provided functions.'''
    return a + b
def subtract(a, b):
    return a - b
def multiply(a, b):
    return a * b
def divide(a, b):
    """
    Divides two numbers and returns the result.

    Parameters:
        a (float or int): The numerator.
        b (float or int): The denominator.

    Returns:
        float: The result of division if b is not zero.
        str: An error message if b is zero.
    """
    if b != 0:
        return a / b
    else:
        return "Division by zero error"
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
print("Addition:", add(num1, num2))
```