Assignment-9.3

Name: Y. Poojitha

Hall Ticket No:2303A51499

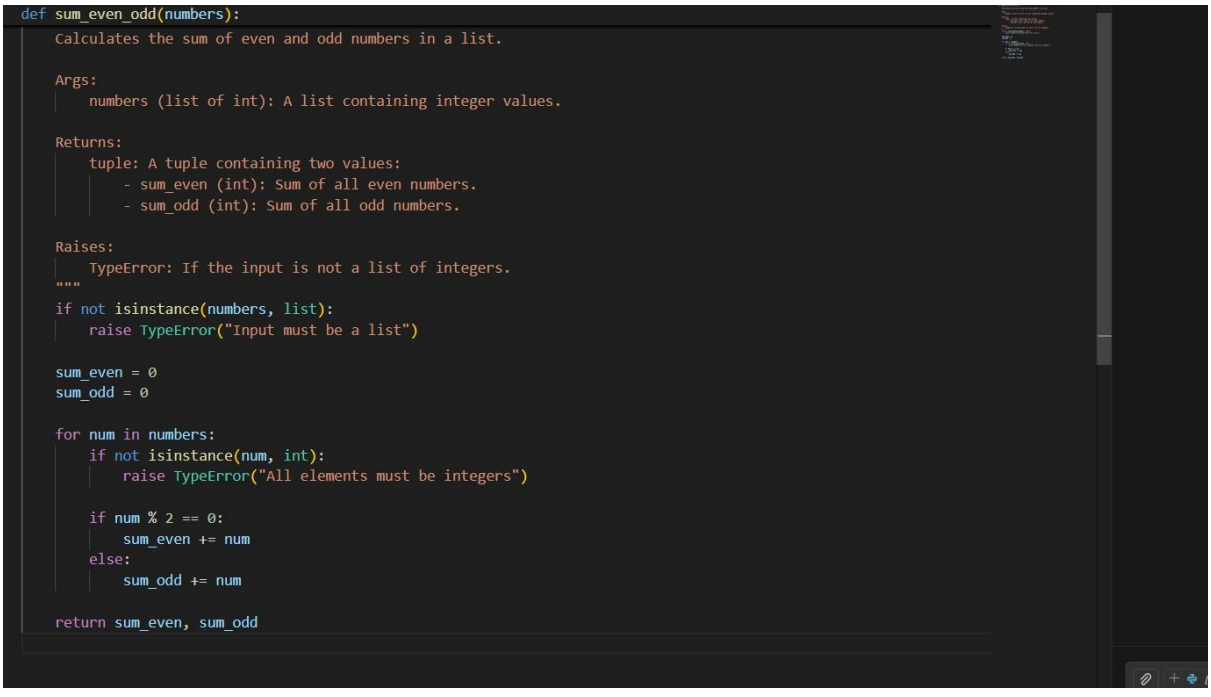
Batch:08

Lab 9: Documentation Generation – Automatic Documentation and Scenario

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

Task 1: Basic Docstring Generation

A. Manual Implementation



```python
def sum_even_odd(numbers):
    Calculates the sum of even and odd numbers in a list.

    Args:
        numbers (list of int): A list containing integer values.

    Returns:
        tuple: A tuple containing two values:
            - 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
```

B. Prompt Used for AI Generation

Generate a Google Style docstring for the following Python function.

Include Args, Returns, and Raises sections.

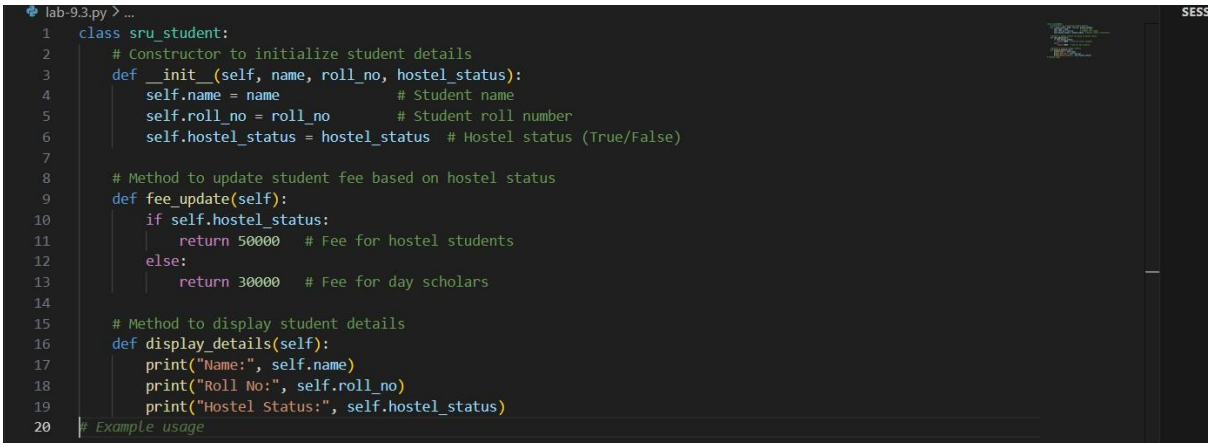
```python
def sum_even_odd(numbers):
    """
    Returns the sum of even and odd numbers from a list.

    Parameters:
        numbers (list): List of integers.

    Returns:
        tuple: (even_sum, odd_sum)
    """
    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)
```

Task 2: Automatic Inline Comments

Scenario

Develop a student management module that is easy for new developers to understand.

A. Manual Inline Comments



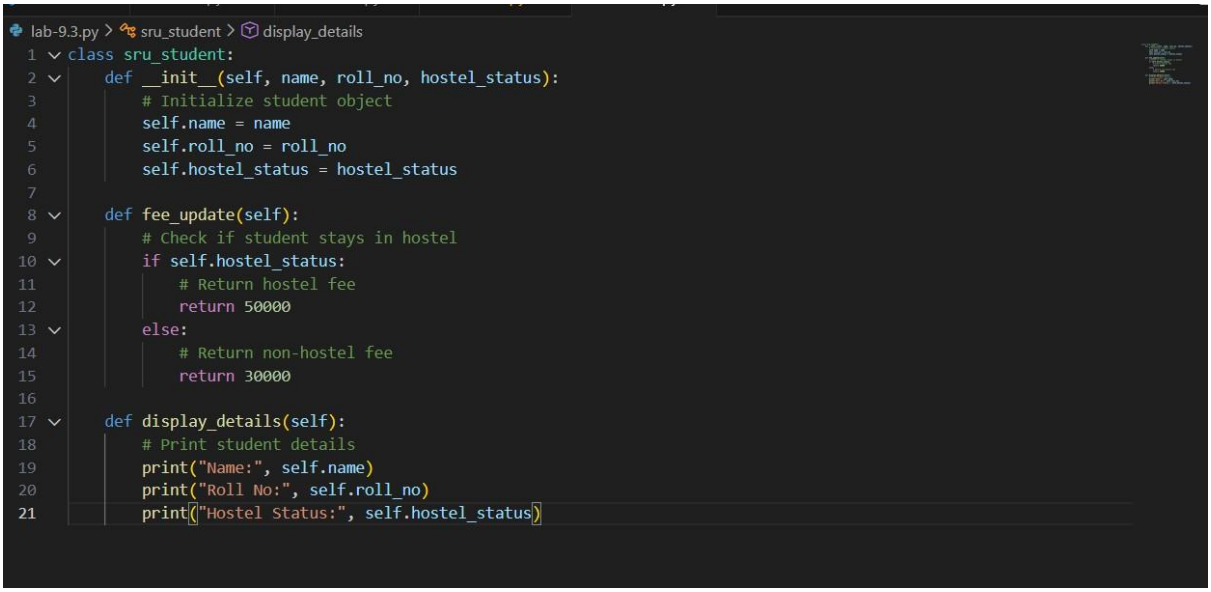
lab-9.3.py > ...

```python
class sru_student:
    # Constructor to initialize student details
    def __init__(self, name, roll_no, hostel_status):
        self.name = name              # Student name
        self.roll_no = roll_no        # Student roll number
        self.hostel_status = hostel_status  # Hostel status (True/False)

    # Method to update student fee based on hostel status
    def fee_update(self):
        if self.hostel_status:
            return 50000   # Fee for hostel students
        else:
            return 30000   # Fee for day scholars

    # Method to display student details
    def display_details(self):
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
# Example usage
```

B. Prompt Used:

Add meaningful inline comments to the following Python class.

Explain each logical block clearly.

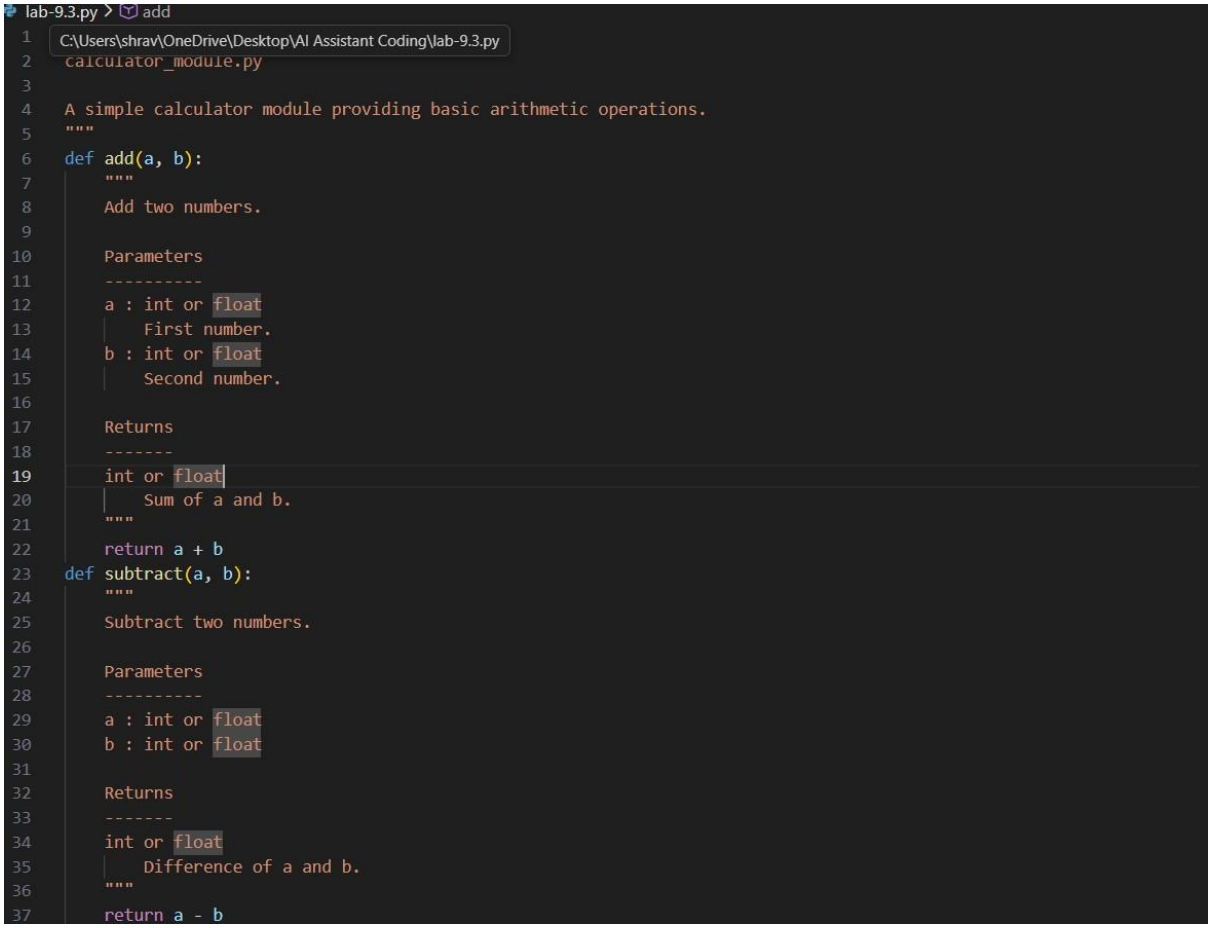


Task 3: Module-Level & Function-Level Documentation

Scenario

Develop a reusable calculator module with structured documentation.

A. Manual NumPy Style Documentation


lab-9.3.py > add
C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding\lab-9.3.py
calculator_module.py
A simple calculator module providing basic arithmetic operations.
"""
def add(a, b):
"""
Add two numbers.
Parameters
----
a : int or float
First number.
b : int or float
Second number.
Returns
----
int or float
Sum of a and b.
"""
return a + b
def subtract(a, b):
"""
Subtract two numbers.
Parameters
----
a : int or float
b : int or float
Returns
----
int or float
Difference of a and b.
"""
return a - b

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

    Parameters
    ----------
    a : int or float
    b : int or float

    Returns
    -------
    int or float
        Product of a and b.
    """
    return a * b
def divide(a, b):
    """
    Divide two numbers.

    Parameters
    ----------
    a : int or float
    b : int or float

    Returns
    -------
    float
        Result of division.

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

B.Prompt Used:

Generate NumPy Style docstrings and a professional modulelevel docstring for this calculator module.

```python
"""
This module provides basic arithmetic operations such as addition,
subtraction, multiplication, and division. It can be reused in
multiple projects requiring simple mathematical calculations.
"""
def add(a, b):
    """Return the sum of a and b."""
    return a + b
```

Conclusion:

This lab demonstrated the importance of structured documentation in software development.

Key Learnings:

- AI-assisted tools help generate documentation quickly.
- AI-generated comments are useful but require human review.
- Manual documentation ensures completeness and accuracy.
- Google and NumPy styles improve readability and maintainability.
- AI is a productivity tool, not a replacement for developer responsibility.

Overall, combining manual expertise with AI assistance produces the best quality documentation.