ASSIGNMENT-9.3

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

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

    Args:
        numbers (list[int]): A list of integers.

    Returns:
        tuple[int, int]: A tuple containing:
            - Sum of even numbers
            - Sum of odd numbers

    Raises:
        TypeError: If the input is not a list or contains non-integer values.

    Example:
        >>> sum_even_odd([1, 2, 3, 4])
        (6, 4)
    """
    if not isinstance(numbers, list):
        raise TypeError("Input must be a list of integers.")

    even_sum = 0
```

```
    odd_sum = 0

    for num in numbers:
        if not isinstance(num, int):
            raise TypeError("All elements must be integers.")
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum
# Example usage
numbers = [1, 2, 3, 4, 5, 6]
even_sum, odd_sum = sum_even_odd(numbers)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

#OUTPUT:

Sum of even numbers: 12

Sum of odd numbers: 9

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

```python
class SRUStudent:
    # Constructor to initialize student attributes
    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  # True if staying in hostel, else
False
        self.fee = 0                        # Initial fee amount

    # Method to update fee
    def fee_update(self, amount):
        # Add academic fee
        self.fee += amount

        # If hostel student, add fixed hostel charge
        if self.hostel_status:
            self.fee += 5000

    # Method to display student details
    def display_details(self):
        print("\n--- Student Details ---")
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", "Yes" if self.hostel_status else "No")
        print("Total Fee:", self.fee)


# -------- Main Program --------
if __name__ == "__main__":
    # Taking user input
    name = input("Enter student name: ")
    roll_no = input("Enter roll number: ")

    hostel_input = input("Is student staying in hostel? (yes/no): ").lower()
    hostel_status = True if hostel_input == "yes" else False

    fee_amount = float(input("Enter academic fee amount: "))

    # Creating student object
    student1 = SRUStudent(name, roll_no, hostel_status)

    # Updating fee
    student1.fee_update(fee_amount)

    # Displaying details
    student1.display_details()
```

#OUTPUT:

Enter student name: Ramana

Enter roll number: 1170

Is student staying in hostel? (yes/no): no

Enter academic fee amount: 17500

--- Student Details ---

Name: Ramana

Roll No: 1170

Hostel Status: No

Total Fee: 17500.0

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

```
"""
calculator.py

A simple calculator module providing basic arithmetic operations.
This module is designed to be reusable across multiple projects.
"""
```

```python
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.

    Examples
    --------
    >>> add(2, 3)
    5
    """
    return a + b


def subtract(a, b):
    """
    Subtract two numbers.

    Parameters
    ----------
    a : int or float
        First number.
    b : int or float
        Second number.

    Returns
    -------
    int or float
        Result of a minus b.

    Examples
    --------
    >>> subtract(5, 3)
    2
    """
    return a - b
```

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

    Parameters
    ----------
    a : int or float
        First number.
    b : int or float
        Second number.

    Returns
    -------
    int or float
        Product of a and b.

    Examples
    --------
    >>> multiply(4, 3)
    12
    """
    return a * b


def divide(a, b):
    """
    Divide two numbers.

    Parameters
    ----------
    a : int or float
        Numerator.
    b : int or float
        Denominator.

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

    Raises
    ------
    ZeroDivisionError
        If b is zero.

    Examples
    --------
    >>> divide(10, 2)
    5.0
```

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


# Optional test block
if __name__ == "__main__":
    print("Add:", add(5, 3))
    print("Subtract:", subtract(5, 3))
    print("Multiply:", multiply(5, 3))
    print("Divide:", divide(10, 2))
```

#OUTPUT:

Add: 8

Subtract: 2

Multiply: 15

Divide: 5.0