

## ASSIGNMENT 9.3

NAME: B.Manishwar

HALL TICKET: 2303A51276

BATCH: 05

### 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

##### PROMPT:

Write a Python function that takes a list of integers and returns the sum of even numbers and the sum of odd numbers.

### INPUT and OUTPUT

```
[15] def sum_even_odd(numbers):
✓ On  """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 integers:
            - The sum of even numbers.
            - The sum of odd numbers.

    Example:
        >>> sum_even_odd([1, 2, 3, 4, 5])
        (6, 9)
    """
    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

# Example usage
if __name__ == "__main__":
    nums = [1, 2, 3, 4, 5]
    even, odd = sum_even_odd(nums)
```

```
[15] print(f"Sum of even numbers: {even}")
✓ 0s print(f"Sum of odd numbers: {odd}")

# --- AI-style docstring for comparison ---

def sum_even_odd(numbers):
    """
    Computes the total sum of even and odd integers from the input list.

    Parameters:
        numbers (list[int]): List of integers to process.

    Returns:
        tuple[int, int]: First element is the sum of even numbers,
                        second element is the sum of odd numbers.

    Example:
        sum_even_odd([1, 2, 3, 4, 5]) -> (6, 9)
    """
    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

Sum of even numbers: 6
Sum of odd numbers: 9
```

## TASK 2:

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

### Prompt:

Create a Python class `sru_student` with manual and AI-generated inline comments and include a comparison for lab submission

### INPUT and OUTPUT:

```

class sru_student:
    # Constructor method for the sru_student class
    def __init__(self, name, roll_no, hostel_status):
        # Assign the provided name to the instance's name attribute
        self.name = name
        # Assign the provided roll_no to the instance's roll_no attribute
        self.roll_no = roll_no
        # Assign the provided hostel_status to the instance's hostel_status attribute
        self.hostel_status = hostel_status

    # Method to update the student's hostel status
    def fee_update(self, new_hostel_status):
        # Update the hostel_status attribute with the new value
        self.hostel_status = new_hostel_status
        # Print a confirmation message indicating the update
        print(f"Hostel status for {self.name} (Roll No: {self.roll_no}) updated to: {self.hostel_status}")

    # Method to display the student's details
    def display_details(self):
        # Print a header for student details
        print("\nStudent Details:")
        # Print the student's name
        print(f"Name: {self.name}")
        # Print the student's roll number
        print(f"Roll No: {self.roll_no}")
        # Print the student's hostel status
        print(f"Hostel Status: {self.hostel_status}")

# Create an instance of the sru_student class
student1 = sru_student("MANISHWAR", "2303A51276", "In-Hostel")

# Display initial details of the student
student1.display_details()

# Update the hostel status of the student
student1.fee_update("HOSTELER")

# Display details after the hostel status update
student1.display_details()

```

```

...
Student Details:
Name: MANISHWAR
Roll No: 2303A51276
Hostel Status: In-Hostel
Hostel status for MANISHWAR (Roll No: 2303A51276) updated to: HOSTELER

Student Details:
Name: MANISHWAR
Roll No: 2303A51276
Hostel Status: HOSTELER

```

### TASKE 3:

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

### Prompt:

Generate a Python calculator program with NumPy-style docstrings,  
AI-generated docstrings, and a comparison for lab submission

```

j)
Os
calculator.py

"""
A simple calculator module that provides basic arithmetic operations.
This module can be reused in different projects.
"""

def add(a, b):
    """
    Add two numbers.

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

    Returns
    -----
    float
        Sum of a and b
    """
    return a + b

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

```

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

    Returns
    -----
    float
        Difference of a and b
    """
    return a - b

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

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

    Returns
    -----
    float
        Product of a and b
  
```

```

        return a * b

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

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

    Returns
    -----
    float
        Division result

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

# Testing the functions
print("Add:", add(10, 5))
print("Subtract:", subtract(10, 5))

```

```

    """
    """

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

# Testing the functions
print("Add:", add(10, 5))
print("Subtract:", subtract(10, 5))
print("Multiply:", multiply(10, 5))
print("Divide:", divide(10, 5))

Add: 15
Subtract: 5
Multiply: 50
Divide: 2.0

```