

AI- Assignment-9.3

Name:G.Harika

Ht.No:2303A51612

Batch:05

Prompt: #You are developing a utility function that processes numerical lists and must be properly

#documented for future maintenance.

#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

Code: def sum_even_odd(numbers):

```
    """Calculate the sum of even and odd numbers in a list.
```

This function takes a list of integers and returns a tuple containing
the sum of even numbers and the sum of odd numbers.

Args:

numbers (list of int): A list of integers to be processed.

Returns:

tuple: A tuple containing two integers: (sum_of_evens, sum_of_odds)

....

```
    sum_of_evens = sum(num for num in numbers if num % 2 == 0)
```

```
    sum_of_odds = sum(num for num in numbers if num % 2 != 0)
```

```
return sum_of_evens, sum_of_odds
```

The screenshot shows a code editor window with a dark theme. On the left is a file tree with several Python files like 'calculator.py', 'fibonacci.py', 'AI-assist.py', and 'AI-assist-test.py'. The main pane displays a Python script with AI-generated inline comments. The comments provide instructions for developing a function that processes a list of integers to calculate the sum of even and odd numbers.

```
1 # You are developing a utility function that processes numerical lists and must be property
2 # Documenting the future existence
3 # Write a Python function to return the sum of even numbers and sum of odd numbers in a
4 # Given list
5 # Manually add a single docstring to the function
6 # Use an AI-assisted tool (assistant, browser, AI) to generate a function-level docstring
7 # Compare the AI-generated docstring with the manually added docstring
8 # Analyze Clarity, correctness, and completeness
9
10 def sum_even_odd(numbers):
11     """Calculate the sum of even and odd numbers in a list.
12
13     This function takes a list of integers and returns a tuple containing
14     the sum of even numbers and the sum of odd numbers.
15
16     Args:
17         numbers (list): A list of integers to be processed.
18
19     Returns:
20         tuple: A tuple containing two integers (sum_of_evens, sum_of_odds)
21
22     Examples:
23         sum_of_evens = calculate_sum([1, 2, 3, 4, 5, 6])
24         sum_of_odds = calculate_sum([1, 3, 5, 7, 9])
25
26         assert sum_of_evens == 12
27         assert sum_of_odds == 25
28
29     """
30
31     sum_evens = 0
32     sum_odds = 0
33
34     for number in numbers:
35         if number % 2 == 0:
36             sum_evens += number
37         else:
38             sum_odds += number
39
40     return sum_evens, sum_odds
```

Prompt: Task 2: Automatic Inline Comments

#You are developing a student management module that must be easy to understand for new developers.

#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: class sru_student:

```
# Constructor to initialize student attributes
def __init__(self, name, roll_no, hostel_status):
```

```
self.name = name      # Store student name
self.roll_no = roll_no    # Store student roll number
self.hostel_status = hostel_status # Store hostel status (True/False)

# Method to update fee based on hostel status
def fee_update(self):
    if self.hostel_status:    # If student stays in hostel
        fee = 50000          # Hostel students pay higher fee
    else:
        fee = 30000          # Non-hostel students pay lower fee
    return fee             # Return calculated fee

# Method to display student details
def display_details(self):
    print("Name:", self.name)  # Print student name
    print("Roll No:", self.roll_no) # Print student roll number
    print("Hostel Status:", "Yes" if self.hostel_status else "No") # Print hostel status
    print("Fee:", self.fee_update()) # Print fee based on hostel status

# Example usage
student1 = sru_student("Shivani", "SRU123", True)
student1.display_details()
```

```
File Edit Selection View Doc Run Terminal Help
File New Open Recent File Recent
calculator.py
# This module provides basic arithmetic operations.
# It includes functions for addition, subtraction, multiplication, and division.
# The module uses NumPy arrays for calculations.

import numpy as np
from typing import Union, Tuple

def add(x: Union[int, float], y: Union[int, float]) -> Union[int, float]:
    """Add two numbers.
    Parameters:
    x (int or float): First number
    y (int or float): Second number
    Returns:
    int or float: Sum of x and y
    """
    return x + y

def subtract(x: Union[int, float], y: Union[int, float]) -> Union[int, float]:
    """Subtract y from x.
    Parameters:
    x (int or float): Minuend
    y (int or float): Subtrahend
    Returns:
    int or float: Difference between x and y
    """
    return x - y

def multiply(x: Union[int, float], y: Union[int, float]) -> Union[int, float]:
    """Multiply x by y.
    Parameters:
    x (int or float): Factor
    y (int or float): Multiplier
    Returns:
    int or float: Product of x and y
    """
    return x * y

def divide(x: Union[int, float], y: Union[int, float]) -> Union[int, float]:
    """Divide x by y.
    Parameters:
    x (int or float): Dividend
    y (int or float): Divisor
    Returns:
    int or float: Quotient of x divided by y
    """
    if y == 0:
        raise ValueError("Division by zero is not allowed")
    return x / y
```

Prompt: Task 3: Module-Level and Function-Level Documentation

#You are building a small calculator module that will be shared across multiple projects and requires structured documentation.

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

Code: "'''

calculator.py

=====

A simple calculator module providing basic arithmetic operations.

This module can be reused across multiple projects where basic mathematical functions are required.

'''

```
def add(a, b):
```

```
    """
```

```
    Add two numbers.
```

Parameters

```
-----
```

```
a : float or int
```

```
    First number.
```

```
b : float or int
```

```
    Second number.
```

Returns

```
-----
```

```
float or int
```

```
The sum of `a` and `b`.
```

Examples

```
-----
```

```
>>> add(2, 3)
```

```
5
```

```
"""
```

```
return a + b
```

```
def subtract(a, b):
```

```
    """
```

```
    Subtract one number from another.
```

Parameters

a : float or int

First number.

b : float or int

Second number to subtract from `a`.

Returns

float or int

The result of `a - b`.

Examples

```
>>> subtract(5, 2)
```

```
3
```

```
....
```

```
return a - b
```

```
def multiply(a, b):
```

```
....
```

Multiply two numbers.

Parameters

a : float or int

First number.

b : float or int

Second number.

Returns

float or int

The product of `a` and `b`.

Examples

```
>>> multiply(4, 3)
```

```
12
```

```
return a * b
```

```
def divide(a, b):
```

Divide one number by another.

Parameters

a : float or int

Numerator.

b : float or int

Denominator.

Returns

float

The result of `a / b`.

Raises

ZeroDivisionError

If `b` is zero.

Examples

>>> divide(10, 2)

5.0

....

if b == 0:

 raise ZeroDivisionError("Division by zero is not allowed.")

return a / b

A screenshot of a Python code editor showing a file named `calculate.py`. The code defines two functions: `divide(a, b)` and `subtract(a, b)`.

```
divide(a, b)
    """ Divide two numbers
    Parameters:
        a : float or int
        b : float or int
    Returns:
        float or int
        The result of 'a' and 'b'.
    """
    return a / b

subtract(a, b)
    """ Subtract one number from another
    """
    return a - b
```