

AI Assisted Coding

Assignment – 9.3

M.Sathwik || 2303A51483 || Batch:- 08

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

```
Lab-9.py > sum_even_odd
1 #Create a Python function that returns the sum of even and odd numbers from a list.
2 # Add a manual Google-style docstring, then generate an AI docstring version.
3 # Compare both and analyze clarity, correctness, and completeness.
4 def sum_even_odd(numbers):
5     """
6     This function takes a list of numbers as input and returns a tuple containing the sum of even numbers and the sum of odd numbers.
7     Parameters:
8     numbers (list): A list of integers or floats.
9     Returns:
10    tuple: A tuple where the first element is the sum of even numbers and the second element is the sum of odd numbers.
11    Example:
12    sum_even_odd([1, 2, 3, 4, 5]) → (6, 9)
13    sum_even_odd([]) → (0, 0)
14    sum_even_odd([-1, -2, -3]) → (-2, -4)
15    """
16    even_sum = 0
17    odd_sum = 0
18    for num in numbers:
19        if isinstance(num, (int, float)): # Check if the element is a number
20            if num % 2 == 0:
21                even_sum += num
22            else:
23                odd_sum += num
24    return even_sum, odd_sum
25 # Test cases
26 print(sum_even_odd([1, 2, 3, 4, 5])) # Expected output: (6, 9)
27 print(sum_even_odd([])) # Expected output: (0, 0)
28 print(sum_even_odd([-1, -2, -3])) # Expected output: (-2, -4)
29 # AI-generated docstring version
30 def sum_even_odd(numbers):
31     """
32     Calculate the sum of even and odd numbers in a given list.
33     Args:
34     numbers (list): A list of integers or floats to be processed.
```

```

32     Calculate the sum of even and odd numbers in a given list.
33
34     Args:
35     numbers (list): A list of integers or floats to be processed.
36
37     Returns:
38     tuple: A tuple containing two elements: the first is the sum of even numbers, and the second is the sum of odd numbers.
39
40     Examples:
41     >>> sum_even_odd([1, 2, 3, 4, 5])
42     (6, 9)
43     >>> sum_even_odd([])
44     (0, 0)
45     >>> sum_even_odd([-1, -2, -3])
46     (-2, -4)
47     """
48
49     even_sum = 0
50     odd_sum = 0
51     for num in numbers:
52         if isinstance(num, (int, float)): # Check if the element is a number
53             if num % 2 == 0:
54                 even_sum += num
55             else:
56                 odd_sum += num
57     return even_sum, odd_sum
58
59 # Test cases
60 print(sum_even_odd([1, 2, 3, 4, 5])) # Expected output: (6, 9)
61 print(sum_even_odd([])) # Expected output: (0, 0)
62 print(sum_even_odd([-1, -2, -3])) # Expected output: (-2, -4)
63
64 # Analysis:
65 # Both docstrings provide a clear and concise explanation of the function's purpose, parameters, return value, and examples.
66 # The manual docstring is straightforward and easy to understand, while the AI-generated docstring follows
67 # a more formal structure with sections for arguments, returns, and examples.
68 # Both docstrings are correct and complete, covering all necessary information for users to understand how to use the function effectively.

```

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

```

64 #Write a Python class sru_student with attributes name, roll_no, hostel_status and methods fee_update() and display_details().
65 # First add manual inline comments for each line/block.
66 # Then generate an AI-commented version of the same code.
67 # Compare both and identify missing, redundant, and incorrect AI comments.
68 class sru_student:
69     def __init__(self, name, roll_no, hostel_status):
70         # Initialize the student's name, roll number, and hostel status
71         self.name = name
72         self.roll_no = roll_no
73         self.hostel_status = hostel_status
74         self.fee = 0 # Initialize fee to 0
75
76     def fee_update(self, amount):
77         # Update the student's fee by adding the specified amount
78         self.fee += amount
79
80     def display_details(self):
81         # Display the student's details including name, roll number, hostel status, and fee
82         print(f"Name: {self.name}")
83         print(f"Roll No: {self.roll_no}")
84         print(f"Hostel Status: {self.hostel_status}")
85         print(f"Fee: {self.fee}")
86 # AI-commented version
87 class sru_student:
88     def __init__(self, name, roll_no, hostel_status):
89         # Constructor to initialize the student's attributes
90         self.name = name # Student's name
91         self.roll_no = roll_no # Student's roll number
92         self.hostel_status = hostel_status # Student's hostel status
93         self.fee = 0 # Initialize fee to 0
94
95     def fee_update(self, amount):
96         # Method to update the student's fee by adding the specified amount
97         self.fee += amount # Add the amount to the existing fee
98

```

```

88     def __init__(self, name, roll_no, hostel_status):
89         # Constructor to initialize the student's attributes
90         self.name = name # Student's name
91         self.roll_no = roll_no # Student's roll number
92         self.hostel_status = hostel_status # Student's hostel status
93         self.fee = 0 # Initialize fee to 0
94
95     def fee_update(self, amount):
96         # Method to update the student's fee by adding the specified amount
97         self.fee += amount # Add the amount to the existing fee
98
99     def display_details(self):
100         # Method to display the student's details
101         print(f"Name: {self.name}") # Print the student's name
102         print(f"Roll No: {self.roll_no}") # Print the student's roll number
103         print(f"Hostel Status: {self.hostel_status}") # Print the student's hostel sta
104         print(f"Fee: {self.fee}") # Print the current fee of the student
105 # Analysis:
106 # The AI comments are generally accurate and provide a clear explanation of each line o
107 # there are some redundant comments, such as "Constructor to initialize the student's a
108 # which is already implied by the method name `__init__`.
109 # Additionally, the comment "Add the amount to the existing fee" is somewhat redundant
110 # the method name `fee update` already suggests that it updates the fee. Overall, while

```

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

```

13 # Create a Python calculator module with 3-4 functions (add, subtract, multiply, divide).
14 # First write manual NumPy-style docstrings for each function.
15 # Then generate an AI-written module-level docstring and function-level docstrings.
16 # Compare manual vs AI documentation and evaluate structure, accuracy, and readability.
17 def add(a, b):
18     """
19     Add two numbers.
20
21     Parameters
22     -----
23     a : int or float
24         The first number to be added.
25     b : int or float
26         The second number to be added.
27
28     Returns
29     -----
30     int or float
31         The sum of a and b.
32
33     Examples
34     -----
35     >>> add(2, 3)
36     5
37     >>> add(1.5, 2.5)
38     4.0
39     """
40     return a + b
41 def subtract(a, b):
42     """
43     Subtract one number from another.
44
45     Parameters
46     -----
47     a : int or float
48         The number from which to subtract.
49     b : int or float
50         The number to be subtracted.
51
52     Returns
53     -----
54     int or float
55         The difference of a and b.
56
57     Examples
58     -----
59     >>> subtract(5, 3)
60     2
61     >>> subtract(2.5, 1.5)
62     1.0
63     """
64     return a - b
65 def multiply(a, b):
66     """
67     Calculate the product of a and b.
68
69     Parameters
70     -----
71     a : int or float
72         The first number to be multiplied.
73     b : int or float
74         The second number to be multiplied.
75
76     Returns
77     -----
78     int or float
79         The product of a and b.
80
81     Examples
82     -----
83     >>> multiply(2, 3)
84     6
85     >>> multiply(1.5, 2.5)
86     3.75
87     """
88     return a * b
89 def divide(a, b):
90     """ Divide one number by another.
91
92     Parameters
93     -----
94     a : int or float
95         The number to be divided.
96     b : int or float
97         The number by which to divide.
98
99     Returns
100    -----
101    int or float
102        The quotient of a and b.
103
104    Raises
105    -----
106    ZeroDivisionError
107        If b is zero.
108    """
109     if b == 0:
110         raise ZeroDivisionError("Division by zero")
111     return a / b

```



```

199     -----
200     >>> divide(6, 3)
201     2.0
202     >>> divide(7, 2)
203     3.5
204     >>> divide(5, 0)
205     Traceback (most recent call last):
206     |     ...
207     |     ZeroDivisionError: division by zero
208     |     """
209     |     if b == 0:
210     |         raise ZeroDivisionError("division by zero")
211     |     return a / b
212     # AI-generated module-level docstring
213     """
214     This module provides basic arithmetic operations including addition, subtraction, multiplication, and division.
215     Functions
216     -----
217     - add(a, b): Returns the sum of a and b.
218     - subtract(a, b): Returns the difference of a and b.
219     - multiply(a, b): Returns the product of a and b.
220     - divide(a, b): Returns the quotient of a and b, raises ZeroDivisionError if b is zero.
221     """
222     # AI-generated function-level docstrings
223     def add(a, b):
224         """
225         Add two numbers and return the result.
226         Parameters:
227         a (int or float): The first number to be added.
228         b (int or float): The second number to be added.
229         Returns:
230         int or float: The sum of a and b.
231         """
232         return a + b
233     def subtract(a, b):
234         """
235         Subtract one number from another and return the result.
236         Parameters:
237         a (int or float): The number from which to subtract.
238         b (int or float): The number to be subtracted.
239         Returns:
240         int or float: The difference of a and b.
241         """

```

```

242         return a - b
243     def multiply(a, b):
244         """
245         Multiply two numbers and return the result.
246         Parameters:
247         a (int or float): The first number to be multiplied.
248         b (int or float): The second number to be multiplied.
249         Returns:
250         int or float: The product of a and b.
251         """
252         return a * b
253     def divide(a, b):
254         """Divide one number by another and return the result. Raises ZeroDivisionError if b is zero.
255         Parameters:
256         a (int or float): The number to be divided.
257         b (int or float): The number by which to divide.
258         Returns:
259         int or float: The quotient of a and b.
260         Raises:
261         ZeroDivisionError: If b is zero.
262         """
263         if b == 0:
264             raise ZeroDivisionError("division by zero")
265         return a / b
266     # Analysis:
267     # The manual NumPy-style docstrings are well-structured and provide detailed information about the parameters, return values,
268     # and examples for each function.
269     # The AI-generated module-level docstring is concise and effectively summarizes the purpose of the module and its functions.
270     # The AI-generated function-level docstrings are accurate and provide clear explanations of the parameters, return values,
271     # and exceptions for each function. However, they are less detailed than the manual docstrings,
272     # lacking examples and a more formal structure. Overall, both the manual and AI-generated docstrings are accurate and readable,
273     # but the manual docstrings offer more comprehensive information for users.
274

```