

# AI Assisted Coding

## Assignment 9.3

Name: R.Bharadwaj

Hall ticket no: 2303A51610

Batch no: 19

### Task 1: Basic Docstring Generation

#### Prompt:

Write a Python function that returns the sum of even numbers and the sum of odd numbers in a given list.

Add a Google Style docstring manually and then generate a docstring using AI assistance for the same function.

#### Code & Output:

The screenshot shows a code editor with a dark theme and a terminal window below it. The code editor has tabs for 'PROBLEMS', 'OUTPUT', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected, showing the command 'python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI\_Assistant\_coding/assignment\_9.3.py"' and its output: 'Sum of even numbers: 6' and 'Sum of odd numbers: 9'. The status bar at the bottom right indicates 'Ln 61, Col 1' and 'Spaces: 4'.

```
assignment_9.3.py X
AI-Assistant_coding > assignment_9.3.py > ...
1  # Task 1: Basic Docstring Generation
2  # Write a Python function that returns the sum of even numbers and the sum
3  # of odd numbers in a given list.
4
5  # Manual Google-style docstring
6  def sum_even_odd(numbers):
7      """
8          Calculate the sum of even and odd numbers in a list.
9
10         Args:
11             numbers (list): A list of integers.
12
13         Returns:
14             tuple: A tuple containing the sum of even numbers
15                  and the sum of odd numbers.
16      """
17
18      even_sum = 0
19      odd_sum = 0
20
21      for num in numbers:
22          if num % 2 == 0:
23              even_sum += num
24          else:
25              odd_sum += num
26
27      return even_sum, odd_sum
28
29
30  # AI-generated style docstring
31  def sum_even_odd_ai(numbers):
...
TERMINAL
python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
Sum of even numbers: 6
Sum of odd numbers: 9
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
Ln 61, Col 1  Spaces: 4  UTF-8  CRL
```

```
assignment_9.3.py > AI_Assistant_coding > assignment_9.3.py > ...
30     # AI-generated style docstring
31     def sum_even_odd_ai(numbers):
32         """
33             Computes the total sum of even and odd integers from a list.
34
35             Parameters:
36                 numbers (list of int): List containing integer values.
37
38             Returns:
39                 tuple: (sum_of_even_numbers, sum_of_odd_numbers)
40         """
41
42         even_sum = 0
43         odd_sum = 0
44
45         for num in numbers:
46             if num % 2 == 0:
47                 even_sum += num
48             else:
49                 odd_sum += num
50
51     return even_sum, odd_sum
52
53
54     # Example usage
55     nums = [1, 2, 3, 4, 5]
56
57     even, odd = sum_even_odd(nums)
58
59     print("Sum of even numbers:", even)
60     print("Sum of odd numbers:", odd)
PROBLEMS OUTPUT TERMINAL PORTS ... | ⌂ ×
> ▾ TERMINAL
python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
Sum of even numbers: 6
Sum of odd numbers: 9
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
Q Ln 61, Col 1 Spaces: 4 UTF-8 CRL
```

## Explanation:

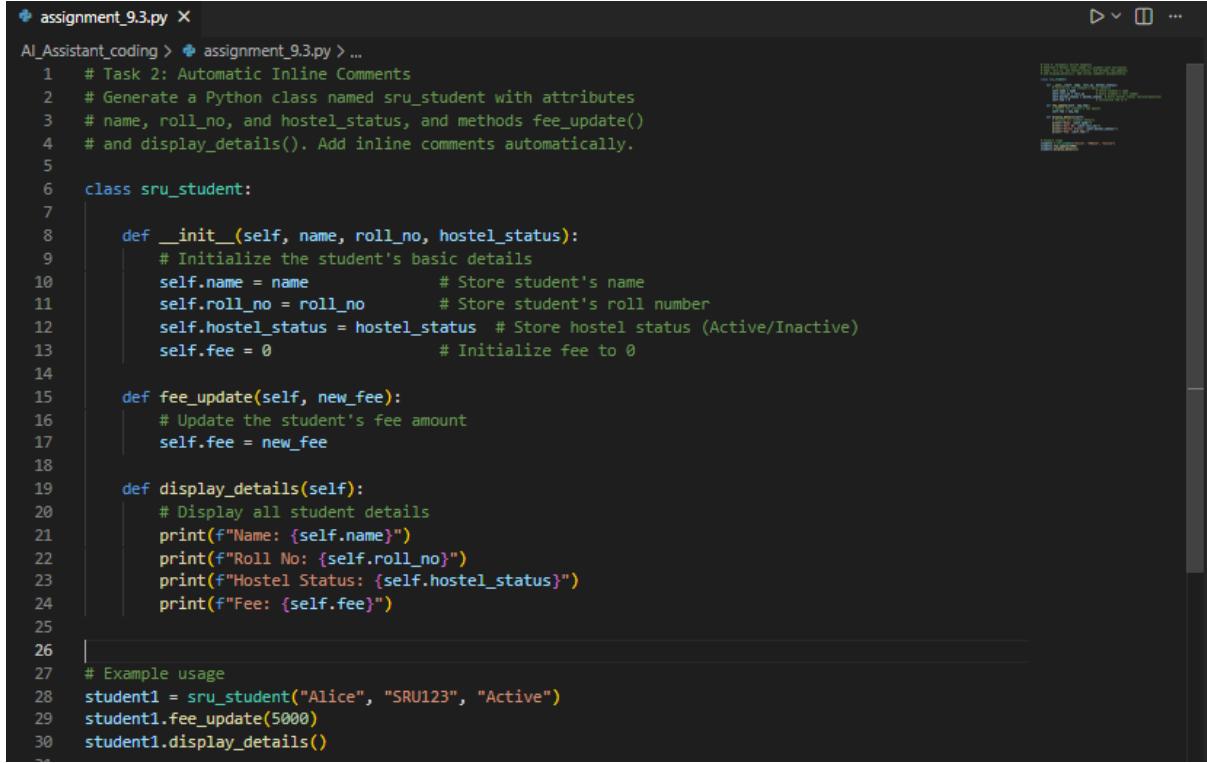
The manual docstring gives a clear and detailed explanation of the function's purpose and return value. The AI-generated version is shorter and accurate but less descriptive. This comparison shows that AI can create correct documentation, yet human refinement is needed to make it more complete and clear.

## Task 2: Automatic Inline Comments

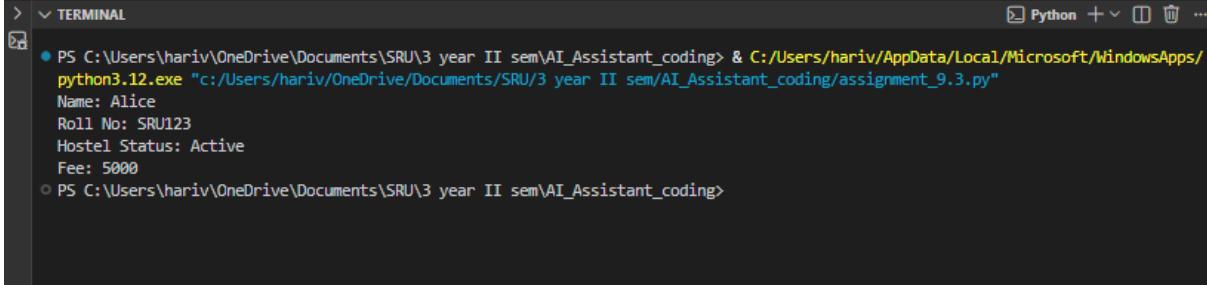
### Prompt:

Generate a Python class named sru\_student with attributes name, roll\_no, and hostel\_status, and methods fee\_update() and display\_details(). Add inline comments automatically.

## Code & Output:



```
assignment_9.3.py X
AI-Assistant_coding > assignment_9.3.py > ...
1 # Task 2: Automatic Inline Comments
2 # Generate a Python class named sru_student with attributes
3 # name, roll_no, and hostel_status, and methods fee_update()
4 # and display_details(). Add inline comments automatically.
5
6 class sru_student:
7
8     def __init__(self, name, roll_no, hostel_status):
9         # Initialize the student's basic details
10        self.name = name          # Store student's name
11        self.roll_no = roll_no    # Store student's roll number
12        self.hostel_status = hostel_status # Store hostel status (Active/Inactive)
13        self.fee = 0              # Initialize fee to 0
14
15    def fee_update(self, new_fee):
16        # Update the student's fee amount
17        self.fee = new_fee
18
19    def display_details(self):
20        # Display all student details
21        print(f"Name: {self.name}")
22        print(f"Roll No: {self.roll_no}")
23        print(f"Hostel Status: {self.hostel_status}")
24        print(f"Fee: {self.fee}")
25
26
27 # Example usage
28 student1 = sru_student("Alice", "SRU123", "Active")
29 student1.fee_update(5000)
30 student1.display_details()
```



```
> < TERMINAL
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
Name: Alice
Roll No: SRU123
Hostel Status: Active
Fee: 5000
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

## Explanation:

Manual comments are clear, specific, and reflect the developer's intent. AI-generated comments are accurate but tend to be more general and sometimes explain obvious code lines. This shows that while AI can speed up documentation, human review is important to ensure the comments are meaningful and not repetitive.

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

## Prompt:

Generate a Python calculator module with functions add, subtract, multiply, and divide.

Add NumPy-style docstrings manually and then generate module-level and function-level documentation using AI assistance.

## **Code & Output:**

The screenshot shows a code editor window with the following details:

- Title Bar:** The title bar displays the file name "assignment\_9.3.py".
- Code Content:** The code is a Python module named "Calculator Module". It includes a docstring at the top describing the module's purpose and available functions. It also defines an "add" function with parameters "a" and "b", which adds them together.
- TERMINAL Tab:** The terminal tab is active, showing the command "python3.12.exe" followed by the path to the script. The output of the script execution is displayed below, showing the results of running the "add" function with various arguments.
- Bottom Status Bar:** The status bar at the bottom indicates the current line (Ln 107), column (Col 30), and other settings like spaces per tab (Spaces: 4) and encoding (UTF-8).

```
assignment_93.py X
AI_Assistant_coding > assignment_93.py > ...
19 def add(a, b):
20     """
21     Returns
22     -----
23     int or float
24     | The sum of a and b.
25     """
26     return a + b
27
28 def subtract(a, b):
29     """
30     Subtract two numbers.
31
32     Parameters
33     -----
34     a : int or float
35     | The first number.
36     b : int or float
37     | The second number.
38
39     Returns
40     -----
41     int or float
42     | The difference of a and b.
43     """
44     return a - b
45
46 def multiply(a, b):
47     """
48     Multiply two numbers.
49
50     Parameters
51     -----
52     a : int or float
53     | The first number.
54     b : int or float
55     | The second number.
56
57     Returns
58     -----
59     int or float
60     | The product of a and b.
61     """
62     return a * b
63
64 def divide(a, b):
65     """
66     Divide two numbers.
67
68     Parameters
69     -----
70     a : int or float
71     | The numerator.
72     b : int or float
73     | The denominator.
74
75     Returns
76     -----
77     int or float
78     | The quotient of a divided by b.
79
80     Raises
81     -----
82     ValueError
83     | If b is zero.
84     """
85     if b == 0:
86         raise ValueError("Cannot divide by zero.")
87     return a / b
88
89 # Example Usage
90 if __name__ == "__main__":
91     print(add(10, 5))
92     print(subtract(10, 5))
93     print(multiply(10, 5))
94     print(divide(10, 5))
```

```
assignment_93.py X
AI_Assistant_coding > assignment_93.py > ...
57 def multiply(a, b):
58     """
59     Returns
60     -----
61     int or float
62     | The product of a and b.
63     """
64     return a * b
65
66 def divide(a, b):
67     """
68     Divide two numbers.
69
70     Parameters
71     -----
72     a : int or float
73     | The numerator.
74     b : int or float
75     | The denominator.
76
77     Returns
78     -----
79     int or float
80     | The quotient of a divided by b.
81
82     Raises
83     -----
84     ValueError
85     | If b is zero.
86     """
87     if b == 0:
88         raise ValueError("Cannot divide by zero.")
89     return a / b
90
91 # Example Usage
92 if __name__ == "__main__":
93     print(add(10, 5))
94     print(subtract(10, 5))
95     print(multiply(10, 5))
96     print(divide(10, 5))
```

## **Explanation:**

Manual NumPy-style docstrings follow a well-structured scientific format with clear sections for parameters and return values. The AI-generated version is concise and useful for general overviews but does not provide detailed parameter-level explanations. While AI is strong at summarizing, manual documentation offers better technical depth and clarity.