

AI Assisted Coding

Assignment 9.3

Name: ch. koushik

Hall ticket no: 2303A51938

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:

```
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):
PROBLEMS OUTPUT TERMINAL PORTS ... | ⌂ X
> 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 X
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 ... | ⌂ X
> 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:

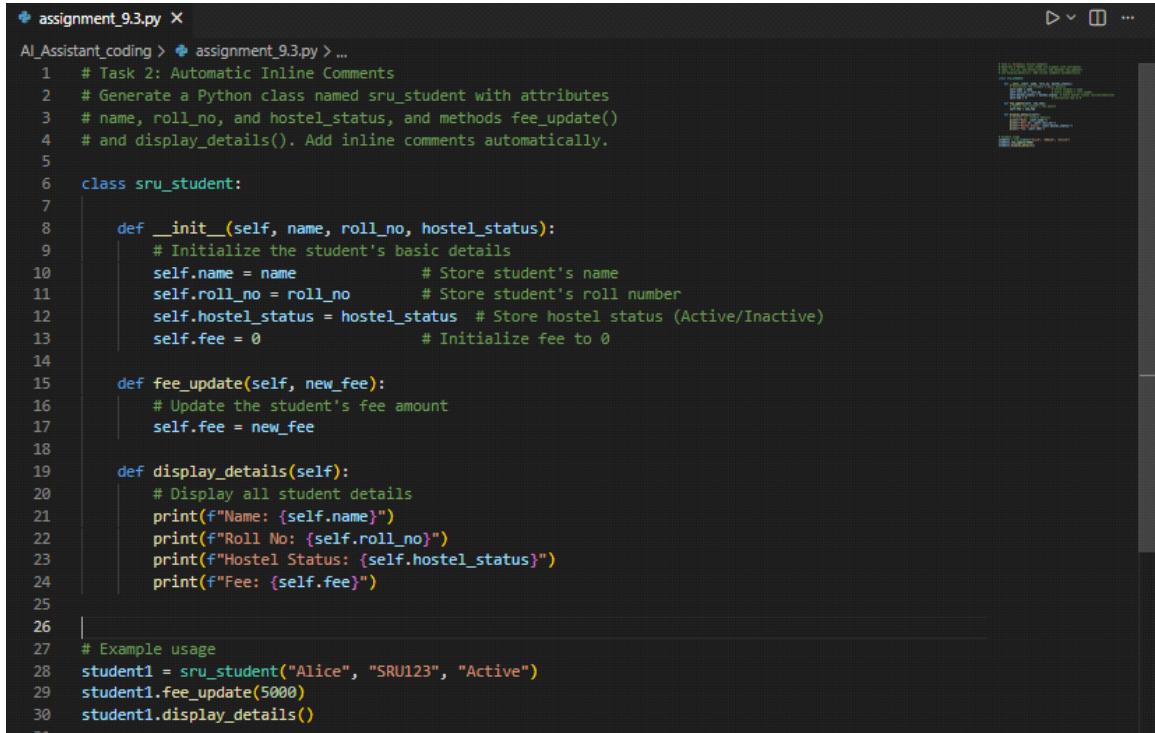
The manual docstring gives a clear and detailed explanation of the function's purpose and return value. The AI-generated version is shorter and accurate but less descriptive. This comparison shows that AI can create correct documentation, yet human refinement is needed to make it more complete and clear.

Task 2: Automatic Inline Comments

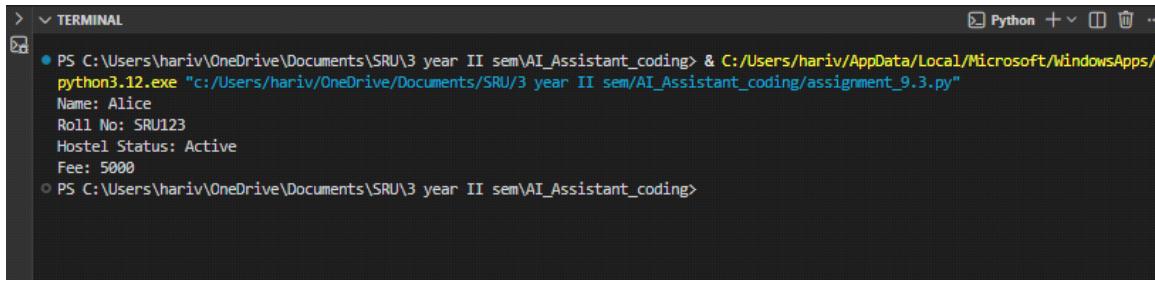
Prompt:

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 interface with a dark theme. On the left, the file `assignment_9.3.py` is open, displaying Python code for a calculator module. The code includes module-level documentation, function-level documentation for `add`, and a terminal window at the bottom showing the execution of the module. The terminal output shows the results of calling the `add` function with various arguments.

```
assignment_9.3.py X
AI_Assistant_coding > assignment_9.3.py > ...

1 """
2 Calculator Module
3 =====
4
5 This module provides basic arithmetic operations:
6 addition, subtraction, multiplication, and division.
7
8 Functions:
9     add(a, b)      -> Returns the sum of two numbers
10    subtract(a, b)  -> Returns the difference of two numbers
11    multiply(a, b)  -> Returns the product of two numbers
12    divide(a, b)   -> Returns the quotient of two numbers
13
14 Raises:
15     ValueError -> If division by zero is attempted
16 """
17
18
19 def add(a, b):
20     """
21     Add two numbers.
22
23     Parameters
24     -----
25     a : int or float
26         The first number.
27     b : int or float
28         The second number.
29
30
31     PROBLEMS OUTPUT TERMINAL PORTS
32     > TERMINAL
33     python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_9.3.py"
34     15
35     5
36     50
37     2.0
38     PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
39
40     Q Ln 107, Col 30 Spaces: 4 UTF-8 CRL
```

```
assignment_9.3.py X
AI_Assistant_coding > assignment_9.3.py > ...
19 def add(a, b):
30     Returns
31     -----
32     int or float
33     | The sum of a and b.
34     """
35     return a + b
36
37
38 def subtract(a, b):
39     """
40     Subtract two numbers.
41
42     Parameters
43     -----
44     a : int or float
45     | The first number.
46     b : int or float
47     | The second number.
48
49     Returns
50     -----
51     int or float
52     | The difference of a and b.
53     """
54     return a - b
55
56
57 def multiply(a, b):
58     """
59     Multiply two numbers.
60
61     Parameters
62     -----
63     a : int or float
64     | The first number.
65     b : int or float
66     | The second number.
67
```

The screenshot shows a code editor window with a dark theme. On the left is a vertical scroll bar. The main area contains Python code for four functions: `add`, `subtract`, `multiply`, and `divide`. The code includes detailed docstrings with sections for parameters, return values, and exceptions. The right side of the editor shows a file tree with various files and folders. At the bottom, there are status indicators: a magnifying glass icon, "Ln 107, Col 30", "Spaces: 4", "UTF-8", and "CRL".

```
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
92 # Example Usage
93 if __name__ == "__main__":
94     print(add(10, 5))
95     print(subtract(10, 5))
96     print(multiply(10, 5))
97     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.