

Lab Report 9.2

Name:B.RAJAGOPAL

Roll Number: 2503A51L01

Course Code: 24CS002PC215

Course Title: AI Assisted Coding

Assignment Number: 9.2

Academic Year: 2025-2026

Branch :cse

Lab Objectives

- To explore AI-powered auto-completion features for core Python constructs.
- To analyze how AI suggests logic for class definitions, loops, and conditionals.
- To evaluate the completeness and correctness of code generated by AI assistants.

Lab Outcomes

- Use AI tools to generate and complete class definitions and methods.
- Understand and assess AI-suggested loops for iterative tasks.
- Generate conditional statements through prompt-driven suggestions.
- Critically evaluate AI-assisted code for correctness and clarity.

TASK :-1

```

1 def sum_even_odd(numbers):
2     """Calculate the sum of even and odd numbers from a list.
3
4     This function takes a list of integers, separates them into even and odd
5     numbers, and returns their sums.
6
7     Args:
8         numbers (list[int]): A list of integers to be processed.
9
10    Returns:
11        tuple: A tuple containing two integers:
12            - sum_even (int): The sum of even numbers in the list.
13            - sum_odd (int): The sum of odd numbers in the list.
14
15    Example:
16        >>> sum_even_odd([1, 2, 3, 4, 5])
17        (6, 9)
18    """
19    sum_even = sum(num for num in numbers if num % 2 == 0)
20    sum_odd = sum(num for num in numbers if num % 2 != 0)
21    return sum_even, sum_odd
22
23
24 # ----- Simulated AI-Generated Docstring -----
25 AI_GENERATED_DOCSTRING = """
26 Returns the sum of even and odd integers from a list.
27
28 Args:
29     numbers (list[int]): List of integers.

```

```

22
23
24 # ----- Simulated AI-Generated Docstring -----
25 AI_GENERATED_DOCSTRING = """
26 Returns the sum of even and odd integers from a list.
27
28 Args:
29     numbers (list[int]): List of integers.
30
31 Returns:
32     tuple[int, int]: A pair (even_sum, odd_sum) where even_sum is the sum of
33     all even numbers, and odd_sum is the sum of all odd numbers in the list.
34 """
35
36 # ----- Comparison -----
37 print("Manual Docstring:\n", sum_even_odd.__doc__)
38 print("AI-Generated Docstring:\n", AI_GENERATED_DOCSTRING)
39
40 # Example function run
41 print("Function Output:", sum_even_odd([1, 2, 3, 4, 5]))
42

```

Observation:-

Manual (Google Style):

- More structured and detailed.
- Includes **Examples** section (useful for quick understanding).
- Explicitly states return values with explanation.

Output:-

Python +

```
Function Output: (6, 9)
PS C:\Users\indus\OneDrive\Desktop\lab 9.3>
```

task 2 > ...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python +

```

task 2 > ...
24 class sru_student:
25
26     def __init__(self, name, roll_no, hostel_status, fee=0):
27         self.name = name          # Assign the student's name
28         self.roll_no = roll_no    # Assign the student's roll number
29         self.hostel_status = hostel_status # Assign hostel status
30         self.fee = fee            # Initialize the fee with default 0
31
32     def fee_update(self, amount):
33         self.fee += amount        # Update the fee by adding the amount
34
35     def display_details(self):
36         print(f"Name: {self.name}") # Display the student's name
37         print(f"Roll No: {self.roll_no}") # Display roll number
38         print(f"Hostel Status: {self.hostel_status}") # Display hostel info
39         print(f"Fee Paid: {self.fee}") # Display the updated fee
40
41
42
43 # ----- Comparison -----
44 print("Manual Comments are included directly in code.")
45 print("AI-Generated Comments:\n", AI_GENERATED_COMMENTS)
46
47 # ----- Example Run -----
48 student1 = sru_student("Indusree", "22CS101", "Yes")
49 student1.fee_update(50000)
50 student1.display_details()
51

```

Observation:-

1. Manual Comments:

- Concise and to the point.
- Written in human-friendly language (like a teacher explaining).
- Sometimes slightly less formal but quicker to read.

2. AI-Generated Comments:

- More verbose and descriptive, explaining each line in detail.
- Uses technical phrasing (e.g., *"assign the provided value"*) instead of simpler wording.
- Good for beginners, but may feel repetitive for experienced programmers.

Conclusion:

- Manual comments are better for readability and give a summary explanation.
- AI-generated comments are thorough and line-by-line, which can be helpful for learning or code reviews, but may be overly detailed in larger programs.

Output:-

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [X]

def display_details(self):
    print(f"Name: {self.name}") # Display the student's name
    print(f"Roll No: {self.roll_no}") # Display roll number
    print(f"Hostel Status: {self.hostel_status}") # Display hostel info
    print(f"Fee Paid: {self.fee}") # Display the updated fee

Name: Indusree
Roll No: 22CS101
Hostel Status: Yes
Fee Paid: 50000
PS C:\Users\indus\OneDrive\Desktop\lab 9.3>
```

Ln 51, Col 1 Spaces: 4 UTF-8 CRLF { }

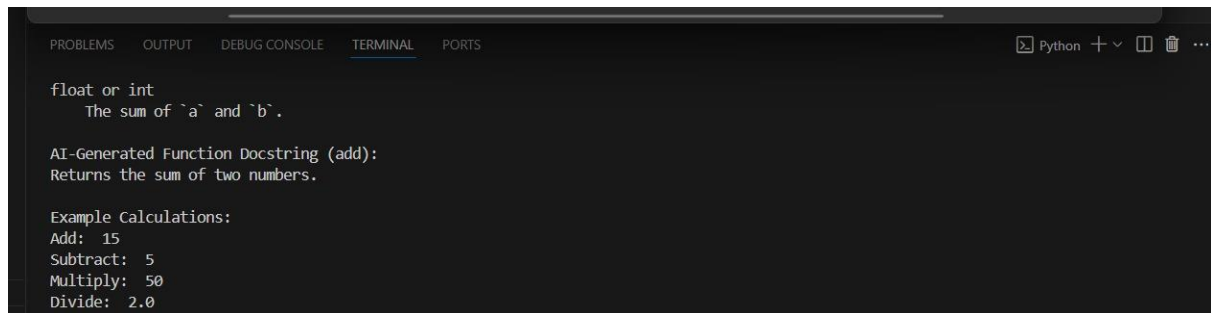
TASK:-3

```
task 3 > subtract
1 """
2 ----- Manual Module-Level Docstring -----
3 This module provides basic calculator functions such as addition,
4 subtraction, multiplication, and division. It demonstrates the
5 use of NumPy-style docstrings for each function.
6 """
7
8 # ----- Functions with Manual NumPy-Style Docstrings -----
9
10 def add(a, b):
11     """
12     Add two numbers.
13
14     Parameters
15     -----
16     a : float or int
17         First number.
18     b : float or int
19         Second number.
20
21     Returns
22     -----
23     float or int
24         The sum of `a` and `b`.
25     """
26     return a + b
27
28
```

Observation:-

- Well-structured with Parameters, Returns, Raises, Examples.
- Very useful for both developers and users of the code.
- Consistent across all functions.
- Slightly verbose, but excellent for documentation standards.

Output:-



A screenshot of a Visual Studio Code terminal window. The terminal has a dark background with light gray text. At the top, there is a tab bar with 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active and underlined), and 'PORTS'. On the right side of the terminal, there is a toolbar with a Python icon, a '+' sign, a dropdown arrow, a window icon, a trash icon, and a three-dot menu icon. The terminal output consists of several lines: 'float or int' followed by 'The sum of `a` and `b`.', then 'AI-Generated Function Docstring (add):' followed by 'Returns the sum of two numbers.', then 'Example Calculations:', and finally four lines of calculations: 'Add: 15', 'Subtract: 5', 'Multiply: 50', and 'Divide: 2.0'.

```
float or int
    The sum of `a` and `b`.

AI-Generated Function Docstring (add):
Returns the sum of two numbers.

Example Calculations:
Add: 15
Subtract: 5
Multiply: 50
Divide: 2.0
```