AI-Assisted Documentation & Docstring Assignment Report

Hall Ticket : 2303a51359

Name : A.Abhilashgoud

Task 1: Basic Docstring Generation

Prompt:

Write a Python function to return the sum of even and odd numbers in a list. Add manual Google-style docstring. Generate AI docstring. Compare both.

Implementation with Manual Google-Style Docstring:

```
def sum_even_odd(numbers):
    """Calculate sum of even and odd numbers in a list.

    Args:
        numbers (list[int]): List of integers.

    Returns:
        tuple: (sum_even, sum_odd)

    Raises:
        TypeError: If input is not a list.
    """
    # Validate input type
    if not isinstance(numbers, list):
        raise TypeError("Input must be a list")

    sum_even = 0  # Store even sum
    sum_odd = 0   # Store odd sum

    # Iterate through numbers
    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num

    return sum_even, sum_odd
```

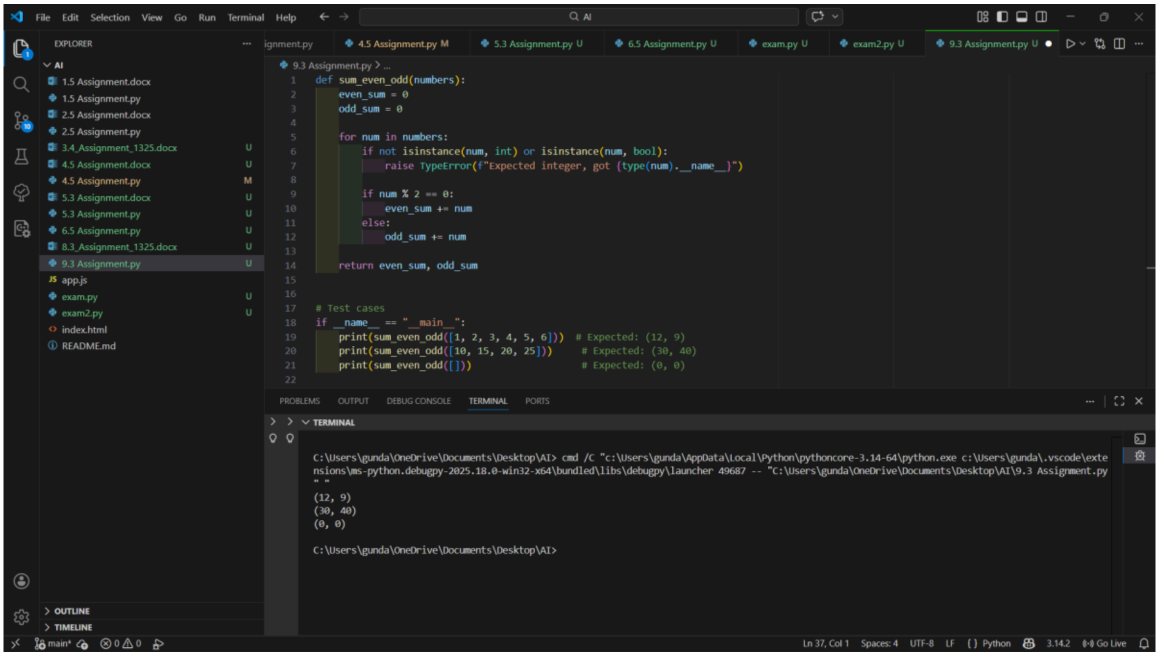
AI-Generated Docstring (Simulated):

```
"""Return the sum of even and odd numbers from a list.

Parameters:
    numbers: List of integers.

Returns:
    Tuple containing even sum and odd sum.
"""
```

Code Explanation:
The function validates input type, iterates through list elements, classifies numbers as even or odd, and returns their respective sums.

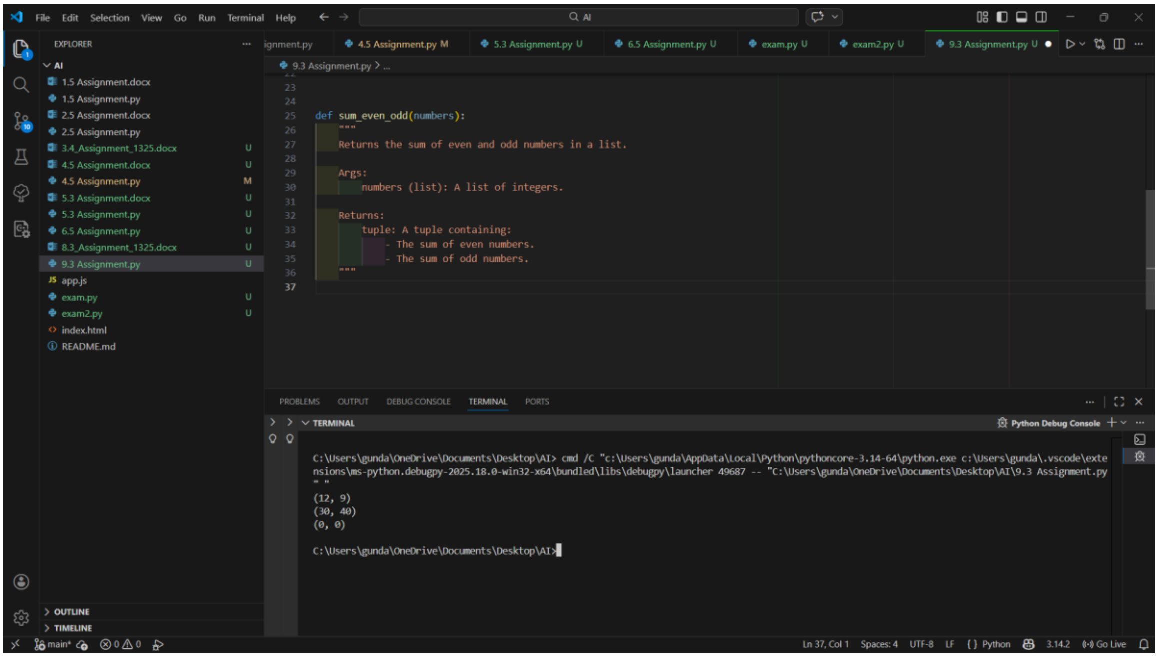
Sample Output:
Input: [1,2,3,4,5]
Output: (6, 9)

Comparison:
Manual docstring includes structured sections (Args, Returns, Raises). AI version is concise but lacks error handling details and structured format.



```python
def sum_even_odd(numbers):
    even_sum = 0
    odd_sum = 0

    for num in numbers:
        if not isinstance(num, int) or isinstance(num, bool):
            raise TypeError(f"Expected integer, got {type(num).__name__}")

        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum


# Test cases
if __name__ == "__main__":
    print(sum_even_odd([1, 2, 3, 4, 5, 6]))  # Expected: (12, 9)
    print(sum_even_odd([10, 15, 20, 25]))     # Expected: (30, 40)
    print(sum_even_odd([]))                   # Expected: (0, 0)
```

```
C:\Users\gunda\OneDrive\Documents\Desktop\AI> cmd /C "c:\Users\gunda\AppData\Local\Python\pythoncore-3.14-64\python.exe c:\Users\gunda\.vscode\exte
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher 49687 -- "C:\Users\gunda\OneDrive\Documents\Desktop\AI\9.3 Assignment.py
" "
(12, 9)
(30, 40)
(0, 0)

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```



```
def sum_even_odd(numbers):
    """
    Returns the sum of even and odd numbers in a list.

    Args:
        numbers (list): A list of integers.

    Returns:
        tuple: A tuple containing:
            - The sum of even numbers.
            - The sum of odd numbers.
    """

```

```
C:\Users\gunda\OneDrive\Documents\Desktop\AI> cmd /C "c:\Users\gunda\AppData\Local\Python\pythoncore-3.14-64\python.exe c:\Users\gunda\.vscode\exte
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher 49687 -- "C:\Users\gunda\OneDrive\Documents\Desktop\AI\9.3 Assignment.py
" "
(12, 9)
(30, 40)
(0, 0)

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

Task 2: Automatic Inline Comments

Prompt:

Create sru_student class with attributes and methods. Add manual inline comments and compare with AI-generated comments.

Manual Inline Comment Version:

```
class sru_student:
  # Initialize student object
  def __init__(self, name, roll_no, hostel_status):
    self.name = name  # Student name
    self.roll_no = roll_no  # Student roll number
    self.hostel_status = hostel_status  # Hostel status (Yes/No)

  # Update student fee
  def fee_update(self, amount):
    self.fee = amount

  # Display student details
  def display_details(self):
    print("Name:", self.name)
    print("Roll No:", self.roll_no)
    print("Hostel Status:", self.hostel_status)
```

AI-Generated Inline Comments (Simulated):

```
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        # Constructor method
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status

    def fee_update(self, amount):
        # Assign fee value
        self.fee = amount

    def display_details(self):
        # Print details
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
```

Code Explanation:

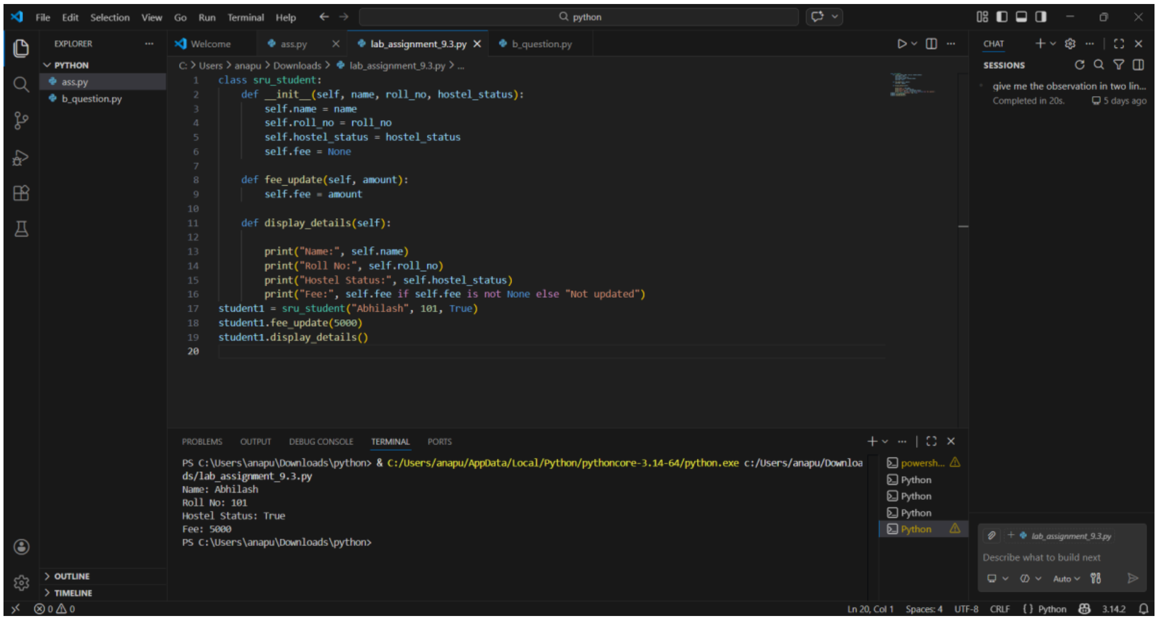
The class stores student details, updates fee information, and prints student information.

Sample Output:

```
Name: Abhilash
Roll No: 101
Hostel Status: True

Fee:5000
```

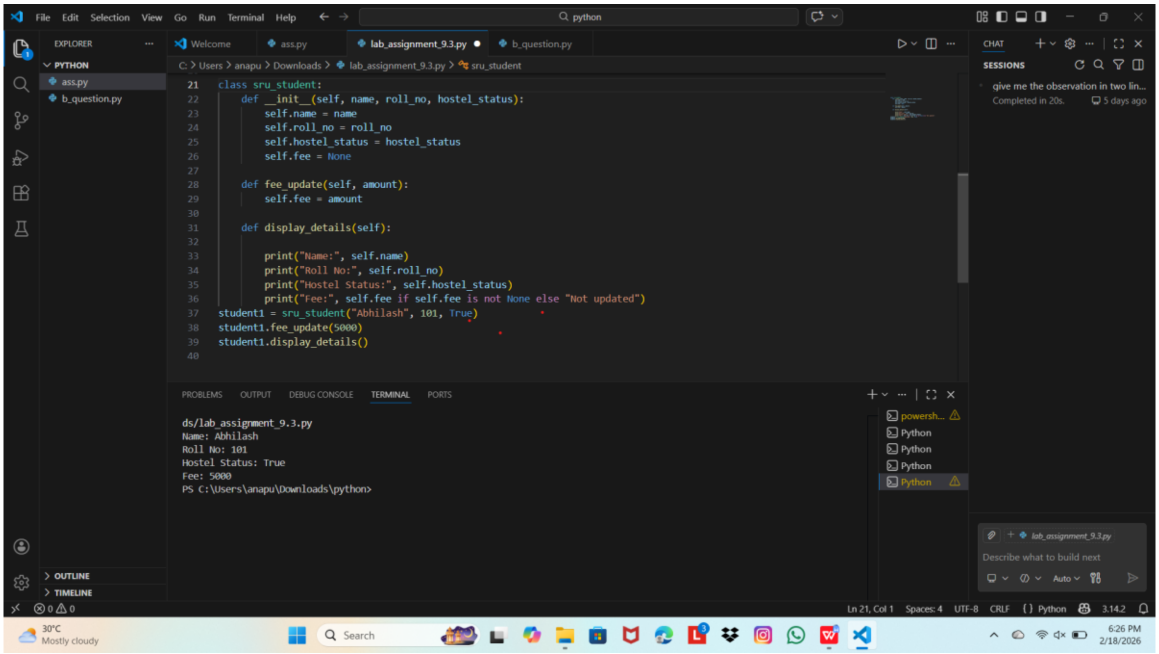
Comparison:

Manual comments explain logic clearly. AI comments are shorter and sometimes obvious. AI may miss context or deeper explanation.



```python
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee = None

    def fee_update(self, amount):
        self.fee = amount

    def display_details(self):

        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee:", self.fee if self.fee is not None else "Not updated")
student1 = sru_student("Abhilash", 101, True)
student1.fee_update(5000)
student1.display_details()
```

```
PS C:\Users\anapu\Downloads\python> & C:/Users/anapu/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/anapu/Downloa
ds/lab_assignment_9.3.py
Name: Abhilash
Roll No: 101
Hostel Status: True
Fee: 5000
PS C:\Users\anapu\Downloads\python>
```



```python
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee = None

    def fee_update(self, amount):
        self.fee = amount

    def display_details(self):

        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee:", self.fee if self.fee is not None else "Not updated")
student1 = sru_student("Abhilash", 101, True)
student1.fee_update(5000)
student1.display_details()
```

```
ds/lab_assignment_9.3.py
Name: Abhilash
Roll No: 101
Hostel Status: True
Fee: 5000
PS C:\Users\anapu\Downloads\python>
```

Task 3: Module-Level and Function-Level Documentation

Prompt:

Create calculator functions. Add manual NumPy-style docstrings. Generate AI module-level docstring. Compare both.

Manual NumPy-Style Function Docstring:

```
def add(a, b):
    """Add two numbers.
    Parameters
    ----------
    a : int or float
    b : int or float

    Returns
    -------
    int or float
        Sum of a and b.
    """
    return a + b



def divide(a, b):
    """Divide two numbers.

    Parameters
    ----------
    a : int or float
    b : int or float

    Returns
    -------
    float

    Raises
    ------
    ZeroDivisionError
    """
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero")
    return a / b
```

AI-Generated Module-Level Docstring (Simulated):

```
"""
This module provides basic arithmetic operations such as addition,
subtraction, multiplication, and division.
```

"""

Code Explanation:

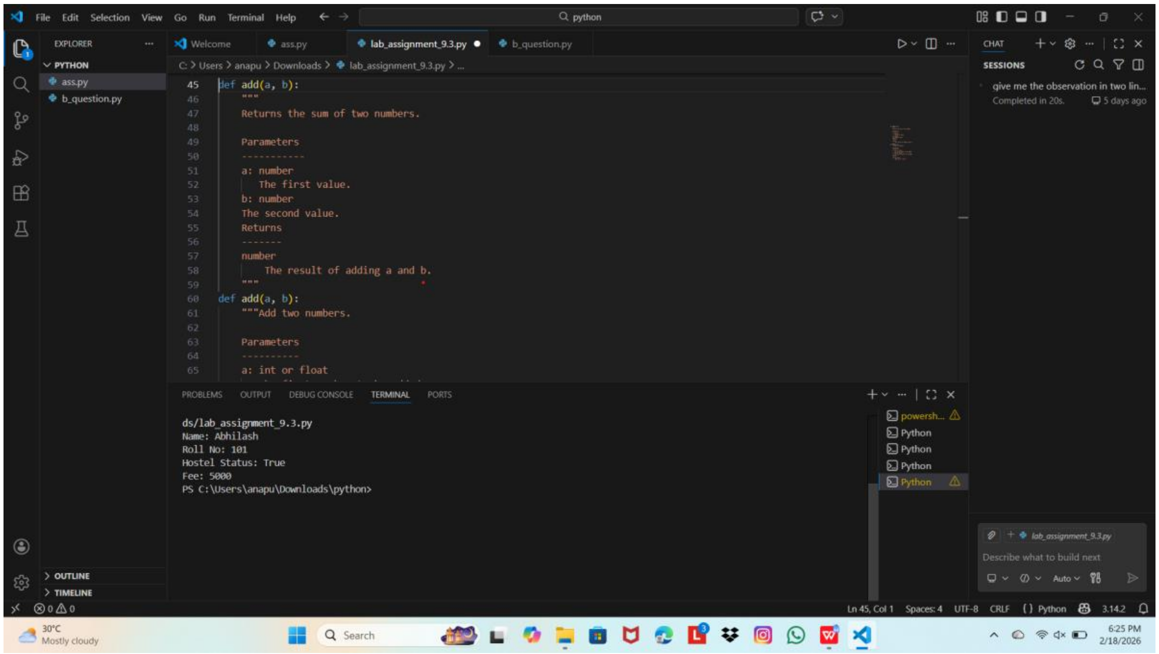
The calculator module performs arithmetic operations with structured documentation for better maintainability.

Sample Output:

add(5,3) → 8
divide(10,2) → 5.0

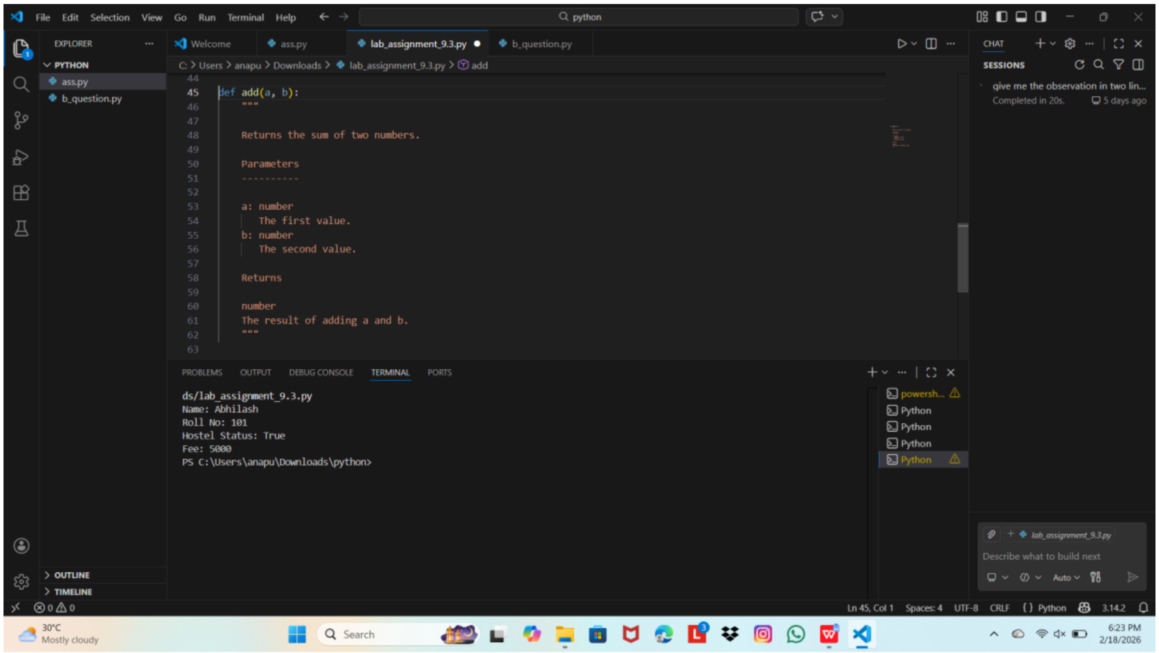
Comparison:

Manual NumPy-style docstrings are more structured and detailed. AI-generated module documentation is concise but lacks parameter depth.



```
def add(a, b):
    """
    Returns the sum of two numbers.

    Parameters
    -----------
    a: number
       The first value.
    b: number
    The second value.
    Returns
    -------
    number
        The result of adding a and b.
    """
def add(a, b):
    """Add two numbers.

    Parameters
    ----------
    a: int or float
```

```
ds/lab_assignment_9.3.py
Name: Abhilash
Roll No: 101
Hostel Status: True
Fee: 5000
PS C:\Users\anapu\Downloads\python>
```



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

    Returns the sum of two numbers.

    Parameters
    ----------

    a: number
       The first value.
    b: number
       The second value.

    Returns

    number
    The result of adding a and b.
    """

```

```
ds/lab_assignment_9.3.py
Name: Abhilash
Roll No: 101
Hostel Status: True
Fee: 5000
PS C:\Users\anapu\Downloads\python>
```