AI Assisted Coding
Lab 9: Documentation Generation – Automatic Documentation and Code Comments

Name: Mujtaba Ahmed

Hall ticket no: 2303A510C2

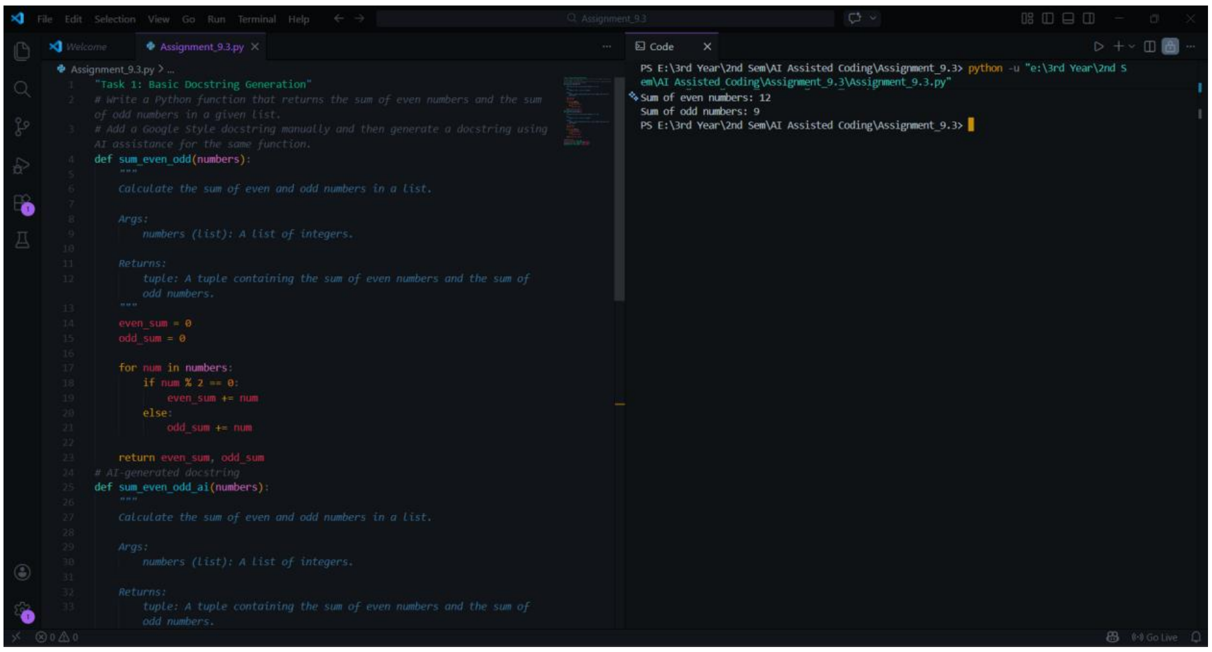
Batch no: 19

Task 1: Basic Docstring Generation

Prompt:

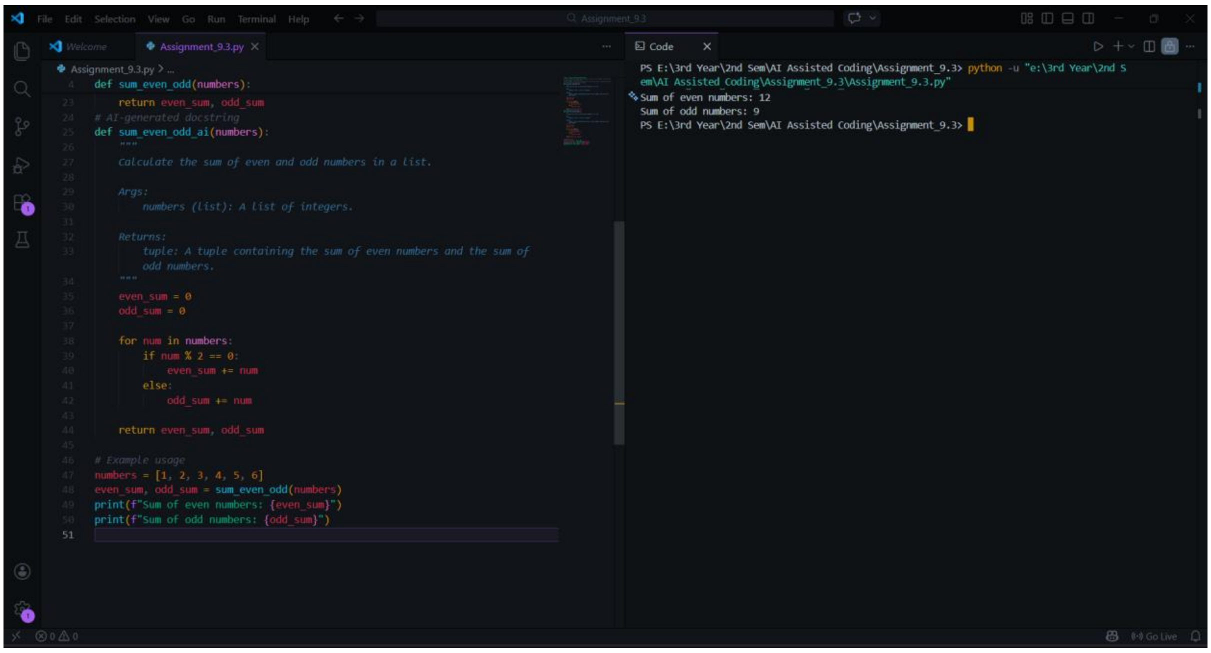
Write a Python function that returns the sum of even numbers and the sum of odd numbers in a given list.
Add a Google Style docstring manually and then generate a docstring using AI assistance for the same function.

Code & Output:



```python
"Task 1: Basic Docstring Generation"
# 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.
def sum_even_odd(numbers):
    """
    Calculate 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 and the sum of
        odd numbers.
    """
    even_sum = 0
    odd_sum = 0

    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum
# AI-generated docstring
def sum_even_odd_ai(numbers):
    """
    Calculate 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 and the sum of
        odd numbers.
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3> python -u "e:\3rd Year\2nd S
em\AI Assisted Coding\Assignment_9.3\Assignment_9.3.py"
Sum of even numbers: 12
Sum of odd numbers: 9
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3>
```



```python
def sum_even_odd(numbers):
    return even_sum, odd_sum
# AI-generated docstring
def sum_even_odd_ai(numbers):
    """
    Calculate 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 and the sum of
        odd numbers.
    """
    even_sum = 0
    odd_sum = 0

    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum

# Example usage
numbers = [1, 2, 3, 4, 5, 6]
even_sum, odd_sum = sum_even_odd(numbers)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3> python -u "e:\3rd Year\2nd S
em\AI Assisted Coding\Assignment_9.3\Assignment_9.3.py"
Sum of even numbers: 12
Sum of odd numbers: 9
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3>
```

Explanation:

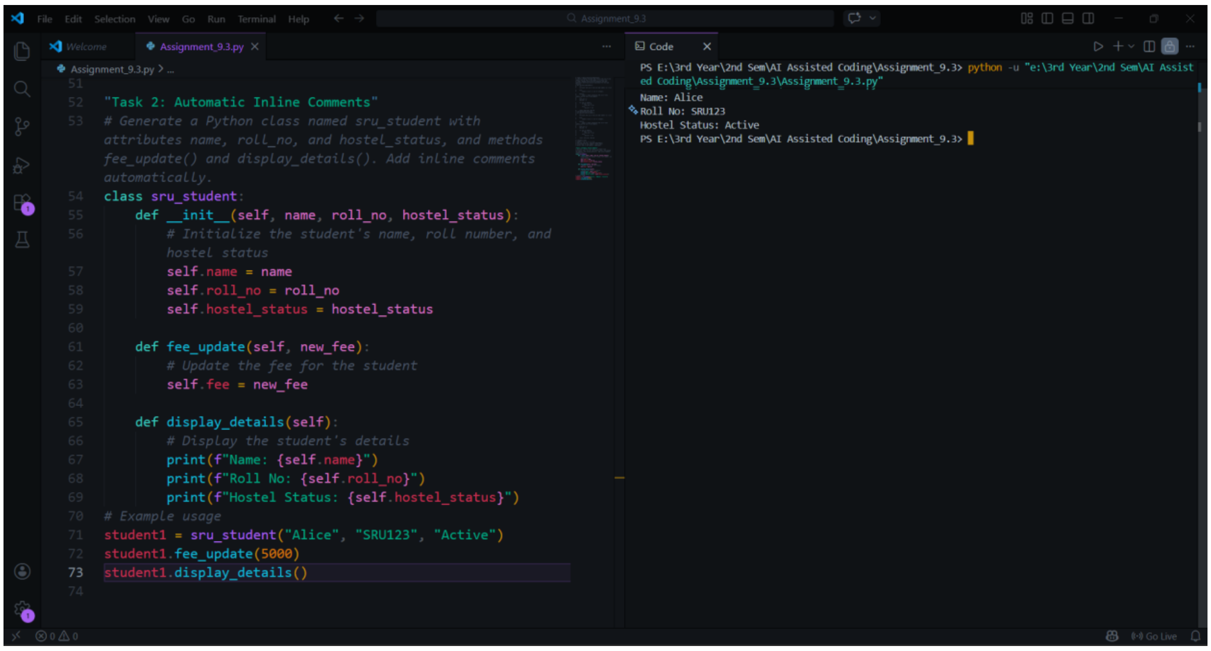
The manually written docstring offers a clearer and more detailed description of the function's objective and the structure of its return value. It specifies the parameter type and explains the output in well-formed sentences, making it easier to understand. In contrast, the AI-generated docstring is brief and to the point, and while it is technically accurate, it does not provide the same level of depth or explanation. This comparison shows that AI-produced documentation can be correct and efficient, but it may still need human editing to improve clarity and completeness.

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:



```python

"Task 2: Automatic Inline Comments"
# 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.
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        # Initialize the student's name, roll number, and hostel status
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status

    def fee_update(self, new_fee):
        # Update the fee for the student
        self.fee = new_fee

    def display_details(self):
        # Display the student's details
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
# Example usage
student1 = sru_student("Alice", "SRU123", "Active")
student1.fee_update(5000)
student1.display_details()

```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3\Assignment_9.3.py"
Name: Alice
Roll No: SRU123
Hostel Status: Active
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3>
```

Explanation:

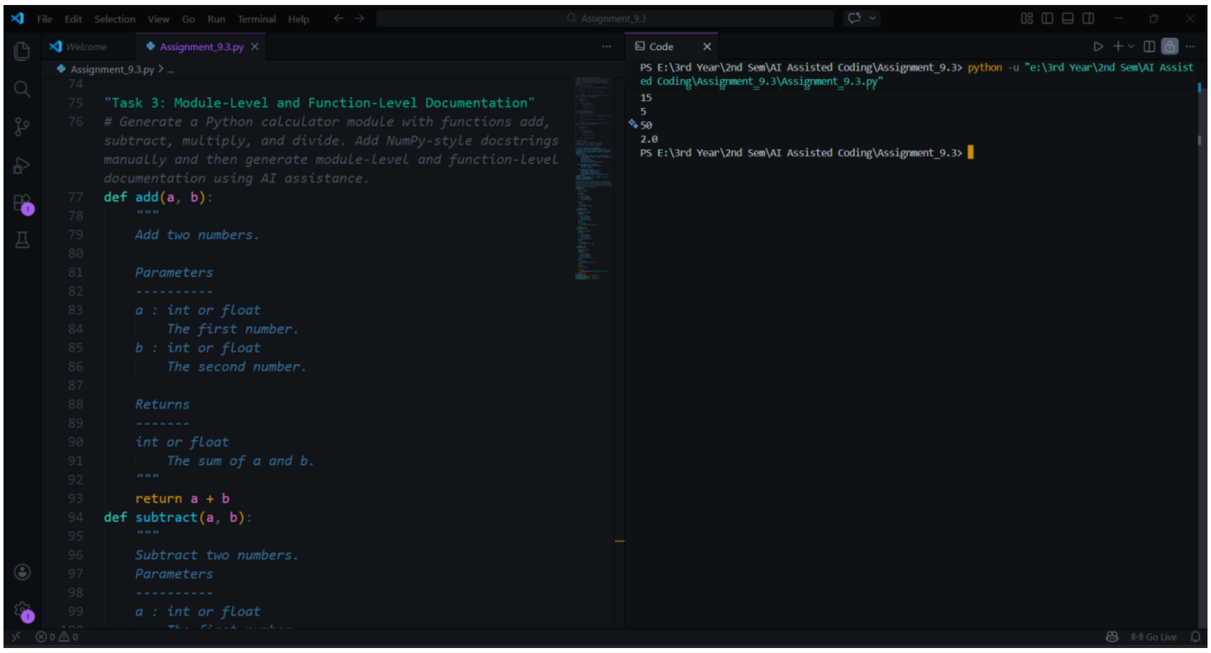
Manually written comments are usually focused and reflect the developer's specific intent. AI-generated comments are accurate as well, but they can sound more general and sometimes explain code that is already obvious.This indicates that AI can speed up the documentation process, but human review is still important to ensure the comments are relevant, clear, and not repetitive.

Task 3: Module-Level and Function-Level Documentation

Prompt:

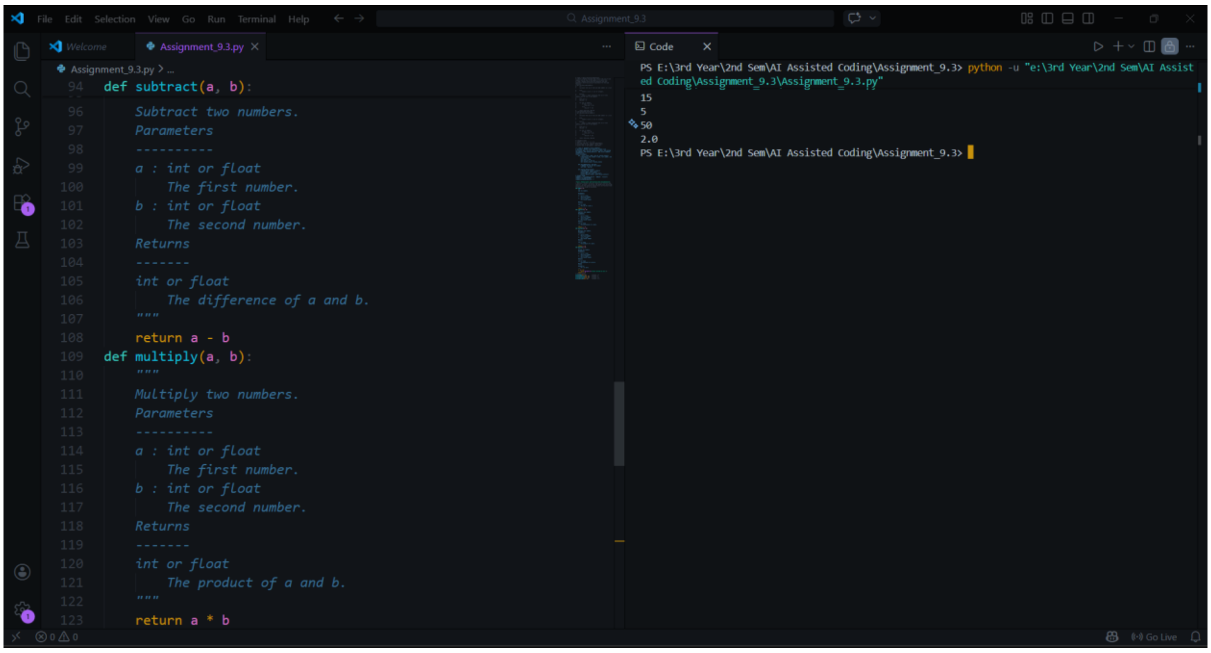
Generate a Python calculator module with functions add, subtract, multiply, and divide. Add NumPy-style docstrings manually and then generate module-level and function-level documentation using AI assistance.

Code & Output:



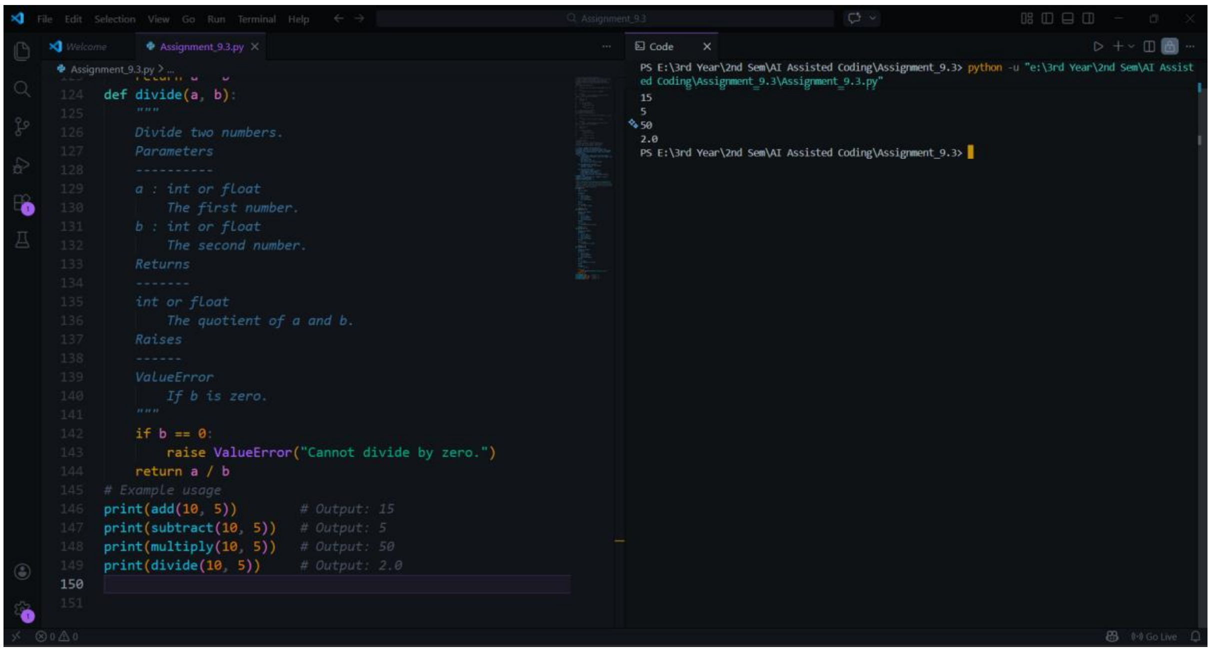
```python
"Task 3: Module-Level and Function-Level Documentation"
# Generate a Python calculator module with functions add,
subtract, multiply, and divide. Add NumPy-style docstrings
manually and then generate module-level and function-level
documentation using AI assistance.
def add(a, b):
    """
    Add two numbers.

    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -------
    int or float
        The sum of a and b.
    """
    return a + b
def subtract(a, b):
    """
    Subtract two numbers.
    Parameters
    ----------
    a : int or float
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3> python -u "e:\3rd Year\2nd Sem\AI Assist
ed Coding\Assignment_9.3\Assignment_9.3.py"
15
5
50
2.0
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3>
```



```python
def subtract(a, b):

    Subtract two numbers.
    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.
    Returns
    -------
    int or float
        The difference of a and b.
    """
    return a - b
def multiply(a, b):
    """
    Multiply two numbers.
    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.
    Returns
    -------
    int or float
        The product of a and b.
    """
    return a * b
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3> python -u "e:\3rd Year\2nd Sem\AI Assist
ed Coding\Assignment_9.3\Assignment_9.3.py"
15
5
50
2.0
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3>
```



```python
def divide(a, b):
    """
    Divide two numbers.
    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.
    Returns
    -------
    int or float
        The quotient of a and b.
    Raises
    ------
    ValueError
        If b is zero.
    """
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b
# Example usage
print(add(10, 5))         # Output: 15
print(subtract(10, 5))    # Output: 5
print(multiply(10, 5))    # Output: 50
print(divide(10, 5))      # Output: 2.0
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3> python -u "e:\3rd Year\2nd Sem\AI Assist
ed Coding\Assignment_9.3\Assignment_9.3.py"
15
5
50
2.0
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3>
```

Explanation:

Manual NumPy-style docstrings use a well-defined structure with clear sections for parameters and return values, which makes them more detailed and technically precise. AI-generated documentation is usually shorter and good for general summaries, but it may not provide in-depth parameter explanations. AI is efficient for summarizing, while manual documentation offers stronger technical clarity.

Final Conclusion:

This lab demonstrated the role of AI in generating documentation and comments automatically. AI-assisted tools significantly reduce documentation effort and improve consistency.