

ASSIGNMENT- 9.3

Name: POOJITHA.EDDE

HT.No: 2303A51356

Batch: 20

Task 1: Basic Docstring Generation

Scenario

You are developing a utility function that processes numerical lists and must be properly documented for future maintenance.

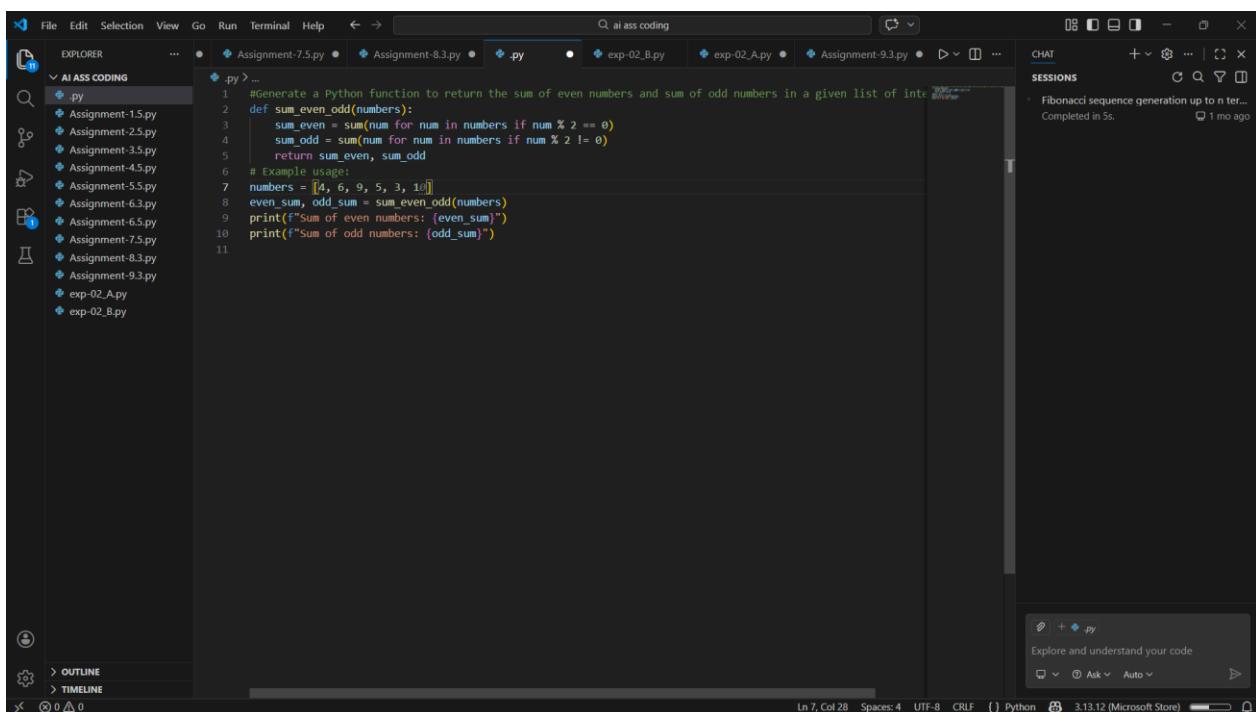
Requirements

- Write a Python function to return the sum of even numbers and sum of odd numbers in a given list
- Manually add a Google Style docstring to the function
- Use an AI-assisted tool (Copilot / Cursor AI) to generate a function-level docstring
- Compare the AI-generated docstring with the manually written docstring
- Analyze clarity, correctness, and completeness

Expected Output

- Python function with manual Google-style docstring
- AI-generated docstring for the same function
- Comparison explaining differences between manual and AI-generated documentation
- Improved understanding of AI-generated function-level documentation

CODE:



```
#Generate a Python function to return the sum of even numbers and sum of odd numbers in a given list of integers
def sum_even_odd(numbers):
    sum_even = sum(num for num in numbers if num % 2 == 0)
    sum_odd = sum(num for num in numbers if num % 2 != 0)
    return sum_even, sum_odd

# Example usage:
numbers = [4, 6, 9, 5, 3, 10]
even_sum, odd_sum = sum_even_odd(numbers)
print("Sum of even numbers: ", even_sum)
print("Sum of odd numbers: ", odd_sum)
```

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has a tree view under 'EXPLORER' labeled 'AI ASS CODING' with files like Assignment-7.5.py, Assignment-8.3.py, -py, exp-02_B.py, exp-02_A.py, Assignment-9.3.py, Assignment-6.3.py, Assignment-6.5.py, Assignment-7.5.py, Assignment-8.3.py, Assignment-9.3.py, exp-02_A.py, and exp-02_B.py. The main editor area contains the following Python code:

```
#Generate a docstring and inline comments using ai for the above function in a professional manner for code review.
def sum_even_odd(numbers):
    """
    Calculate the sum of even and odd numbers in a given list.

    Parameters:
    numbers (list): A list of integers to be processed.

    Returns:
    tuple: A tuple containing the sum of even numbers and the sum of odd numbers.
    """
    # Calculate the sum of even numbers using a generator expression
    sum_even = sum(num for num in numbers if num % 2 == 0)

    # Calculate the sum of odd numbers using a generator expression
    sum_odd = sum(num for num in numbers if num % 2 != 0)

    # Return the results as a tuple
    return sum_even, sum_odd

# Example usage:
numbers = [4, 6, 9, 5, 3, 1]
even_sum, odd_sum = sum_even_odd(numbers)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

The status bar at the bottom shows 'Ln 21, Col 28 Spaces:4 UTF-8 CRLF () Python 3.13.12 (Microsoft Store)'. The right sidebar includes a 'CHAT' section with a message about Fibonacci sequence generation and a 'SESSIONS' section.

OUTPUT:

The screenshot shows the Microsoft Visual Studio Code interface with the terminal tab active. The terminal window displays the following command and its output:

```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/.py"
Sum of even numbers: 10
Sum of odd numbers: 18
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
```

The status bar at the bottom shows 'Ln 7, Col 28 Spaces:4 UTF-8 CRLF () Python 3.13.12 (Microsoft Store)'. The right sidebar includes a 'CHAT' section with a message about Fibonacci sequence generation and a 'SESSIONS' section.

```

#Generate a docstring and inline comments using ai for the above function in a professional manner for code readability.
def sum_even_odd(numbers):
    """
    Calculate the sum of even and odd numbers in a given list.

    Parameters:
    numbers (list): A list of integers to be processed.

    Returns:
    tuple: A tuple containing the sum of even numbers and the sum of odd numbers.
    """
    # calculate the sum of even numbers using a generator expression
    sum_even = sum(num for num in numbers if num % 2 == 0)

    # calculate the sum of odd numbers using a generator expression
    sum_odd = sum(num for num in numbers if num % 2 != 0)

```

The screenshot shows the VS Code interface with the 'AI ASS CODING' extension active. The 'EXPLORER' sidebar shows several Python files. The main editor window displays the AI-generated docstring and inline comments for the `sum_even_odd` function. The 'TERMINAL' tab shows the command-line output of running the script, which prints the sum of even numbers (10) and the sum of odd numbers (18). The status bar at the bottom indicates the file is Python 3.13.12.

OBSERVATION:

The function correctly calculates the sum of even and odd numbers in a clean and efficient way. Without documentation, the logic is understandable but not immediately clear to someone new reading the code. The AI-generated docstring improves clarity by clearly explaining the input, output, and purpose of the function. The added inline comments also make the step-by-step logic easier to follow. Overall, AI documentation enhances readability and professionalism, but it should always be reviewed to ensure it fully matches the code.

Task 2: Automatic Inline Comments

Scenario

You are developing a student management module that must be easy to understand for new developers.

Requirements

- Write a Python program for an `sru_student` class with the following:
 - Attributes: `name`, `roll_no`, `hostel_status`
 - Methods: `fee_update()` and `display_details()`
- Manually write inline comments for each line or logical block
- Use an AI-assisted tool to automatically add inline comments
- Compare manual comments with AI-generated comments
- Identify missing, redundant, or incorrect AI comments

Expected Output

- Python class with manually written inline comments
- AI-generated inline comments added to the same code
- Comparative analysis of manual vs AI comments

- Critical discussion on strengths and limitations of AI-generated comments

CODE:

```

1 #generate a python program for an sru_student class with attributes : name , rollno,hostel_status and methods
2 class sru_student:
3     def __init__(self, name, rollno, hostel_status):
4         self.name = name
5         self.rollno = rollno
6         self.hostel_status = hostel_status
7         self.fee = 0
8     def fee_update(self, amount):
9         self.fee += amount
10    def display_details(self):
11        print("Name: " + self.name)
12        print("Roll No: " + str(self.rollno))
13        print("Hostel Status: " + self.hostel_status)
14        print("Fee: " + str(self.fee))
15    # Example usage
16 student1 = sru_student("poojitha", "1356", "Hosteller")
17 student1.fee_update(50000)
18 student1.display_details()
19

```



```

1 #using ai generate inline comments for each line of the above code for better understanding and readability.
2 class sru_student:
3     # Initialize the student object with name, roll number, hostel status, and fee
4     def __init__(self, name, rollno, hostel_status):
5         self.name = name
6         self.rollno = rollno
7         self.hostel_status = hostel_status
8         self.fee = 0 # initialize fee to 0
9     def fee_update(self, amount):
10        # Update the fee by adding the specified amount
11        self.fee += amount
12    def display_details(self):
13        # Display the details of the student
14        print("Name: " + self.name)
15        print("Roll No: " + str(self.rollno))
16        print("Hostel Status: " + self.hostel_status)
17        print("Fee: " + str(self.fee))
18    # Example usage
19 student1 = sru_student("poojitha", "1356", "Hosteller") # Create an instance of sru_student
20 student1.fee_update(50000) # Update the fee for the student
21 student1.display_details() # Display the details of the student
22

```

OUTPUT:

The screenshot shows two instances of Microsoft Visual Studio Code side-by-side, demonstrating AI-generated Python code for a `sru_student` class.

Top Window:

```

1 #generate a python program for an sru_student class with attributes : name , rollno,hostel_status and method fee_update
2 class sru_student:
3     def __init__(self, name, rollno, hostel_status):
4         self.name = name
5         self.rollno = rollno
6         self.hostel_status = hostel_status
7         self.fee = 0
8     def fee_update(self, amount):
9         self.fee += amount
10    def display_details(self):
11        print("Name: " + self.name)
12        print("Roll No: " + str(self.rollno))
13        print("Hostel Status: " + self.hostel_status)
14        print("Fee: " + str(self.fee))
15    # Example usage
16 student1 = sru_student("poojitha", "1356", "Hosteller")
17 student1.fee_update(50000)

```

Bottom Window:

```

1 #using ai generate inline comments for each line of the above code for better understanding and readability.
2 class sru_student:
3     # Initialize the student object with name, roll number, hostel status, and fee
4     def __init__(self, name, rollno, hostel_status):
5         self.name = name
6         self.rollno = rollno
7         self.hostel_status = hostel_status
8         self.fee = 0 # Initialize fee to 0
9     def fee_update(self, amount):
10        # Update the fee by adding the specified amount
11        self.fee += amount
12    def display_details(self):
13        # Display the details of the student
14        print("Name: " + self.name)
15        print("Roll No: " + str(self.rollno))
16        print("Hostel Status: " + self.hostel_status)
17        print("Fee: " + str(self.fee"))

```

Both windows show the same terminal output:

```

PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & c:/users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/users/pooji/OneDrive/Desktop/ai ass coding/.py"
Name: poojitha
Roll No: 1356
Hostel Status: Hosteller
Fee: 50000
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>

```

OBSERVATION :

The `sru_student` class is simple, organized, and easy to understand. The AI-generated inline comments explain the purpose of attributes and methods clearly, which makes the program more beginner-friendly. The comments help readers understand how the fee is updated and how details are displayed. However, some comments describe very obvious actions, which may not be necessary in professional code. Still, AI assistance improves clarity and makes the code easier to maintain.

Task 3: Module-Level and Function-Level Documentation

Scenario

You are building a small calculator module that will be shared across multiple projects and requires structured documentation.

Requirements

- Write a Python script containing 3–4 functions (e.g., add, subtract, multiply, divide) • Manually write NumPy Style docstrings for each function
- Use AI assistance to generate:
 - A module-level docstring
 - Individual function-level docstrings
- Compare AI-generated docstrings with manually written ones
- Evaluate documentation structure, accuracy, and readability

Expected Output

- Python script with manual NumPy-style docstrings
- AI-generated module-level and function-level documentation
- Comparison between AI-generated and manual documentation
- Clear understanding of structured documentation for multi-function scripts

CODE:

```
#Generate a python code containing 3-4 functions : add,subtract,multiply , divide
def add(a, b):
    """Return the sum of a and b."""
    return a + b

def subtract(a, b):
    """Return the difference of a and b."""
    return a - b

def multiply(a, b):
    """Return the product of a and b."""
    return a * b

def divide(a, b):
    """Return 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:
num1 = 56
num2 = 21
print(f"Addition: {add(num1, num2)}")      # Output: 77
print(f"Subtraction: {subtract(num1, num2)}") # Output: 35
print(f"Multiplication: {multiply(num1, num2)}") # Output: 1176
print(f"Division: {divide(num1, num2)}")      # Output: 2.6666666666666665
```

A screenshot of Microsoft Visual Studio Code interface. The left sidebar shows a tree view under 'EXPLORER' labeled 'AI ASS CODING' containing files like Assignment-7.5.py, Assignment-8.3.py, .py, exp-02_B.py, exp-02_A.py, Assignment-9.3.py, Assignment-1.5.py, Assignment-2.5.py, Assignment-3.5.py, Assignment-4.5.py, Assignment-5.5.py, Assignment-6.3.py, Assignment-6.5.py, Assignment-7.5.py, Assignment-8.3.py, Assignment-9.3.py, exp-02_A.py, and exp-02_B.py. The main editor area displays Python code for arithmetic operations:

```
26 def multiply(a, b):
27     """Return the product of a and b.
28
29     Parameters:
30         a (float): The first number.
31         b (float): The second number.
32
33     Returns:
34         float: The quotient of a and b.
35
36     Raises:
37         ValueError: If b is zero, as division by zero is undefined.
38
39     if b == 0:
40         raise ValueError("Cannot divide by zero.")
41     return a / b
42
43 # Example usage:
44 num1 = 56
45 num2 = 21
46 print(f"Addition: {add(num1, num2)}")      # Output: 77
47 print(f"Subtraction: {subtract(num1, num2)}") # Output: 35
48 print(f"Multiplication: {multiply(num1, num2)}") # Output: 1176
49
50 print(f"Division: {divide(num1, num2)}")      # Output: 2.6666666666666665
```

The status bar at the bottom indicates 'Ln 60, Col 75 Spaces:4 UTF-8 CRLF () Python 3.13.12 (Microsoft Store)'. The bottom right corner shows a small AI assistant window.

OUTPUT:

A screenshot of Microsoft Visual Studio Code interface, similar to the previous one but with a different file open. The left sidebar shows a tree view under 'EXPLORER' labeled 'AI ASS CODING' containing files like Assignment-7.5.py, Assignment-8.3.py, .py, exp-02_B.py, exp-02_A.py, Assignment-9.3.py, Assignment-1.5.py, Assignment-2.5.py, Assignment-3.5.py, Assignment-4.5.py, Assignment-5.5.py, Assignment-6.3.py, Assignment-6.5.py, Assignment-7.5.py, Assignment-8.3.py, Assignment-9.3.py, exp-02_A.py, and exp-02_B.py. The main editor area displays Python code for arithmetic operations:

```
1 #Generate a python code containing 3-4 functions : add,subtract,multiply , divide
2 def add(a, b):
3     """Return the sum of a and b."""
4     return a + b
5 def subtract(a, b):
6     """Return the difference of a and b."""
7     return a - b
8 def multiply(a, b):
9     """Return the product of a and b."""
10    return a * b
11 def divide(a, b):
12
13     """Return the quotient of a and b. Raises ValueError if b is zero."""
14     if b == 0:
15         raise ValueError("Cannot divide by zero.")
16     return a / b
17
18 # Example usage:
```

The terminal output window at the bottom shows the execution of the script:

```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & c:/users/pooji/appdata/local/microsoft/windowsapps/python3.13.exe "c:/users/pooji/0neDrive/Desktop/ai ass coding/.py"
Addition: 77
Subtraction: 35
Multiplication: 1176
Division: 2.6666666666666665
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
```

The status bar at the bottom indicates 'Ln 24, Col 75 Spaces:4 UTF-8 CRLF () Python 3.13.12 (Microsoft Store)'. The bottom right corner shows a small AI assistant window.

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- EXPLORER** view: Shows files in the "AI ASS CODING" folder, including Assignment-7.5.py, Assignment-8.3.py, -py, exp-02_A.py, exp-02_B.py, Assignment-9.3.py, Assignment-1.5.py, Assignment-2.5.py, Assignment-3.5.py, Assignment-4.5.py, Assignment-5.5.py, Assignment-6.3.py, Assignment-6.5.py, Assignment-7.5.py, Assignment-8.3.py, Assignment-9.3.py, exp-02_A.py, and exp-02_B.py.
- CODE EDITOR**: A Python script named "-py" is open, showing AI-generated docstrings for the `multiply` and `divide` functions. The `multiply` function's docstring includes parameters (a float), returns (float), and raises (ValueError if b is zero). The `divide` function's docstring includes parameters (a float, b float), returns (float), and raises (ValueError if b is zero).
- TERMINAL**: Displays command-line output from a Python environment, showing addition, multiplication, subtraction, and division operations.
- CHAT**: Shows a session titled "Fibonacci sequence generation up to n ter..." completed in 5s, 1 mo ago.
- SESSIONS**: Shows a single session entry.
- STATUS BAR**: Shows file path (C:\Users\pooji\OneDrive\Desktop\ai ass coding & c:/users/pooji/appdata/local/microsoft/windowsapps/python3.13.exe "c:/users/pooji/oneDrive/Desktop/ai ass coding/"), line (Ln 60), column (Col 75), spaces (Spaces:4), encoding (UTF-8), CRLF, Python 3.13.12 (Microsoft Store), and a battery icon.

OBSERVATION:

The calculator functions are logically correct and work as expected. In the original version, the documentation is brief and does not provide detailed information about parameters or exceptions. The AI-generated module-level and function-level docstrings add structure and clarity by explaining inputs, outputs, and error handling in a professional format. This makes the script more suitable for collaborative projects and real-world applications. Overall, AI improves documentation quality while keeping the program logic unchanged.