

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:

The screenshot shows a code editor window titled "Assignment_9.3". The left pane displays the Python file "Assignment_9.3.py" with the following content:

```
Assignment_9.3.py >_
# 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.
    """

The right pane shows the terminal output of the command "python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3\Assignment_9.3.py"". The output is:
```

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"
Sum of even numbers: 12
Sum of odd numbers: 9
PS E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3>

The screenshot shows a code editor with two tabs: "Assignment_9.3.py" and "Code". The "Assignment_9.3.py" tab contains Python code for calculating the sum of even and odd numbers in a list. The "Code" tab shows the output of running the script with the command "python -u e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3\Assignment_9.3.py". The output indicates that the sum of even numbers is 12 and the sum of odd numbers is 9.

```
Assignment_9.3.py > ...
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}")
```

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:

The screenshot shows a code editor window titled "Assignment_9.3.py". The code defines a class "sru_student" with methods for initializing student details, updating fee, and displaying student information. An example usage creates a student object "student1" and demonstrates the update and display methods. To the right of the code, the terminal output shows the execution of the script with input "Alice" and "SRU123", resulting in the output "Name: Alice" and "Hostel Status: Active".

```
File Edit Selection View Go Run Terminal Help < >
Assignment_9.3.py > ...
51
52 "# Task 2: Automatic Inline Comments"
53 "# Generate a Python class named sru_student with
54 # attributes name, roll_no, and hostel_status, and methods
55 # fee_update() and display_details(). Add inline comments
56 # automatically.
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
```

```
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:

A screenshot of a code editor window titled "Assignment_9.3.py". The code defines four functions: add, subtract, multiply, and divide. Each function has a docstring with parameters and returns descriptions. The "add" function is shown in detail:

```
74
75 "# Task 3: Module-Level and Function-Level Documentation"
76 # Generate a Python calculator module with functions add,
77 # subtract, multiply, and divide. Add NumPy-style docstrings
78 # manually and then generate module-level and function-level
79 # documentation using AI assistance.
80
81 def add(a, b):
82     """
83     Add two numbers.
84
85     Parameters
86     -----
87     a : int or float
88         The first number.
89     b : int or float
90         The second number.
91
92     Returns
93     -----
94     int or float
95         The sum of a and b.
96
97     """
98     return a + b
99
100 def subtract(a, b):
101     """
102     Subtract two numbers.
103     Parameters
104     -----
105     a : int or float
106         The first number.
107     b : int or float
108         The second number.
109
110     Returns
111     -----
112     int or float
113         The difference of a and b.
114
115     """
116     return a - b
117
118 def multiply(a, b):
119     """
120     Multiply two numbers.
121     Parameters
122     -----
123     a : int or float
124         The first number.
125     b : int or float
126         The second number.
127
128     Returns
129     -----
130     int or float
131         The product of a and b.
132
133     """
134     return a * b
```

A screenshot of a code editor window titled "Assignment_9.3.py". The code defines four functions: subtract, multiply, and divide. The "subtract" function is shown in detail:

```
94
95     def subtract(a, b):
96         """
97         Subtract two numbers.
98         Parameters
99         -----
100        a : int or float
101            The first number.
102        b : int or float
103            The second number.
104
105        Returns
106        -----
107        int or float
108            The difference of a and b.
109
110        """
111        return a - b
112
113    def multiply(a, b):
114        """
115        Multiply two numbers.
116        Parameters
117        -----
118        a : int or float
119            The first number.
120        b : int or float
121            The second number.
122
123        Returns
124        -----
125        int or float
126            The product of a and b.
127
128        """
129        return a * b
130
131    def divide(a, b):
132        """
133        Divide two numbers.
134        Parameters
135        -----
136        a : int or float
137            The dividend.
138        b : int or float
139            The divisor.
140
141        Returns
142        -----
143        float
144            The quotient of a and b.
145
146        """
147        if b == 0:
148            raise ValueError("Division by zero is not allowed")
149        else:
150            return a / b
```

The screenshot shows a code editor interface with two panes. The left pane displays a Python file named 'Assignment_9.3.py' containing a function definition for 'divide'. The right pane shows a terminal window with the command 'python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_9.3\Assignment_9.3.py"' running, which outputs the generated documentation. The documentation includes detailed descriptions for parameters 'a' and 'b', a return value 'int or float', and a note about raising a ValueError if 'b' is zero.

```
124 def divide(a, b):
125     """
126     Divide two numbers.
127     Parameters
128     -----
129     a : int or float
130         The first number.
131     b : int or float
132         The second number.
133     Returns
134     -----
135     int or float
136         The quotient of a and b.
137     Raises
138     -----
139     ValueError
140         If b is zero.
141     """
142     if b == 0:
143         raise ValueError("Cannot divide by zero.")
144     return a / b
145 # Example usage
146 print(add(10, 5))      # Output: 15
147 print(subtract(10, 5))  # Output: 5
148 print(multiply(10, 5))  # Output: 50
149 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 Assisted 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.