

JSCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:M. Tech		Assignment Type: Lab	AcademicYear:2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
rCourseCode	24CS002PC215	CourseTitle	AI Assisted Problem Solving Using Python
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	10.11.2025	Time(s)	
Duration	2 Hours	Applicable to Batches	
AssignmentNumber:9.3(Present assignment number)/24(Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	<p>Lab 9: Documentation Generation: Automatic documentation and code comments</p> <p><b>Lab Objectives:</b></p> <ul style="list-style-type: none"> <li>To understand the importance of documentation and code comments in software development.</li> <li>To explore how AI-assisted coding tools can generate meaningful documentation and inline comments.</li> <li>To practice generating function-level and module-level docstrings automatically.</li> <li>To evaluate the quality, accuracy, and limitations of AI-generated documentation.</li> <li>To develop a small automated tool for documentation generation in Python..</li> </ul> <p><b>Lab Outcomes (LOs):</b> After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> <li>Apply AI-assisted coding tools to generate docstrings and inline comments for Python code.</li> <li>Critically analyze AI-generated documentation for correctness,</li> </ul>		Week4 - Wednesda y

	<p>completeness, and readability.</p> <ul style="list-style-type: none"> <li>• Create structured documentation (function-level, module-level) following standard formats.</li> <li>• Design and implement a mini documentation generator tool to automate code commenting and docstring creation.</li> </ul> <p><b>Task Description#1 Basic Docstring Generation</b></p> <ul style="list-style-type: none"> <li>• Write python function to return sum of even and odd numbers in the given list.</li> <li>• Incorporate manual <b>docstring</b> in code with Google Style</li> <li>• Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.</li> <li>• Compare the AI-generated docstring with your manually written one.</li> </ul> <p><b>Expected Outcome#1:</b> Students understand how AI can produce function-level documentation.</p> <p><b>Task Description#2 Automatic Inline Comments</b></p> <ul style="list-style-type: none"> <li>• Write python program for <b>sru_student</b> class with attributes like name, roll no., hostel_status and <b>fee_update</b> method and <b>display_details</b> method.</li> <li>• Write comments manually for each line/code block</li> <li>• Ask an AI tool to add inline comments explaining each line/step.</li> <li>• Compare the AI-generated comments with your manually written one.</li> </ul> <p><b>Expected Output#2:</b> Students critically analyze AI-generated code comments.</p> <p><b>Task Description#3</b></p> <ul style="list-style-type: none"> <li>• Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).</li> <li>• Incorporate manual <b>docstring</b> in code with NumPy Style</li> <li>• Use AI assistance to generate a module-level docstring + individual function docstrings.</li> <li>• Compare the AI-generated docstring with your manually written one.</li> </ul> <p><b>Expected Output#3:</b> Students learn structured documentation for multi-function scripts</p> <p><b>Push documentation whole workspace as .md file in GitHub</b></p>	
--	--	--

	<b>Repository</b>  <b>Note: Report should be submitted a word document for all tasks in a single document with prompts, comments &amp; code explanation, and output and if required, screenshots</b>	
--	--	--

## Task Description#1 Basic Docstring Generation

- Write python function to return sum of even and odd numbers in the given list.
- Incorporate manual **docstring** in code with Google Style
- Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.
- Compare the AI-generated docstring with your manually written one.

**PROMPT:** Write a Python function to return the sum of even and odd numbers in a list. Add a **manual docstring** in Google Style, then use AI to generate its own docstring and compare both.

### Manual docstring in Google Style:

```
1 def sum_even_odd_manual(numbers):
2     """
3     Calculates the sum of even and odd numbers in a given list.
4
5     Args:
6     |     numbers (list): A list of integers entered by the user.
7
8     Returns:
9     |     tuple: A tuple containing two integers:
10    |         - The first value is the sum of even numbers.
11    |         - The second value is the sum of odd numbers.
12
13    Example:
14    |     >>> sum_even_odd_manual([1, 2, 3, 4, 5])
15    |         (6, 9)
16    """
17    even_sum = 0
18    odd_sum = 0
19
20    for num in numbers:
21        if num % 2 == 0:
22            even_sum += num
23        else:
24            odd_sum += num
25
26    return even_sum, odd_sum
27
28
29 # --- User Input Section ---
30 numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
```

```

24         odd_sum += num
25
26     return even_sum, odd_sum
27
28
29 # --- User Input Section ---
30 numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
31 even_sum, odd_sum = sum_even_odd_manual(numbers)
32 print("Manual Docstring Function Result:")
33 print("Sum of Even Numbers:", even_sum)
34 print("Sum of Odd Numbers:", odd_sum)
35

```

### AI-Generated Docstring Version:

```

SK 1.py  TASK 1.1.py  def sum_even_odd_ai(numbers): Untitled-1  TASK 3.py
def sum_even_odd_ai(numbers):
    """
    Returns the sum of even and odd numbers from the given list.

    Parameters:
        numbers (list): List of integers.

    Returns:
        tuple: (sum_of_even, sum_of_odd)
    """
    even_sum = sum(num for num in numbers if num % 2 == 0)
    odd_sum = sum(num for num in numbers if num % 2 != 0)
    return even_sum, odd_sum

# --- User Input Section ---
numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
even_sum, odd_sum = sum_even_odd_ai(numbers)
print("AI-Generated Docstring Function Result:")
print("Sum of Even Numbers:", even_sum)
print("Sum of Odd Numbers:", odd_sum)

```

**Expected Outcome#1:** Students understand how AI can produce function-level documentation.

**Practical output:**

**Manual docstring in Google Style:**

```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"
Enter numbers separated by spaces: 1 2 3 4 5 6 7 8
Manual Docstring Function Result:
Sum of Even Numbers: 20
Sum of Odd Numbers: 16
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>
```

**AI-Generated Docstring Version:**

```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"
Enter numbers separated by spaces: 1 2 3 4 5 6 7 8 9 2
AI-Generated Docstring Function Result:
Sum of Even Numbers: 22
Sum of Odd Numbers: 25
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>
```

### Comparison Table

Aspect	Manual Docstring (Google Style)	AI-Generated Docstring
Structure	Follows Google Style (Args, Returns, Example)	Generic short form (Parameters, Returns)
Detail Level	More descriptive and clear for beginners	Concise and minimal
Clarity	Explains every parameter and output explicitly	Summarizes in short
Professional Use	Best for documentation and learning	Quick and efficient for developers
Example Included	Yes	No

### Task Description#2 Automatic Inline Comments

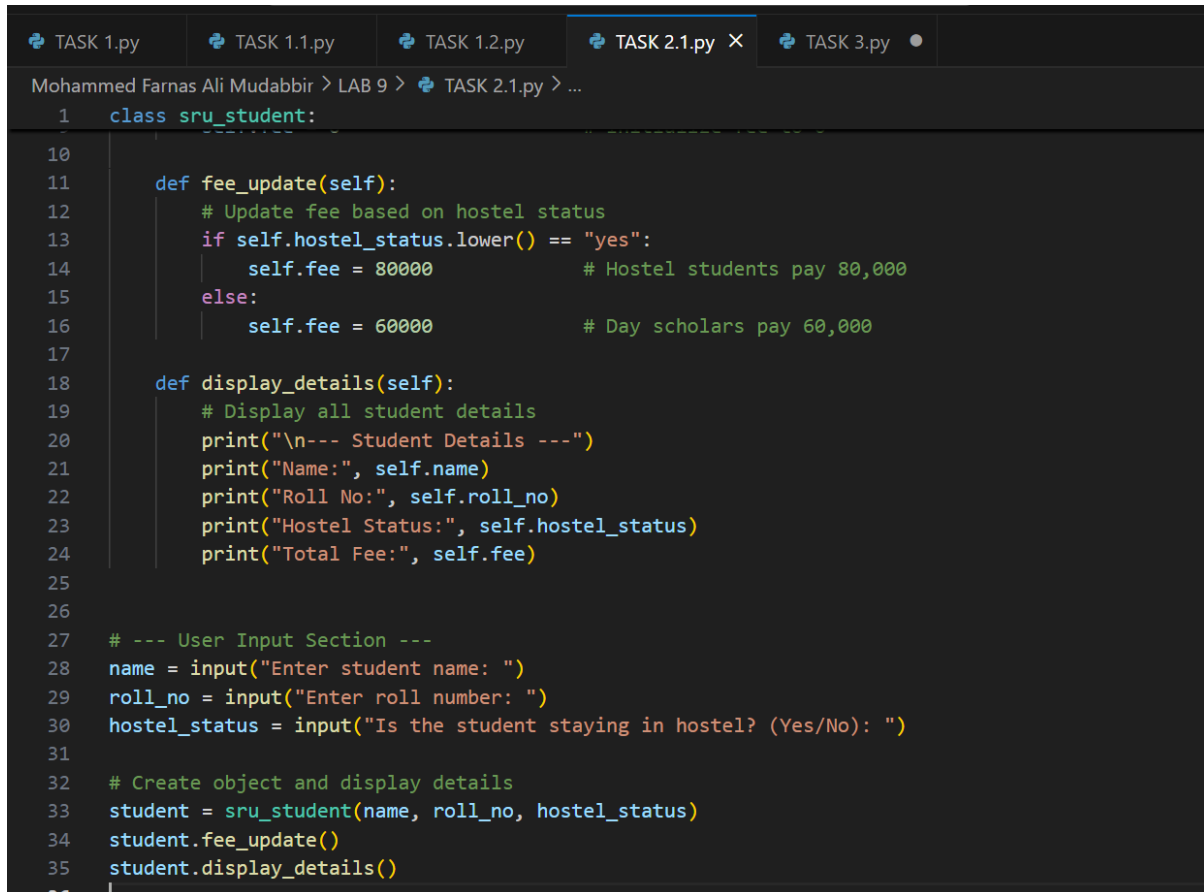
- Write python program for **sru\_student** class with attributes like name, roll no., hostel\_status and **fee\_update** method and **display\_details** method.

- Write comments manually for each line/code block
- Ask an AI tool to add inline comments explaining each line/step.

Compare the AI-generated comments with your manually written one.

**PROMPT:** Write a Python program for an `sru_student` class with attributes `name`, `roll_no`, and `hostel_status`, including methods `fee_update()` and `display_details()`. Add manual comments for each line, then let AI generate inline comments and compare both.

### Manual docstring in Google Style:



```

1  class sru_student:
2      """
3      A class representing a student in a school.
4      Attributes:
5      - name: Student's name (str)
6      - roll_no: Student's roll number (int)
7      - hostel_status: Whether the student stays in hostel (bool)
8      - fee: Student's fee (int)
9      """
10
11     def fee_update(self):
12         """
13         Update fee based on hostel status
14         """
15         if self.hostel_status.lower() == "yes":
16             self.fee = 80000 # Hostel students pay 80,000
17         else:
18             self.fee = 60000 # Day scholars pay 60,000
19
20     def display_details(self):
21         """
22         Display all student details
23         """
24         print("\n--- Student Details ---")
25         print("Name:", self.name)
26         print("Roll No:", self.roll_no)
27         print("Hostel Status:", self.hostel_status)
28         print("Total Fee:", self.fee)
29
30     # --- User Input Section ---
31     name = input("Enter student name: ")
32     roll_no = input("Enter roll number: ")
33     hostel_status = input("Is the student staying in hostel? (Yes/No): ")
34
35     # Create object and display details
36     student = sru_student(name, roll_no, hostel_status)
37     student.fee_update()
38     student.display_details()

```

### AI-Generated Docstring Version:

```
SK 4.py TASK 5.py TASK 1.1 MANUAL.py TASK 1.2 AI.py TASK 2.1 MANUAL.py class sru_
1 class sru_student:
11     def fee_update(self):
13         if self.hostel_status.lower() == "yes":
14             self.fee = 80000 # Set fee to 80000 for hostel students
15         else:
16             self.fee = 60000 # Set fee to 60000 for non-hostel students
17
18     def display_details(self):
19         # Print student details and total fee
20         print("\n--- Student Details ---")
21         print("Name:", self.name)
22         print("Roll No:", self.roll_no)
23         print("Hostel Status:", self.hostel_status)
24         print("Total Fee:", self.fee)
25
26
27 # Take input from user for student details
28 name = input("Enter student name: ")
29 roll_no = input("Enter roll number: ")
30 hostel_status = input("Is the student staying in hostel? (Yes/No): ")
31
32 # Create a student object and display fee details
33 student = sru_student(name, roll_no, hostel_status)
34 student.fee_update()
35 student.display_details()
```

**Expected Output#2:** Students critically analyze AI-generated code comments.

**Practical output:**

**Manual docstring in Google Style:**

```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\One
9 assessment\TASK 2.1 MANUAL.py"
Enter student name: RAHIMATHUNNISA RIMSHA
Enter roll number: 2205A41L24
Is the student staying in hostel? (Yes/No): NO

--- Student Details ---
Name: RAHIMATHUNNISA RIMSHA
Roll No: 2205A41L24
Hostel Status: NO
Total Fee: 60000
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> |
```





AI-Generated Docstring Version:

```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\
tempCodeRunnerFile.python"
Enter student name: RIMSHA
Enter roll number: 2932423536F3
Is the student staying in hostel? (Yes/No): YES

--- Student Details ---
Name: RIMSHA
Roll No: 2932423536F3
Hostel Status: YES
Total Fee: 80000
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>
```

⚡ Manual vs AI-Generated Comment Comparison

Aspect	Manual Comments (Written by Student)	AI-Generated Comments (From Copilot/Cursor AI) 
Style	Simple, clear, and descriptive	Concise, professional, and formal
Purpose	Explains <b>why</b> each line is written and what it means	Explains <b>what</b> each line of code does
Tone	Friendly and educational (beginner-friendly)	Technical and developer-focused
Detail Level	More detailed — focuses on logic and reasoning behind code	Brief — focuses only on the functionality of code
Example	"# Update fee based on hostel 	"# Check if student is a hostel

Example	"# Update fee based on hostel status" → tells the purpose of that step	"# Check if student is a hostel resident and update fee accordingly" → states the direct action
Readability	Easier for beginners to follow and understand logic	More suitable for experienced programmers
Depth of Explanation	Explains initialization, logic, and output purpose	Describes code behavior without deeper reasoning
Usefulness in Learning	✅ Excellent for understanding flow and concept	⚙️ Good for documentation and quick code review
Accuracy	Human-level clarity and contextual understanding	AI may sometimes give generic or repeated phrasing
Manual Comment	AI-Generated Comment	
# Initialize student attributes	# Initialize the student object with name, roll number, and hostel status	
# Update fee based on hostel status	# Check if student is a hostel resident and update fee accordingly	
# Display all student details	# Print student details and total fee	

### Task Description#3

- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual **docstring** in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function docstrings.
- Compare the AI-generated docstring with your manually written one.

**PROMPT:** Write a Python script with 4 functions — add, subtract, multiply, and divide. Add manual NumPy-style docstrings for each function. Then use AI to generate module-level and function docstrings, and compare both.

## Manual docstring in Google Style:

```
TASK 5.py TASK 1.1 MANUAL.py TASK 1.2 AI.py TASK 2.1 MANUAL.py TASK 2.2 AI.py TASK 3.1 MANUAL.py > ...
rimsha python assessment > lab 9 assessment > TASK 3.1 MANUAL.py > ...
80 def divide(a, b):
96     """
97     Raises
98     -----
99     ValueError
100     |     If b is zero.
101
102     Example
103     -----
104     >>> divide(10, 2)
105     5
106     """
107     if b == 0:
108         raise ValueError("Cannot divide by zero.")
109     return a / b
110
111 # --- User Input Section ---
112 print("Simple Calculator")
113 num1 = float(input("Enter first number: "))
114 num2 = float(input("Enter second number: "))
115
116 print("Addition:", add(num1, num2))
117 print("Subtraction:", subtract(num1, num2))
118 print("Multiplication:", multiply(num1, num2))
119 print("Division:", divide(num1, num2))]
```

```
TASK 1.py × TASK 1.1.py TASK 1.2.py TASK 2.1.py TASK 2.2.py Untitled-1 TASK 3.py

31
32 def subtract(a, b):
33     """
34     Subtract two numbers.
35
36     Parameters
37     -----
38     a : float
39     | First number.
40     b : float
41     | Second number to subtract from the first.
42
43     Returns
44     -----
45     float
46     | Result of a - b.
47
48     Example
49     -----
50     >>> subtract(10, 4)
51     6
52     """
53     return a - b
54
55
56 def multiply(a, b):
57     """
58     Multiply two numbers.
59
60     Parameters
61     -----
62     a : float
63     | First number.
64     b : float
65     | Second number.
66
67     Returns
68     -----
69     float
70     | Product of a and b.
71
72     Example
73     -----
74     >>> multiply(2, 5)
75     10
76     """
77     return a * b
78
79
80 def divide(a, b):
81     """
82     Divide two numbers.
83
84     Parameters
85     -----
86     a : float
87     | Numerator.
88     b : float
```

```

80  def divide(a, b):
88      b : float
89      |
90      Denominator.
91
92      Returns
93      -----
94      float
95      |
96      Result of division (a / b).
97
98      Raises
99      -----
100     ValueError
101     |
102     If b is zero.
103
104     Example
105     -----
106     >>> divide(10, 2)
107     5
108     """
109     if b == 0:
110         raise ValueError("Cannot divide by zero.")
111     return a / b
112
113 # --- User Input Section ---
114 print("Simple Calculator")
115 num1 = float(input("Enter first number: "))
116 num2 = float(input("Enter second number: "))
117
118 # --- User Input Section ---
119 print("Simple Calculator")
120 num1 = float(input("Enter first number: "))
121 num2 = float(input("Enter second number: "))
122
123 print("Addition:", add(num1, num2))
124 print("Subtraction:", subtract(num1, num2))
125 print("Multiplication:", multiply(num1, num2))
126 print("Division:", divide(num1, num2))

```

## AI-Generated Docstring Version:

```
TASK 1.py × TASK 1.1.py TASK 1.2.py TASK 2.1.py TASK 2.2.py TASK 3.1.
Mohammed Farnas Ali Mudabbir > LAB 9 > TASK 3.2.py > ...
1  """
2  A basic calculator module that provides functions for addition, subtraction,
3  multiplication, and division of two numbers.
4  """
5
6  def add(a, b):
7      """Returns the sum of two numbers."""
8      return a + b
9
10
11 def subtract(a, b):
12     """Returns the result of subtracting b from a."""
13     return a - b
14
15
16 def multiply(a, b):
17     """Returns the product of two numbers."""
18     return a * b
19
20
21 def divide(a, b):
22     """Returns the quotient of a divided by b. Raises an error if b is zero."""
23     if b == 0:
24         raise ValueError("Cannot divide by zero.")
25     return a / b
26
27
28 # --- User Input Section ---
29 print("Simple Calculator (AI-Generated Docstring Version)")
```

```
Mohammed Farnas Ali Mudabbir > LAB 9 > TASK 3.2.py > ...
21  def divide(a, b):
24      raise ValueError("Cannot divide by zero.")
25      return a / b
26
27
28  # --- User Input Section ---
29  print("Simple Calculator (AI-Generated Docstring Version)")
30  num1 = float(input("Enter first number: "))
31  num2 = float(input("Enter second number: "))
32
33  print("\nResults:")
34  print("Addition:", add(num1, num2))
35  print("Subtraction:", subtract(num1, num2))
36  print("Multiplication:", multiply(num1, num2))
37
38  try:
39      print("Division:", divide(num1, num2))
40  except ValueError as e:
41      print("Error:", e)
42
```

**Expected Output#3:** Students learn structured documentation for multi-function scripts

**Practical output:**

**Manual docstring in Google Style:**

```
> v TERMINAL Code - rimsh
● PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\
9 assessment\TASK 3.1 MANUAL.py"
Simple Calculator
Enter first number: 2
Enter second number: 9
Addition: 11.0
Subtraction: -7.0
Multiplication: 18.0
Division: 0.2222222222222222
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> |
```

### AI-Generated Docstring Version:

```
● PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\De
tempCodeRunnerFile.python"
Simple Calculator (AI-Generated Docstring Version)
Enter first number: 6
Enter second number: 8

Results:
Addition: 14.0
Subtraction: -2.0
Multiplication: 48.0
Division: 0.75
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> |
```





## Comparison: Manual vs AI-Generated Docstrings

Aspect	Manual NumPy-Style (Student)	AI-Generated (Copilot/Cursor AI)
Structure	Detailed NumPy style with <code>Parameters</code> , <code>Returns</code> , <code>Example</code>	Short single-line description
Module Docstring	Explains purpose and structure	Briefly summarizes functionality
Detail Level	Highly descriptive and formatted	Concise and to the point
Clarity	Excellent for learners and documentation	Suitable for professional quick reference
Examples Included	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> No