

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName: B. Tech		Assignment Type: Lab	AcademicYear:2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator)	
		Dr. T. Sampath Kumar	
		Dr. Pramoda Patro	
		Dr. Brij Kishor Tiwari	
		Dr.J.Ravichander	
		Dr. Mohammand Ali Shaik	
		Dr. Anirodh Kumar	
		Mr. S.Naresh Kumar	
		Dr. RAJESH VELPULA	
		Mr. Kundhan Kumar	
		Ms. Ch.Rajitha	
		Mr. M Prakash	
		Mr. B.Raju	
		Intern 1 (Dharma teja)	
		Intern 2 (Sai Prasad)	
		Intern 3 (Sowmya)	
		NS_2 (Mounika)	
CourseCode	24CS002PC215	CourseTitle	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week4 - Wednesday	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	Lab 8: Documentation Generation: Automatic documentation and code comments Lab Objectives: <ul style="list-style-type: none"> To understand the importance of documentation and code comments in software development. To explore how AI-assisted coding tools can generate meaningful documentation and 	Week4 - Wednesday	

	<p>inline comments.</p> <ul style="list-style-type: none"> • To practice generating function-level and module-level docstrings automatically. • To evaluate the quality, accuracy, and limitations of AI-generated documentation. • To develop a small automated tool for documentation generation in Python.. <p>Lab Outcomes (LOs): After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • Apply AI-assisted coding tools to generate docstrings and inline comments for Python code. • Critically analyze AI-generated documentation for correctness, completeness, and readability. • Create structured documentation (function-level, module-level) following standard formats. • Design and implement a mini documentation generator tool to automate code commenting and docstring creation. <p>Task Description#1 Basic Docstring Generation</p> <ul style="list-style-type: none"> • 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. <p>Expected Outcome#1: Students understand how AI can produce function-level documentation.</p> <p>Task Description#2 Automatic Inline Comments</p> <ul style="list-style-type: none"> • 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. <p>Expected Output#2: Students critically analyze AI-generated code comments.</p> <p>Task Description#3</p> <ul style="list-style-type: none"> • 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. <p>Expected Output#3: Students learn structured documentation for multi-function scripts</p> <p>Push documentation whole workspace as .md file in GitHub Repository</p> <p>Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots</p>	
--	--	--

TASK 1:
CODE BY USER:

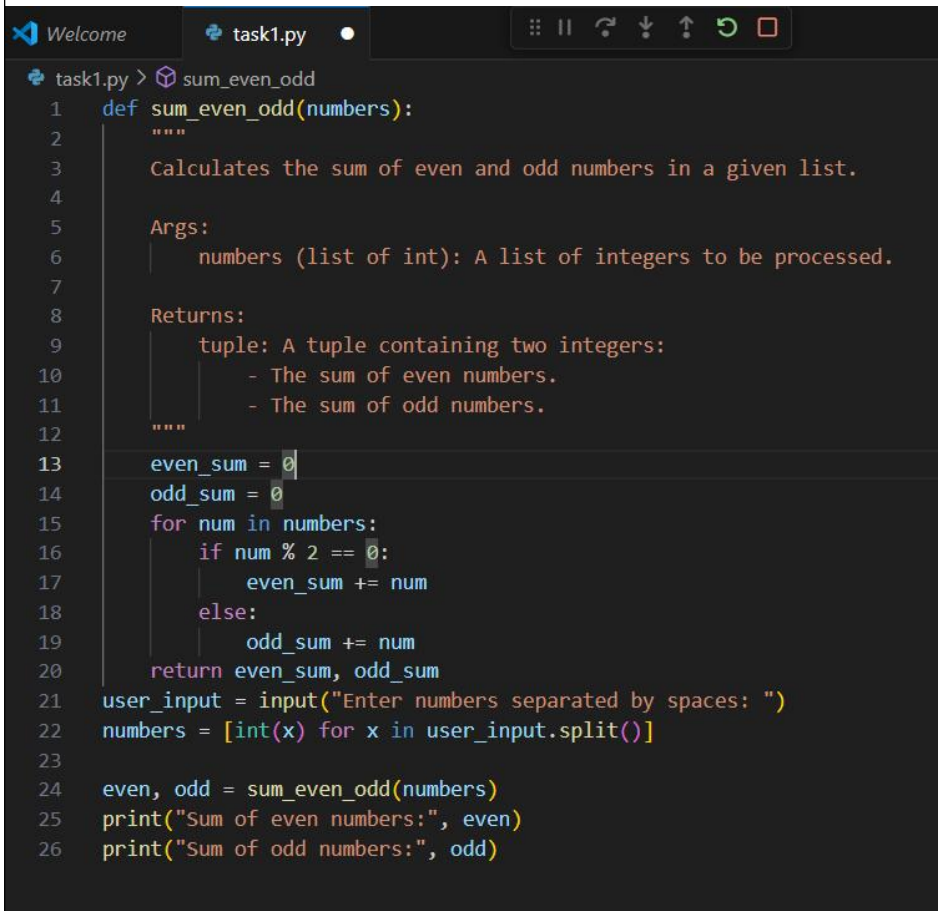
```
Welcome task1.py X
task1.py > ...
1 def sum_even_odd(numbers):
2     even_sum = 0
3     odd_sum = 0
4     for num in numbers:
5         if num % 2 == 0:
6             even_sum += num
7         else:
8             odd_sum += num
9     return even_sum, odd_sum
10 user_input = input("Enter numbers separated by spaces: ")
11 numbers = [int(x) for x in user_input.split()]
12
13 even, odd = sum_even_odd(numbers)
14 print("Sum of even numbers:", even)
15 print("Sum of odd numbers:", odd)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai> & 'c:\Users\Anusha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Anusha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\python\python.exe' 'c:\Users\Anusha\OneDrive\Desktop\AI\ai\task1.py'
Enter numbers separated by spaces: 2 5 8 3 7
Sum of even numbers: 10
Sum of odd numbers: 15
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai> ^C
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai> c:: cd 'c:\Users\Anusha\OneDrive\Desktop\AI\ai' & python.exe 'c:\Users\Anusha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\python\python.exe' 'c:\Users\Anusha\OneDrive\Desktop\AI\ai\task1.py'
```

MANUAL DOCSTRING CODE :

```
Welcome task1.py X
task1.py > sum_even_odd
1 def sum_even_odd(numbers):
2     """
3     This function adds up all the even numbers and all the odd numbers in the list.
4     You give it a list of numbers, and it tells you the total of the evens and the total of the odds.
5     Returns both sums as a pair.
6     """
7     even_sum = 0
8     odd_sum = 0
9     for num in numbers:
10         if num % 2 == 0:
11             even_sum += num
12         else:
13             odd_sum += num
14     return even_sum, odd_sum
15 user_input = input("Enter numbers separated by spaces: ")
16 numbers = [int(x) for x in user_input.split()]
17
18 even, odd = sum_even_odd(numbers)
19 print("Sum of even numbers:", even)
20 print("Sum of odd numbers:", odd)
```

DOCSTRING IN CODE WITH GOOGLE STYLE :



```

1  def sum_even_odd(numbers):
2      """
3      Calculates the sum of even and odd numbers in a given list.
4
5      Args:
6      |   numbers (list of int): A list of integers to be processed.
7
8      Returns:
9      |   tuple: A tuple containing two integers:
10     |       - The sum of even numbers.
11     |       - The sum of odd numbers.
12     """
13     even_sum = 0
14     odd_sum = 0
15     for num in numbers:
16         if num % 2 == 0:
17             even_sum += num
18         else:
19             odd_sum += num
20     return even_sum, odd_sum
21 user_input = input("Enter numbers separated by spaces: ")
22 numbers = [int(x) for x in user_input.split()]
23
24 even, odd = sum_even_odd(numbers)
25 print("Sum of even numbers:", even)
26 print("Sum of odd numbers:", odd)

```

COMPARISION : The user docstring is informal, simple and easy to understand to a friend where as ai docstring is formal, structured and formatting (like args and returns).

TASK 2:
CODE BY USER:

```

Welcome task1.py task2.py X
task2.py > ...
1 class sru_student:
2     def __init__(self, name, roll_no, hostel_status):
3         self.name = name
4         self.roll_no = roll_no
5         self.hostel_status = hostel_status
6         self.fee_paid = False
7
8     def fee_update(self, status):
9         self.fee_paid = status
10
11    def display_details(self):
12        print("Name:", self.name)
13        print("Roll No:", self.roll_no)
14        print("Hostel Status:", self.hostel_status)
15        print("Fee Paid:", "Yes" if self.fee_paid else "No")
16
17    name = input("Enter student name: ")
18    roll_no = input("Enter roll number: ")
19    hostel_status = input("Hostel status (Yes/No): ")
20
21    student = sru_student(name, roll_no, hostel_status)
22
23    fee_status = input("Has the fee been paid? (Yes/No): ")
24    student.fee_update(fee_status.lower() == "yes")
25
26    student.display_details()

```

OUTPUT:

```

'c:\Users\Anusha\OneDrive\Desktop\AI\ai\task2.py'
Enter student name: bhavana
Enter roll number: 2403a52249
Hostel status (Yes/No): yes
Has the fee been paid? (Yes/No): yes
Name: bhavana
Roll No: 2403a52249
Hostel Status: yes
Fee Paid: Yes
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>

```

MANUAL COMMENTS:

```
Welcome task1.py task2.py
task2.py > sru_student
1 class sru_student:
2     # This sets up the student with their name, roll number, hostel status, and fee status
3     def __init__(self, name, roll_no, hostel_status):
4         self.name = name
5         self.roll_no = roll_no
6         self.hostel_status = hostel_status
7         self.fee_paid = False
8
9     # This method updates whether the fee is paid or not
10    def fee_update(self, status):
11        self.fee_paid = status
12
13    # This method prints out all the details of the student
14    def display_details(self):
15        print("Name:", self.name)
16        print("Roll No:", self.roll_no)
17        print("Hostel Status:", self.hostel_status)
18        print("Fee Paid:", "Yes" if self.fee_paid else "No")
19
20    # Ask the user for student details
21    name = input("Enter student name: ")
22    roll_no = input("Enter roll number: ")
23    hostel_status = input("Hostel status (Yes/No): ")
24
25    # Make a student object with the details
26    student = sru_student(name, roll_no, hostel_status)
27
28    # Ask if the fee is paid and update it
29    fee_status = input("Has the fee been paid? (Yes/No): ")
30    student.fee_update(fee_status.lower() == "yes")
31
32    # Show all the details
33    student.display_details()
34    name = input("Enter student name: ")
35    roll_no = input("Enter roll number: ")
36    hostel_status = input("Hostel status (Yes/No): ")
37
38    student = sru_student(name, roll_no, hostel_status)
39
40    fee_status = input("Has the fee been paid? (Yes/No): ")
41    student.fee_update(fee_status.lower() == "yes")
42
43    student.display_details()
```

INLINE COMMENTS EXPLAINED BY THE AI :


```

Welcome task1.py task2.py
task2.py > sru_student > _init_
1 class sru_student:
2     """
3     A class representing a student at SRU with attributes for name, roll number, hostel status, and fee payment status.
4     Attributes:
5         name (str): The name of the student.
6         roll_no (str or int): The roll number of the student.
7         hostel_status (str): The hostel accommodation status of the student.
8         fee_paid (bool): Indicates whether the student's fee has been paid.
9     Methods:
10         _init_(name, roll_no, hostel_status):
11             Initializes a new SRU student with the given name, roll number, and hostel status. Fee status is set to unpaid by default.
12         fee_update(status):
13             Updates the fee payment status of the student.
14         display_details():
15             Prints all details of the student including name, roll number, hostel status, and fee payment status.
16     """
17     # This sets up the student with their name, roll number, hostel status, and fee status
18     def _init_(self, name, roll_no, hostel_status):
19         self.name = name
20         self.roll_no = roll_no
21         self.hostel_status = hostel_status
22         self.fee_paid = False
23
24     # This method updates whether the fee is paid or not
25     def fee_update(self, status):
26         self.fee_paid = status
27
28     # This method prints out all the details of the student
29     def display_details(self):
30         print("Name:", self.name)
31         print("Roll No:", self.roll_no)
32         print("Hostel Status:", self.hostel_status)
33         print("Fee Paid:", "Yes" if self.fee_paid else "No")
34
35     # Ask the user for student details
36     name = input("Enter student name: ")
37     roll_no = input("Enter roll number: ")
38     hostel_status = input("Hostel status (Yes/No): ")
39
40     # Make a student object with the details
41     student = sru_student(name, roll_no, hostel_status)
42
43     # Ask if the fee is paid and update it
44     fee_status = input("Has the fee been paid? (Yes/No): ")
45     student.fee_update(fee_status.lower() == "yes")
46
47     # Show all the details
48     student.display_details()
49     name = input("Enter student name: ")
50     roll_no = input("Enter roll number: ")
51     hostel_status = input("Hostel status (Yes/No): ")
52
53     student = sru_student(name, roll_no, hostel_status)
54
55     fee_status = input("Has the fee been paid? (Yes/No): ")
56     student.fee_update(fee_status.lower() == "yes")
57
58     student.display_details()

```

COMPARISION : The user docstring is informal, simple and easy to understand to a friend where as ai docstring is formal, structured and suitable for technical documentation.

TASK 3:
CODE BY USER :

```
Welcome task1.py task2.py task3.py X
task3.py > ...
1  def add(a, b):
2      return a + b
3
4  def subtract(a, b):
5      return a - b
6
7  def multiply(a, b):
8      return a * b
9
10 def divide(a, b):
11     if b == 0:
12         return "Cannot divide by zero"
13     return a / b
14 x = float(input("Enter first number: "))
15 y = float(input("Enter second number: "))
16
17 op = input("Choose operation (+, -, *, /): ")
18
19 if op == "+":
20     print("Result:", add(x, y))
21 elif op == "-":
22     print("Result:", subtract(x, y))
23 elif op == "*":
24     print("Result:", multiply(x, y))
25 elif op == "/":
26     print("Result:", divide(x, y))
27 else:
28     print("Invalid operation")

PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai> c::; cd
ms\Python\Python313\python.exe' 'c:\Users\Anusha\
her' '54038' '--' 'c:\Users\Anusha\OneDrive\Desкто
Enter first number: 10
Enter second number: 5
Choose operation (+, -, *, /): +
Result: 15.0
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>
```

MANUAL DOCSTRING WITH CODE IN NUMPY STYLE:


```

task3.py > ...
1  """
2  Simple calculator for two numbers.
3  """
4
5  def add(a, b):
6      """Add two numbers."""
7      return a + b
8
9  def subtract(a, b):
10     """Subtract b from a."""
11     return a - b
12
13  def multiply(a, b):
14     """Multiply two numbers."""
15     return a * b
16
17  def divide(a, b):
18     """Divide a by b. Returns error if b is zero."""
19     if b == 0:
20         return "Cannot divide by zero"
21     return a / b
22
23  x = float(input("Enter first number: "))
24  y = float(input("Enter second number: "))
25  op = input("Choose operation (+, -, *, /): ")
26
27  if op == "+":
28      print("Result:", add(x, y))
29  elif op == "-":
30      print("Result:", subtract(x, y))
31  elif op == "*":
32      print("Result:", multiply(x, y))
33  elif op == "/":
34      print("Result:", divide(x, y))
35  else:
36      print("Invalid operation")

```

GENERATED BY A MODULE-LEVEL DOCSTRING+INDIVIDUAL FUNCTION :

```

Welcome task1.py task2.py task3.py X
task3.py > ...
1  """
2  Calculator for two numbers.
3  """
4
5  def add(a, b):
6      """Add two numbers."""
7      return a + b
8
9  def subtract(a, b):
10     """Subtract b from a."""
11     return a - b
12
13  def multiply(a, b):
14     """Multiply two numbers."""
15     return a * b
16
17  def divide(a, b):
18     """Divide a by b. Error if b is zero."""
19     if b == 0:
20         return "Cannot divide by zero"
21     return a / b
22
23  x = float(input("Enter first number: "))
24  y = float(input("Enter second number: "))
25  op = input("Choose operation (+, -, *, /): ")
26
27  if op == "+":
28      print("Result:", add(x, y))
29  elif op == "-":
30      print("Result:", subtract(x, y))
31  elif op == "*":
32      print("Result:", multiply(x, y))
33  elif op == "/":
34      print("Result:", divide(x, y))
35  else:
36      print("Invalid operation")

```

- **COMPARISION** : The user docstring is very short ,informal,and direct where as ai docstring is longer,formal and follows documentation standards like numpy,return values.
