

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		1. Dr. Mohammed Ali Shaik 2. Dr. T Sampath Kumar 3. Mr. S Naresh Kumar 4. Dr. V. Rajesh 5. Dr. Brij Kishore 6. Dr Pramoda Patro 7. Dr. Venkataramana 8. Dr. Ravi Chander 9. Dr. Jagjeeth Singh	
CourseCode	24CS002PC215	CourseTitle	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week2-Tuesday	Time(s)	
Duration	2 Hours	Applicable to Batches	24CSBTB01 To 24CSBTB39
AssignmentNumber:3.2(Present assignment number)/24(Total number of assignments)			

Q.No.	Question	Expected Time to complete
1	<p>Lab 3: Prompt Engineering – Improving Prompts and Context Management</p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> To understand how prompt structure and wording influence AI-generated code. To explore how context (like comments and function names) helps AI generate relevant output. To evaluate the quality and accuracy of code based on prompt clarity. To develop effective prompting strategies for AI-assisted programming. <p>Lab Outcomes (LOs): After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> Generate Python code using Google Gemini in Google Colab. Analyze the effectiveness of code explanations and suggestions by Gemini. Set up and use Cursor AI for AI-powered coding assistance. Evaluate and refactor code using Cursor AI features. Compare AI tool behavior and code quality across different platforms. 	03.08.2025 EOD

Task Description#1

- Ask AI to write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example
- Expected code:

```
Ass1.py > ...
1  def calculate_compound_interest(principal, rate, time, n):
2
3      return principal * (1 + rate / n) ** (n * time)
4
5  # Example usage:
6  # Calculate the amount after 3 years for $1000 at 5% annual interest, compounded quarterly.
7  amount = calculate_compound_interest(1000, 0.05, 3, 4)
8  # Line-by-line explanation:
9
10 # Define a function to calculate compound interest.
11 def calculate_compound_interest(principal, rate, time, n):
12     # Calculate and return the compound interest formula result.
13     return principal * (1 + rate / n) ** (n * time)
14
15 # Example usage:
16 # Calculate the amount after 3 years for $1000 at 5% annual interest, compounded quarterly.
17 amount = calculate_compound_interest(1000, 0.05, 3, 4)
18 # Print the calculated amount.
19 print(amount)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
1160.7545177229981
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
1160.7545177229981
brungisrikar@Brungis-MacBook-Pro WTMP %
```

Task Description#2

- Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.

Expected Output#2

- AI-generated function evolves from unclear to accurate multi-statistical operation.

```
Ass1.py > calculate_stats

3 def calculate_stats(numbers):
4     """
5     if not numbers:
6         return {'average': None, 'median': None, 'mode': None}
7
8     # Average
9     average = sum(numbers) / len(numbers)
10
11    # Median
12    sorted_nums = sorted(numbers)
13    n = len(sorted_nums)
14    if n % 2 == 1:
15        median = sorted_nums[n // 2]
16    else:
17        median = (sorted_nums[n // 2 - 1] + sorted_nums[n // 2]) / 2
18
19    # Mode
20    counts = Counter(numbers)
21    max_count = max(counts.values())
22    mode = [num for num, count in counts.items() if count == max_count]
23    if len(mode) == len(numbers):
24        mode = None # No mode if all values are unique
25    elif len(mode) == 1:
26        mode = mode[0] # Single mode
27
28    return {'average': average, 'median': median, 'mode': mode}
29
30 # Example usage:
31 nums = [1, 2, 2, 3, 4]
32 stats = calculate_stats(nums)
33 print(stats)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/
1160.7545177229981
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/
1160.7545177229981
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/
brungisrikar@Brungis-MacBook-Pro WTMP % hi
zsh: command not found: hi
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/
{'average': 2.4, 'median': 2, 'mode': 2}
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/
{'average': 2.4, 'median': 2, 'mode': 2}
```

●

Task Description#3

- Provide multiple examples of input-output to the AI for convert_to_binary(num) function. Observe how AI uses few-shot prompting to generalize.

Expected Output#3

- Enhanced AI output with clearer prompts

```
Ass1.py > ...
1 # Function to convert a number to binary
2 def convert_to_binary(num):
3     return bin(num)[2:]
4
5 # Example inputs and outputs for convert_to_binary(num)
6 examples = [
7     (5, "101"),
8     (10, "1010"),
9     (0, "0"),
10    (255, "11111111"),
11    (1, "1"),
12    (32, "100000"),
13 ]
14
15 # Displaying the examples
16 for num, expected_output in examples:
17     result = convert_to_binary(num)
18     print(f"Input: {num} | Expected Output: {expected_output} | Actual Output: {result}")
19
20 # Explanation:
21 # Few-shot prompting means providing several input-output examples to guide the AI in understanding the task.
22 # Here, we show how different numbers are converted to their binary representations.
23 # This helps the AI generalize the pattern: for any integer input, output its binary string (without the '0b' pre
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
brungisrikar@Brungis-MacBook-Pro WTMP % hi
zsh: command not found: hi
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
{'average': 2.4, 'median': 2, 'mode': 2}
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
{'average': 2.4, 'median': 2, 'mode': 2}
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
Input: 5 | Expected Output: 101 | Actual Output: 101
Input: 10 | Expected Output: 1010 | Actual Output: 1010
Input: 0 | Expected Output: 0 | Actual Output: 0
Input: 255 | Expected Output: 11111111 | Actual Output: 11111111
Input: 1 | Expected Output: 1 | Actual Output: 1
Input: 32 | Expected Output: 100000 | Actual Output: 100000
```

Task Description#4

- Create an user interface for an hotel to generate bill based on customer requirements

Expected Output#4

- Consistent functions with shared logic

Ass1.py > ...

```
1  import tkinter as tk
2  from tkinter import messagebox
3
4  # Sample menu and prices
5  MENU = {
6      "Room": 2000,
7      "Breakfast": 250,
8      "Lunch": 400,
9      "Dinner": 500,
10     "Laundry": 150,
11     "Spa": 800
12 }
13
14 def calculate_bill(selections):
15     total = 0
16     details = []
17     for item, var in selections.items():
18         qty = var.get()
19         if qty > 0:
20             cost = MENU[item] * qty
21             details.append(f"{item} x {qty} = ₹{cost}")
22             total += cost
23     return details, total
24
25 def generate_bill():
26     details, total = calculate_bill(selections)
27     if not details:
28         messagebox.showinfo("Bill", "No services selected.")
29         return
30     bill_text = "\n".join(details) + f"\n\nTotal Bill: ₹{total}"
31     messagebox.showinfo("Bill", bill_text)
```

```

Ass1.py > ...
32
33 # UI setup
34 root = tk.Tk()
35 root.title("Hotel Bill Generator")
36
37 tk.Label(root, text="Select Services and Quantity", font=("Arial", 14)).pack(pady=10)
38
39 frame = tk.Frame(root)
40 frame.pack()
41
42 selections = {}
43 for idx, (item, price) in enumerate(MENU.items()):
44     tk.Label(frame, text=f"{item} (₹{price})", width=15, anchor='w').grid(row=idx, column=0, padx=5, pady=5)
45     var = tk.IntVar()
46     tk.Spinbox(frame, from_=0, to=10, textvariable=var, width=5).grid(row=idx, column=1, padx=5)
47     selections[item] = var
48
49 tk.Button(root, text="Generate Bill", command=generate_bill, font=("Arial", 12)).pack(pady=20)
50
51 root.mainloop()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

• brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
• brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
{'average': 2.4, 'median': 2, 'mode': 2}
• brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
{'average': 2.4, 'median': 2, 'mode': 2}
• brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
Input: 5 | Expected Output: 101 | Actual Output: 101
Input: 10 | Expected Output: 1010 | Actual Output: 1010
Input: 0 | Expected Output: 0 | Actual Output: 0
Input: 255 | Expected Output: 1111111 | Actual Output: 1111111
Input: 1 | Expected Output: 1 | Actual Output: 1
Input: 32 | Expected Output: 100000 | Actual Output: 100000
• brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
/usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py

```

Task Description#5

- Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

Expected Output#5

- Code quality difference analysis for various prompts

```

Ass1.py > ...
1 def convert_temperature(value, from_unit, to_unit):
2
3     elif from_unit == 'F':
4         celsius = (value - 32) * 5 / 9
5     elif from_unit == 'K':
6         celsius = value - 273.15
7     else:
8         raise ValueError("Invalid from_unit. Use 'C', 'F', or 'K'.")
9
10    # Convert Celsius to target unit
11    if to_unit == 'C':
12        return celsius
13    elif to_unit == 'F':
14        return celsius * 9 / 5 + 32
15    elif to_unit == 'K':
16        return celsius + 273.15
17    else:
18        raise ValueError("Invalid to_unit. Use 'C', 'F', or 'K'.")
19
20    # Example usage
21    result = convert_temperature(100, 'C', 'F')
22    print(result)

```

- Explanation:

Ass1.py > ...

```
25 #Explanation:
26 """
27     Converts temperature between Celsius, Fahrenheit, and Kelvin.
28
29     Parameters:
30     value (float): The temperature value to convert.
31     from_unit (str): The unit of the input temperature ('C', 'F', 'K').
32     to_unit (str): The unit to convert to ('C', 'F', 'K').
33
34     Returns:
35     float: Converted temperature value.
36
37     Example:
38     convert_temperature(100, 'C', 'F') -> 212.0
39 """
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/python3 /Users/brungisrikar/Desktop/WTMP/Ass1.py
brungisrikar@Brungis-MacBook-Pro WTMP % /usr/local/bin/python3 /Users/brungisrika
212.0
```

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

Evaluation Criteria:

Criteria	Max Marks
Task#1	0.5
Task#2	0.5
Task #3	0.5
Task #4	0.5
Task #5	0.5
Total	2.5 Marks