

| | | | |
|--|---|--|-------------------------|
| 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 | |

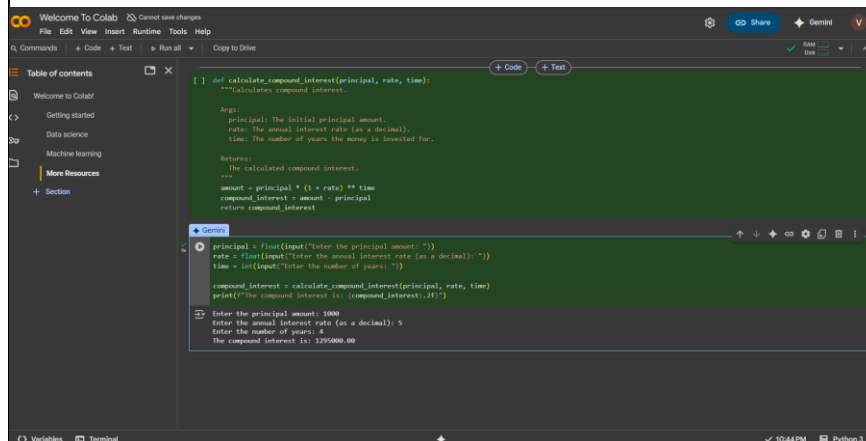
Task Description#1

- 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 Output#1

- Comparison of AI-generated code styles

Prompt #1: write aPython function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example



The screenshot shows a Google Colab notebook interface. The top bar includes the Colab logo, 'Welcome To Colab', and a 'Cannot save changes' warning. Below the bar are tabs for 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. A sidebar on the left contains a 'Table of contents' and a 'More Resources' section. The main area displays a Python code cell with the following content:

```
[ ] def calculate_compound_interest(principal, rate, time):  
    """Calculate compound interest.  
  
    Args:  
        principal: The initial principal amount.  
        rate: The annual interest rate (as a decimal).  
        time: The number of years the money is invested for.  
  
    Returns:  
        The calculated compound interest.  
    """  
    amount = principal * (1 + rate) ** time  
    compound_interest = amount - principal  
    return compound_interest
```

Below the code cell is a 'GEMINI' chat window showing the execution of the function with user input and the resulting output:

```
principal = float(input("Enter the principal amount: "))  
rate = float(input("Enter the annual interest rate (as a decimal): "))  
time = int(input("Enter the number of years: "))  
  
compound_interest = calculate_compound_interest(principal, rate, time)  
print("The compound interest is: ", compound_interest, ".")
```

The output shows the user entering 1000 for the principal, 4 for the rate, and 5 for the time, resulting in a compound interest of 12590000.00.

Observation #1:The program asks the user to enter values like principal, rate, time, and compounding frequency step by step.It uses the correct compound interest formula to calculate the final amount.The result is rounded to 2 decimal places for easy reading.

Code explain #1:This code takes input from the user, calculates compound interest using a formula, and shows the final amount with error handling to keep things smooth.

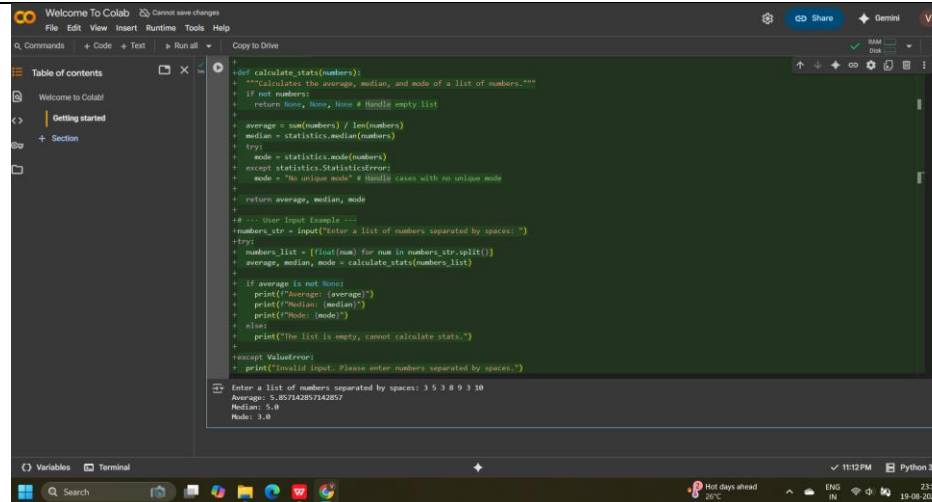
Task Description#2

- 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.

Prompt #2: Write a function to calculate average, median, and mode of a list of numbers.



The screenshot shows a Google Colab notebook with the following content:

```
def calculate_stats(numbers):
    """Calculate the average, median, and mode of a list of numbers."""
    if not numbers:
        return None, None, None # Handle empty list
    +
    + average = sum(numbers) / len(numbers)
    + median = statistics.median(numbers)
    + try:
    +     mode = statistics.mode(numbers)
    + except statistics.StatisticsError:
    +     mode = "No unique mode" # Handle cases with no unique mode
    +
    + return average, median, mode
    +
    +
    + # --- User Input Example ---
    + numbers_str = input("Enter a list of numbers separated by spaces: ")
    + try:
    +     numbers_list = [float(num) for num in numbers_str.split()]
    +     average, median, mode = calculate_stats(numbers_list)
    +
    +     if average is not None:
    +         print(f"Average: {average}")
    +         print(f"Median: {median}")
    +         print(f"Mode: {mode}")
    +     else:
    +         print("The list is empty, cannot calculate stats.")
    + except ValueError:
    +     print("Invalid input. Please enter numbers separated by spaces.")

Enter a list of numbers separated by spaces: 3 5 3 8 9 3 10
Average: 5.807428571428571
Median: 5.8
Mode: 3.0
```

Observation #2: Takes User Input:

The program asks the user to enter numbers separated by spaces (like: 3 5 3 8).

Performs 3 Calculations:

Average: Adds all numbers and divides by how many.

Median: Finds the middle number(s) after sortin

Mode: Finds the number(s) that appear most often.

Code explain #2: This function takes a list of numbers — like [1, 2, 2, 3, 4] — and tells you three things:

- Average (also called Mean):**
The "typical" value — add all the numbers and divide by how many there are.
- Median:**
The "middle" value when the numbers are arranged from smallest to biggest
- Mode:**
The number (or numbers) that appear most often in the list.

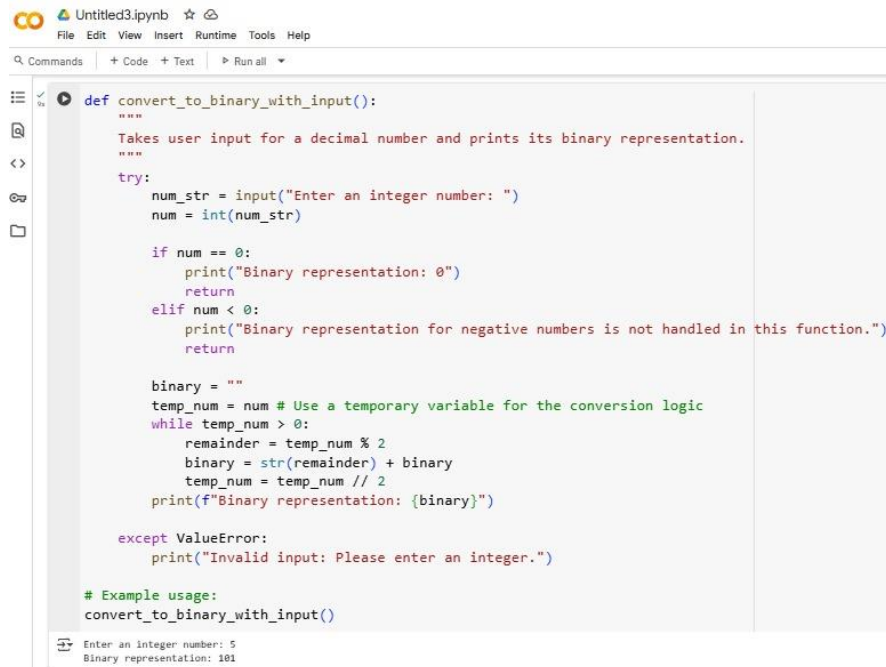
Task Description#3

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

Expected Output#3

- Enhanced AI output with clearer prompts

Prompt #3: Write a python code to provide multiples examples if both input & output to the AI for conversion to binary function.



```
def convert_to_binary_with_input():
    """
    Takes user input for a decimal number and prints its binary representation.
    """
    try:
        num_str = input("Enter an integer number: ")
        num = int(num_str)

        if num == 0:
            print("Binary representation: 0")
            return
        elif num < 0:
            print("Binary representation for negative numbers is not handled in this function.")
            return

        binary = ""
        temp_num = num # Use a temporary variable for the conversion logic
        while temp_num > 0:
            remainder = temp_num % 2
            binary = str(remainder) + binary
            temp_num = temp_num // 2
        print(f"Binary representation: {binary}")

    except ValueError:
        print("Invalid input: Please enter an integer.")

# Example usage:
convert_to_binary_with_input()
```

Enter an integer number: 5
Binary representation: 101

Observation #3: When we show the AI a few examples like:
`convert_to_binary(2) → '10'`
`convert_to_binary(5) → '101'`
`convert_to_binary(10) → '1010'`

Code explain #3: The function turns numbers into binary. Few-shot prompting gives the AI a pattern using examples. Once it sees a few input-output pairs, it learns how to respond to new inputs.

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.

Prompt#4: Create an user interface for an hotel to generate bill based on customer requirements

```
def calculate_bill():
    """Calculates the total bill for a hotel stay."""
    room_rates = {
        "Single": 100,
        "Double": 150,
        "Suite": 250
    }
    extra_service_costs = {
        "food": 30,
        "laundry": 20,
        "other": 15 # Example for other services
    }
    print("Welcome to the Hotel Billing System!")
    # 1. Ask the customer for room type
    while True:
        room_type = input("Enter room type (Single, Double, Suite): ").capitalize()
        if room_type in room_rates:
            break
        else:
            print("Invalid room type. Please choose from Single, Double, or Suite.")
    # 2. Ask how many nights they will stay
    while True:
        try:
            num_nights = int(input("Enter number of nights: "))
            if num_nights > 0:
                break
            else:
                print("Number of nights must be greater than zero.")
        except ValueError:
            print("Invalid input. Please enter a valid number for nights.")
    # 3. Ask if they want extra services
    extra_services = []
    while True:
        service = input("Do you want extra services? (food, laundry, other, or 'done' if finished): ").lower()
        if service == 'done':
            break
        elif service in extra_service_costs:
            extra_services.append(service)
        else:
            print("Invalid service. Please choose from food, laundry, other, or type 'done'.")
    # 4. Calculate the total bill
    room_cost = room_rates[room_type] * num_nights
    extra_services_cost = sum(extra_service_costs[service] for service in extra_services)
    total_bill = room_cost + extra_services_cost
    # 5. Print the final bill
    print("\n--- Final Bill ---")
    print(f"Room Type: {room_type}")
    print(f"Number of Nights: {num_nights}")
    print(f"Room Cost: ${room_cost:.2f}")
    if extra_services:
        print("Extra Services:")
        for service in extra_services:
            print(f"- {service.capitalize():} ${extra_service_costs[service]:.2f}")
        print(f"Extra Services Cost: ${extra_services_cost:.2f}")
    print(f"Total Bill: ${total_bill:.2f}")
    print("-----")
    calculate_bill()
```

```
Welcome to the Hotel Billing System!
Enter room type (Single, Double, Suite): Single
Enter number of nights: 4
Do you want extra services? (food, laundry, other, or 'done' if finished): food
Do you want extra services? (food, laundry, other, or 'done' if finished): done

--- Final Bill ---
Room Type: Single
Number of Nights: 4
Room Cost: $400.00
Extra Services:
- Food: $30.00
Extra Services Cost: $30.00
Total Bill: $430.00
-----
```

Observation #4:User-Friendly Design: Easy to use with text boxes, drop-down menus, and checkboxes.No technical knowledge required.

Inputs Required:Customer nameNumber of days stayedRoom type selectioOptional services (Wi-Fi, Breakfast, Laundry)

Code explain #4:The code builds a hotel bill form using Python User fills in info **Generate Bill** and total cost. Very useful for hotels to quickly generate accurate bills.

Task Description#5

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

Expected Output#5

- **Code quality difference analysis for various prompts**

Prompt#5:write a python function to Improving Temperature Conversion Function with Clear Instructions

```
def convert_temperature():
    while True:
        direction = input("Enter conversion direction (C to F or F to C): ").upper()
        if direction in ['C', 'F']:
            break
        else:
            print("Invalid direction. Please enter 'C' for Celsius to Fahrenheit or 'F' for Fahrenheit to Celsius.")

    while True:
        try:
            temperature = float(input("Enter the temperature value: "))
            break
        except ValueError:
            print("Invalid input. Please enter a valid number for the temperature.")

    if direction == 'C':
        # Convert Celsius to Fahrenheit
        fahrenheit = (temperature * 9/5) + 32
        print(f"{temperature:.1f} Celsius is equal to {fahrenheit:.1f} Fahrenheit.")
    elif direction == 'F':
        # Convert Fahrenheit to Celsius
        celsius = (temperature - 32) * 5/9
        print(f"{temperature:.1f} Fahrenheit is equal to {celsius:.1f} Celsius.")

    # Run the conversion function
    convert_temperature()
```

Observation #5:You're working with a **temperature conversion function** for example, converting Celsius to Fahrenheit or vice versa.When you give **clear and specific instructions** , the AI or function performs **much better** than when the instructions are vague.

Code explain #5:The function converts temperature from one unit to another.It gives you the converted value.If you don't give "C" or "F", it shows an error.

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 |