<table>
<tr><td colspan="2"><b>SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE</b></td><td colspan="2"><b>DEPARTMENT OF COMPUTER SCIENCE ENGINEERING</b></td></tr>
<tr><td colspan="2"><b>ProgramName:</b>B. Tech</td><td><b>Assignment Type: Lab</b></td><td><b>AcademicYear:</b>2025-2026</td></tr>
<tr><td colspan="2"><b>CourseCoordinatorName</b></td><td colspan="2">Venkataramana Veeramsetty</td></tr>
<tr><td colspan="2"><b>Instructor(s)Name</b></td><td colspan="2">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</td></tr>
<tr><td><b>CourseCode</b></td><td>24CS002PC215</td><td><b>CourseTitle</b></td><td>AI Assisted Coding</td></tr>
<tr><td><b>Year/Sem</b></td><td>II/I</td><td><b>Regulation</b></td><td>R24</td></tr>
<tr><td><b>Date and Day of Assignment</b></td><td>Week2-Tuesday</td><td><b>Time(s)</b></td><td></td></tr>
<tr><td><b>Duration</b></td><td>2 Hours</td><td><b>Applicableto Batches</b></td><td>24CSBTB01 To 24CSBTB39</td></tr>
<tr><td colspan="4"><b>AssignmentNumber:3.2</b>(Present assignment number)/<b>24</b>(Total number of assignments)</td></tr>
<tr><td colspan="4"></td></tr>
</table>

Q.No.	**Question**	***ExpectedTime to complete***

1

03.08.
2025
EOD

Lab 3: Prompt Engineering – Improving Prompts and Context Management

Lab Objectives:

- 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.

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- 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.

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

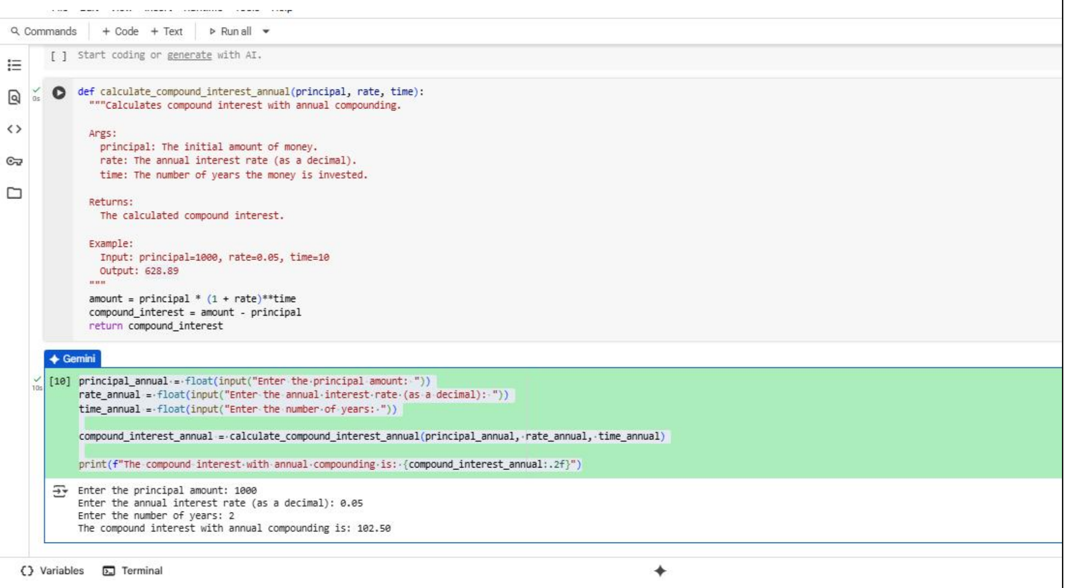
Expected Output#1

- Comparison of AI-generated code styles

PROMPT#1

1. Write a Python function to calculate compound interest. Give only the function name.
2. Now add a docstring.
3. Add an example input and output in the docstring.

CODE#1



```
[ ] Start coding or generate with AI.
```

```python
def calculate_compound_interest_annual(principal, rate, time):
  """Calculates compound interest with annual compounding.

  Args:
    principal: The initial amount of money.
    rate: The annual interest rate (as a decimal).
    time: The number of years the money is invested.

  Returns:
    The calculated compound interest.

  Example:
    Input: principal=1000, rate=0.05, time=10
    Output: 628.89
  """
  amount = principal * (1 + rate)**time
  compound_interest = amount - principal
  return compound_interest
```

```python
[10] principal_annual = float(input("Enter the principal amount: "))
rate_annual = float(input("Enter the annual interest rate (as a decimal): "))
time_annual = float(input("Enter the number of years: "))

compound_interest_annual = calculate_compound_interest_annual(principal_annual, rate_annual, time_annual)

print(f"The compound interest with annual compounding is: {compound_interest_annual:.2f}")
```

```
Enter the principal amount: 1000
Enter the annual interest rate (as a decimal): 0.05
Enter the number of years: 2
The compound interest with annual compounding is: 102.50
```

OBSERVATION#1

1. The function is defined with a clear name.
2. A docstring is provided to explain the purpose.
3. The standard compound interest formula is used.

Task Description#2

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

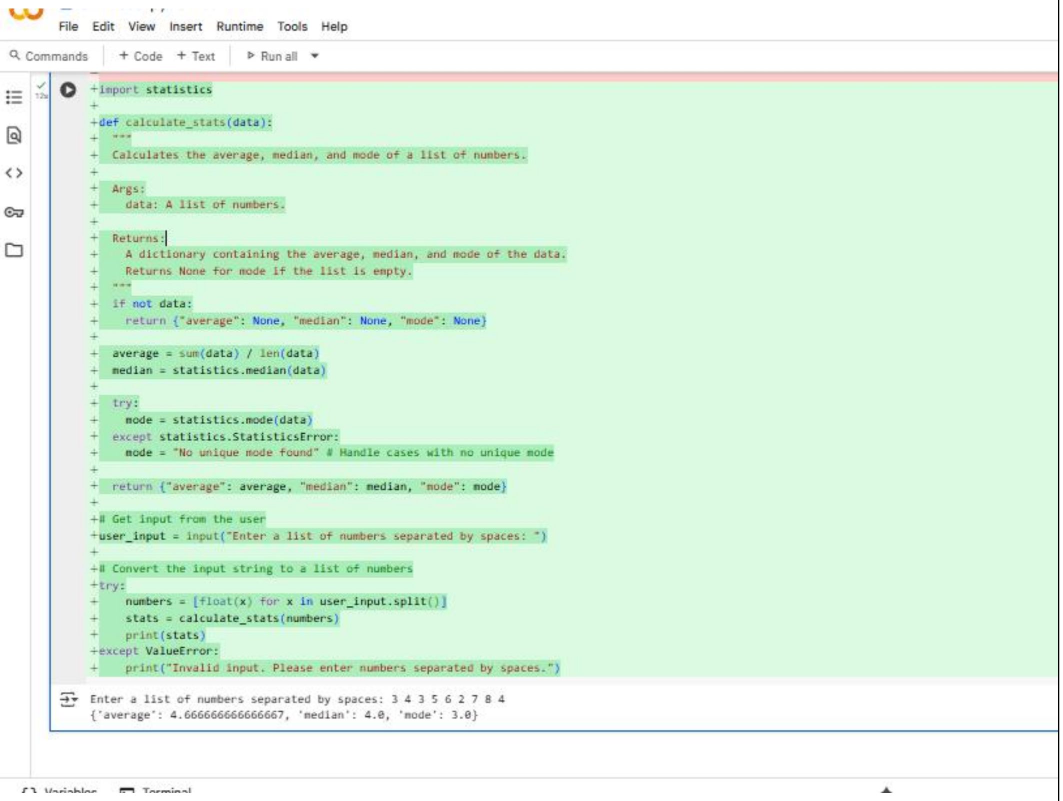
Expected Output#2

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

PROMPT#2

1. write a python function to calculate the avearge,median and mode of the given list of numbers.

CODE#2



```
+import statistics
+
+def calculate_stats(data):
+  """
+  Calculates the average, median, and mode of a list of numbers.
+
+  Args:
+    data: A list of numbers.
+
+  Returns:
+    A dictionary containing the average, median, and mode of the data.
+    Returns None for mode if the list is empty.
+  """
+  if not data:
+    return {"average": None, "median": None, "mode": None}
+
+  average = sum(data) / len(data)
+  median = statistics.median(data)
+
+  try:
+    mode = statistics.mode(data)
+  except statistics.StatisticsError:
+    mode = "No unique mode found" # Handle cases with no unique mode
+
+  return {"average": average, "median": median, "mode": mode}
+
+# Get input from the user
+user_input = input("Enter a list of numbers separated by spaces: ")
+
+# Convert the input string to a list of numbers
+try:
+    numbers = [float(x) for x in user_input.split()]
+    stats = calculate_stats(numbers)
+    print(stats)
+except ValueError:
+    print("Invalid input. Please enter numbers separated by spaces.")
```

```
Enter a list of numbers separated by spaces: 3 4 3 5 6 2 7 8 4
{'average': 4.666666666666667, 'median': 4.0, 'mode': 3.0}
```

OBSERVATION#2

1. It takes a list of numbers as input.
2. It computes avearge using the sum of numbers divided by their count.
3. It computes median by finding the middle value of the sorted list.
4. It computes mode as the most frequently occurring number.

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

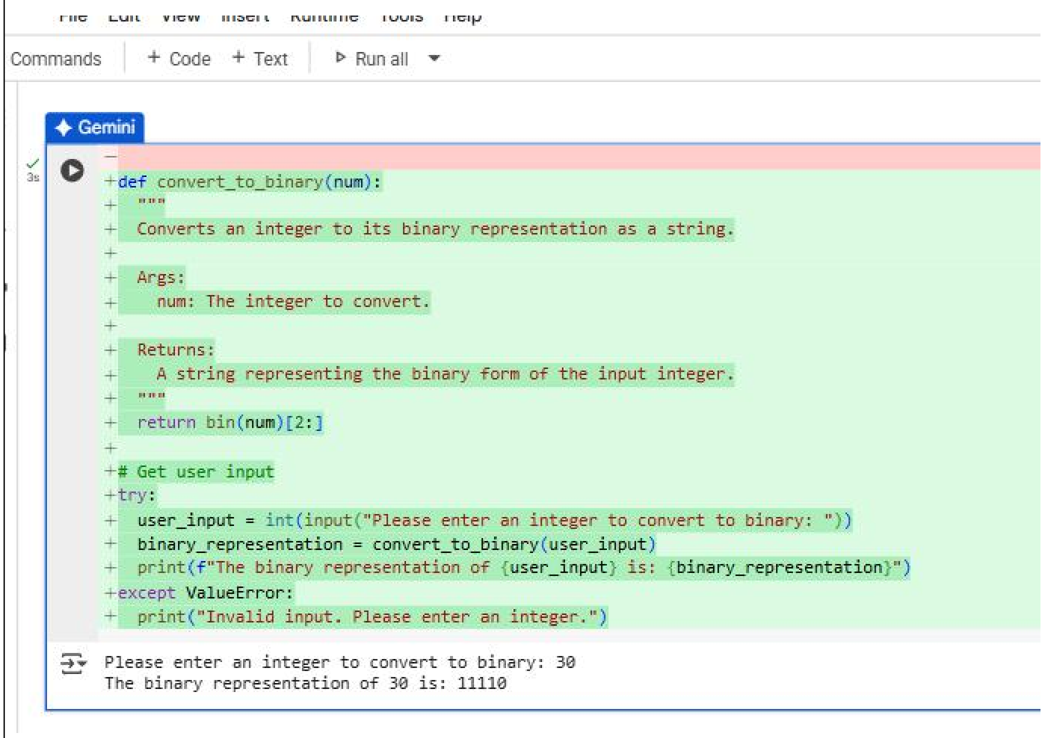
- Enhanced AI output with clearer prompts

PROMPT#3

1. Write a Python function named convert_to_binary(num).

2. The function should take an integer input.
3. It should return the binary representation as a string.

CODE#3



```
-
+def convert_to_binary(num):
+  """
+  Converts an integer to its binary representation as a string.
+
+  Args:
+    num: The integer to convert.
+
+  Returns:
+    A string representing the binary form of the input integer.
+  """
+  return bin(num)[2:]
+
+# Get user input
+try:
+  user_input = int(input("Please enter an integer to convert to binary: "))
+  binary_representation = convert_to_binary(user_input)
+  print(f"The binary representation of {user_input} is: {binary_representation}")
+except ValueError:
+  print("Invalid input. Please enter an integer.")
```

```
Please enter an integer to convert to binary: 30
The binary representation of 30 is: 11110
```

OBSERVATION#3

1. AI learns the pattern of converting decimal to binary.
2. It generalizes the rule to give binary digits

Task Description#4

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

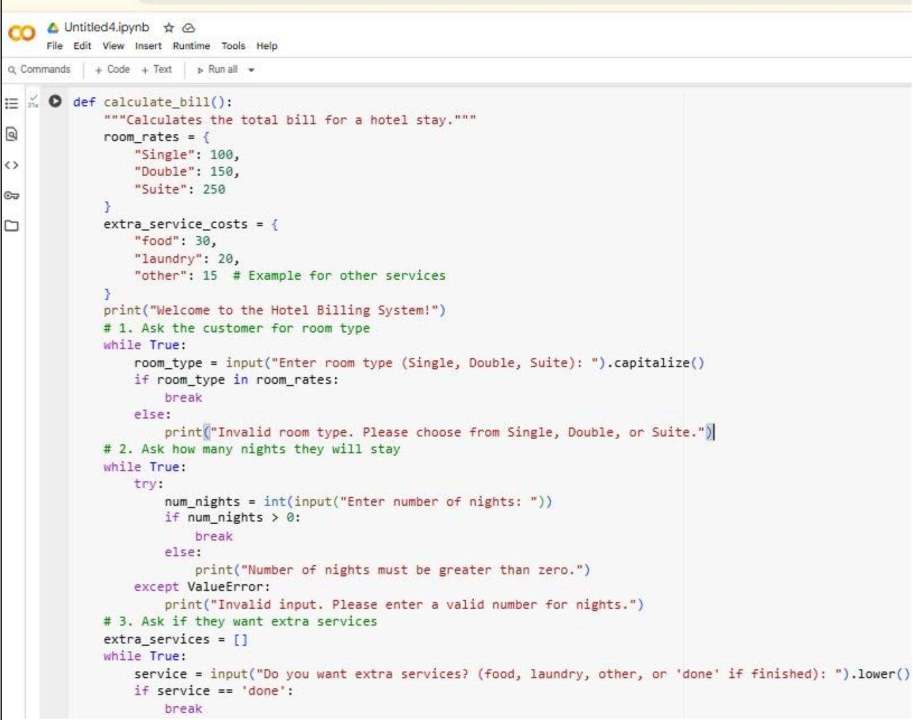
Expected Output#4

- Consistent functions with shared logic

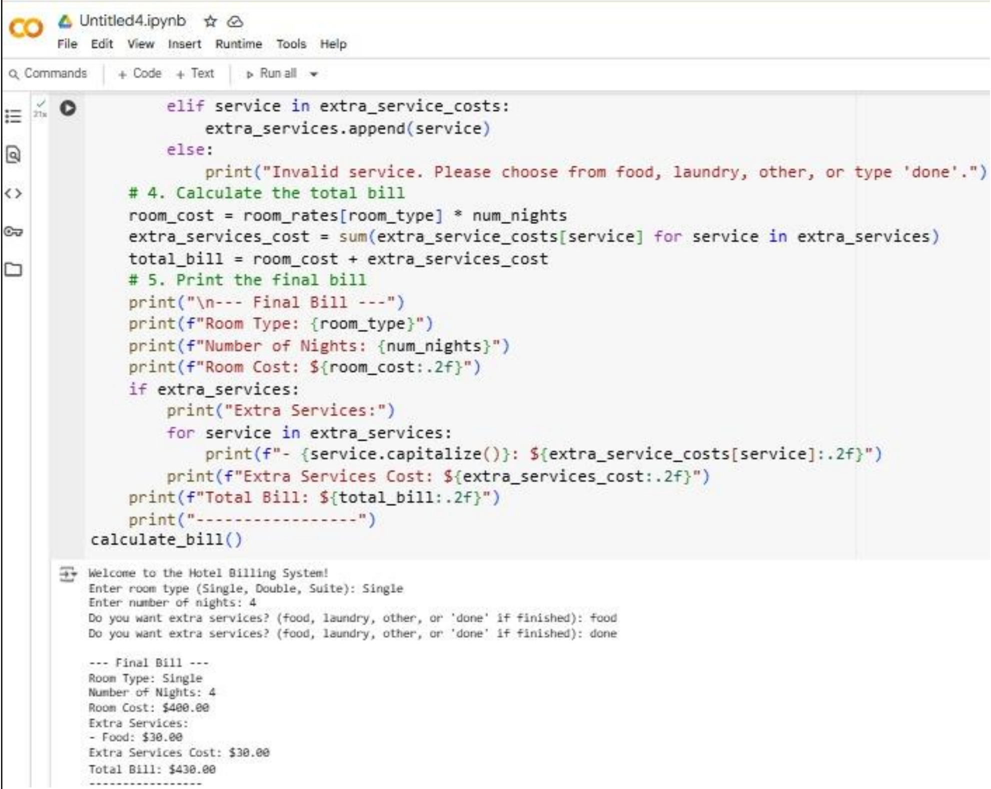
PROMPT#4

1. Create a hotel billing system user interface.
2. Customer can choose a room type.
3. Customer can select meals.
4. When clicking generate bill the program should show the total amount.

CODE#4


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all

```python
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
```


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all

```python
        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

1. user can select room type and meals.
2. Calculates total bill based on selections.
3. Displays detailed bill.

Task Description#5

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

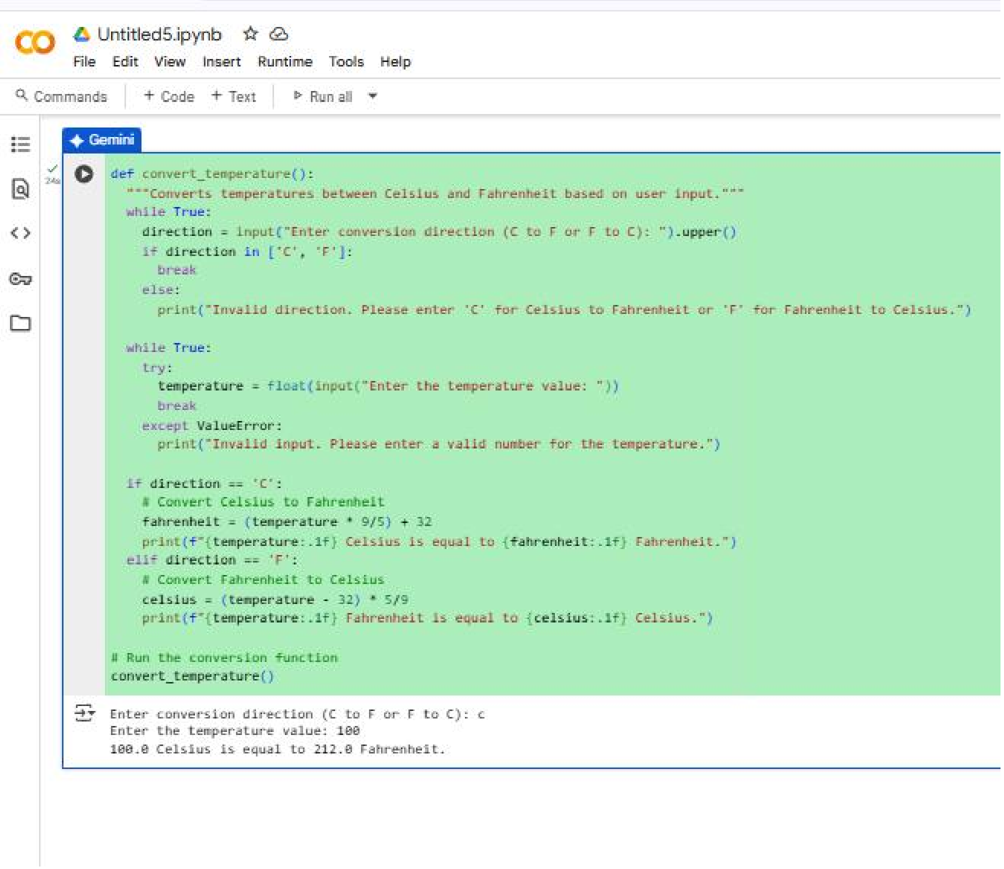
Expected Output#5

- Code quality difference analysis for various prompts.

PROMPT#5

1. Write a function convert_temperature(value, scale).
2. conversions between celsius,fahrenheit,kelvin.
3. Use scale parameter to choose target unit.
4. Return the converted value with examples.

CODE#5



```python
def convert_temperature():
  """Converts temperatures between Celsius and Fahrenheit based on user input."""
  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()
```

```
Enter conversion direction (C to F or F to C): c
Enter the temperature value: 100
100.0 Celsius is equal to 212.0 Fahrenheit.
```

OBSERVATION#5

1. The function converts temperatures between units.
2. Uses a parameter to choose the target scale.
3. Works for Celsius, Fahrenheit, and Kelvin.
4. Gives correct output with examples.

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**