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		**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:3.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	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.	03.08.2025 EOD

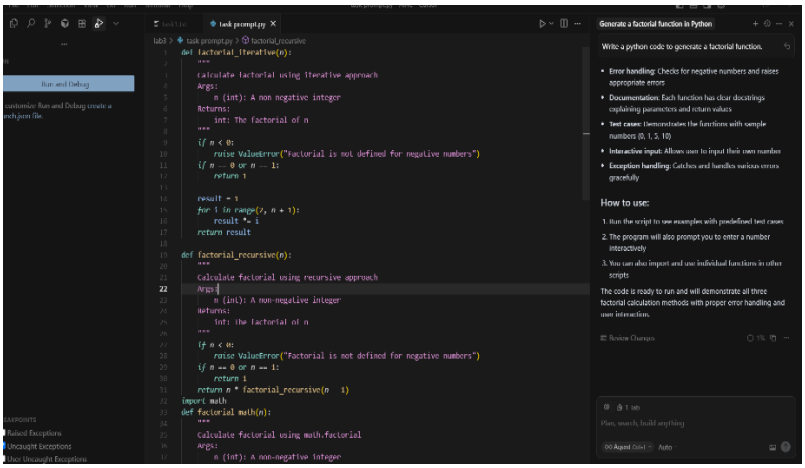
Task Description#1

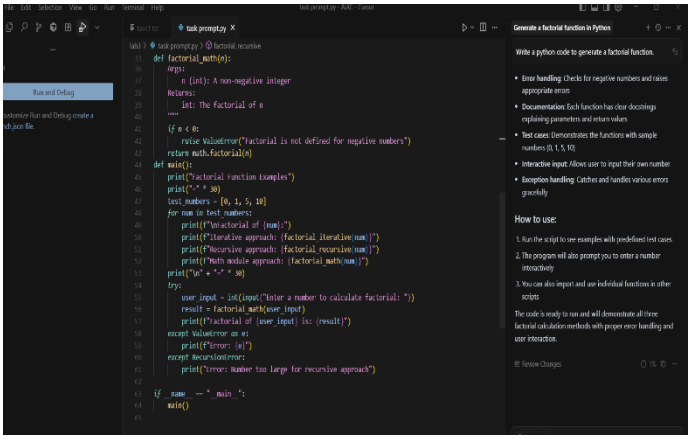
- Try 3 different prompts to generate a factorial function.

Expected Output#1

- Comparison of AI-generated code styles

PROMPT 1: *WRITE A PYTHON CODE TO GENERATE A FACTORIAL FUNCTION.*





OUTPUT:

```
Problems  Output  Debug Console  Terminal  Ports                Python Debug Console

Factorial of 0:
Iterative approach: 1
Recursive approach: 1
Math module approach: 1

Factorial of 1:
Iterative approach: 1
Recursive approach: 1
Math module approach: 1

Factorial of 5:
Iterative approach: 120
Recursive approach: 120
Math module approach: 120

Factorial of 10:
Iterative approach: 3628800
Recursive approach: 3628800
Math module approach: 3628800

==========
Enter a number to calculate factorial: 6
Factorial of 6 is: 720
PS C:\Users\MEGHANA\Documents\AIAC>
```

PROMPT 2: *WRITE A FUNCTION TO FIND THE FACTORIAL OF THE GIVEN NUMBER.*

OUTPUT:

```
==============================
Enter a number to calculate factorial: 6
==============================
Enter a number to calculate factorial: 6
Factorial of 6 is: 720
Enter a number to calculate factorial: 6
Factorial of 6 is: 720
Factorial of 6 is: 720
PS C:\Users\MEGHANA\Documents\AIAC> ^C
PS C:\Users\MEGHANA\Documents\AIAC>
PS C:\Users\MEGHANA\Documents\AIAC>  c:; cd 'c:\Users\MEGHANA\Documents\AIAC'; & 'c:
```

PROMPT 3: WRITE A PYTHON CODE TO KNOWN THE FACTORIAL OF THE GIVEN INPUT

task1.txt | task prompt.py | task prompt2.py | task prompt3.py

Task prompt3.py > ...

```python
def factorial(n):
    """Calculate factorial of a number"""
    if n < 0:
        return "Error: Factorial is not defined for negative numbers"
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

def main():
    try:
        # Get input from user
        num = int(input("Enter a number to calculate factorial: "))

        # Calculate factorial
        result = factorial(num)

        # Display result
        if isinstance(result, str):
            print(result)
        else:
            print(f"Factorial of {num} is: {result}")

    except ValueError:
        print("Error: Please enter a valid integer")
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Enter a number to calculate factorial: 5
Factorial of 5 is: 120
PS C:\Users\MEGHANA\Documents\AIAC>
```

USE OF USING THREE DIFFERENT PROMPTS:

To explore alternative solutions or specific programming paradigms and we will be getting a knowledge of writing a different types of codes.

Task Description#2

- Provide a clear example input-output prompt to generate a sorting function.

PROMPT: *WRITE A PYTHON CODE TO GENERATE A SORTING FUNCTION.*

task1.txt | task prompt.py | task prompt2.py | task prompt3.py | TASK2.py

lab3 > TASK2.py > ...

```python
def generate_sorting_function():
    """
    Generates a comprehensive sorting function with multiple algorithms
    Returns a function that can sort lists using different methods
    """
    import random
    import time

    def bubble_sort(arr):
        n = len(arr)
        for i in range(n):
            for j in range(0, n - i - 1):
                if arr[j] > arr[j + 1]:
                    arr[j], arr[j + 1] = arr[j + 1], arr[j]
        return arr

    def quick_sort(arr):
        if len(arr) <= 1:
            return arr
        pivot = arr[len(arr) // 2]
        left = [x for x in arr if x < pivot]
        middle = [x for x in arr if x == pivot]
        right = [x for x in arr if x > pivot]
        return quick_sort(left) + middle + quick_sort(right)

    def merge_sort(arr):
        if len(arr) <= 1:
            return arr
        mid = len(arr) // 2
        left = merge_sort(arr[:mid])
        right = merge_sort(arr[mid:])
        return merge(left, right)

    def merge(left, right):
        result = []
        i = j = 0
        while i < len(left) and j < len(right):
            if left[i] <= right[j]:
                result.append(left[i])
                i += 1
            else:
                result.append(right[j])
                j += 1
        result.extend(left[i:])
        result.extend(right[j:])
        return result

    def smart_sort(arr, algorithm='auto'):
        if algorithm == 'auto':
            if len(arr) <= 10:
                algorithm = 'bubble'
            elif len(arr) <= 100:
                algorithm = 'merge'
            else:
                algorithm = 'quick'

        if algorithm == 'bubble':
            return bubble_sort(arr.copy())
        elif algorithm == 'quick':
            return quick_sort(arr.copy())
        elif algorithm == 'merge':
            return merge_sort(arr.copy())
        else:
            return sorted(arr)

    return smart_sort

# Generate and test the sorting function
sort_func = generate_sorting_function()
```

```
sort_func = generate_sorting_function()

# Example usage and testing
if __name__ == "__main__":
    # Test data
    test_lists = [
        [64, 34, 25, 12, 22, 11, 90],
        [5, 2, 4, 6, 1, 3],
        [1],
        []
    ]

    for lst in test_lists:
        original = lst.copy()
        sorted_list = sort_func(lst)
        print(f"Original: {original}")
        print(f"Sorted:   {sorted_list}")
        print(f"Algorithm: {'auto'}")
        print("-" * 30)
```

OUTPUT:

```
Original: [64, 34, 25, 12, 22, 11, 90]
Sorted:   [11, 12, 22, 25, 34, 64, 90]
Algorithm: auto
------------------------------
Original: [5, 2, 4, 6, 1, 3]
Sorted:   [1, 2, 3, 4, 5, 6]
Algorithm: auto
------------------------------
Original: [1]
Sorted:   [1]
Algorithm: auto
------------------------------
Original: []
Sorted:   []
Algorithm: auto
------------------------------
```

Expected Output#2

- Functional sorting code from AI

Task Description#3

- Start with the vague prompt “Generate python code to calculate power bill” and improve it step-by-step

PROMPT: WRITE A PYTHON CODE TO CALCULATE THE POWER BILL

```
def calculate_power_bill():
    print("=== POWER BILL CALCULATOR ===")
    try:
        # Get customer details
        customer_name = input("Enter customer name: ")
        customer_id = input("Enter customer ID: ")

        # Get usage details
        current_reading = float(input("Enter current meter reading (kWh): "))
        previous_reading = float(input("Enter previous meter reading (kWh): "))

        # Calculate units consumed
        units_consumed = current_reading - previous_reading

        if units_consumed < 0:
            print("Error: Current reading cannot be less than previous reading!")
            return

        # Define rate structure (slab-based pricing)
        if units_consumed <= 100:
            rate_per_unit = 2.50
        elif units_consumed <= 300:
            rate_per_unit = 3.75
        elif units_consumed <= 500:
            rate_per_unit = 5.25
        else:
            rate_per_unit = 7.50

        # Calculate charges
        energy_charge = units_consumed * rate_per_unit
        fixed_charge = 150.00  # Monthly fixed charge
        tax_rate = 0.08  # 8% tax
        tax_amount = (energy_charge + fixed_charge) * tax_rate
        total_bill = energy_charge + fixed_charge + tax_amount

        # Display bill details
        print("\n" + "="*40)
```

```
lab3 > TASK3.py > ...
    def calculate_power_bill():
            total_bill = energy_charge + fixed_charge + tax_amount

            # Display bill details
            print("\n" + "="*40)
            print("POWER BILL STATEMENT")
            print("="*40)
            print(f"Customer Name: {customer_name}")
            print(f"Customer ID: {customer_id}")
            print(f"Previous Reading: {previous_reading:.2f} kWh")
            print(f"Current Reading: {current_reading:.2f} kWh")
            print(f"Units Consumed: {units_consumed:.2f} kWh")
            print(f"Rate per Unit: ₹{rate_per_unit:.2f}")
            print(f"Energy Charge: ₹{energy_charge:.2f}")
            print(f"Fixed Charge: ₹{fixed_charge:.2f}")
            print(f"Tax (8%): ₹{tax_amount:.2f}")
            print("-"*40)
            print(f"TOTAL BILL: ₹{total_bill:.2f}")
            print("="*40)

        except ValueError:
            print("Error: Please enter valid numeric values!")

    if __name__ == "__main__":
        calculate_power_bill()
```

OUTPUT:

```
Enter customer name: MEGHANA
Enter customer ID: 13
Enter current meter reading (kWh): 2000
Enter previous meter reading (kWh): 1000

========================================
POWER BILL STATEMENT
========================================
Customer Name: MEGHANA
Customer ID: 13
Previous Reading: 1000.00 kWh
Current Reading: 2000.00 kWh
Units Consumed: 1000.00 kWh
Rate per Unit: ₹7.50
Energy Charge: ₹7500.00
Fixed Charge: ₹150.00
Tax (8%): ₹612.00
----------------------------------------
TOTAL BILL: ₹8262.00
========================================
```

Expected Output#3

- Enhanced AI output with clearer prompts

Task Description#4

- Write structured comments to help AI generate two linked functions (e.g., login_user() and register_user()).

PROMPT: GENERATE A PYTHON FUNCTION TO LINK THE LOGIN USER AND REGISTER USER.

```
def user_auth_system():
    """Simple user authentication system with register and login functionality"""
    users = {}  # Dictionary to store username:password pairs
    def register_user():
        username = input("Enter username to register: ")
        if username in users:
            print("Username already exists!")
            return False
        password = input("Enter password: ")
        users[username] = password
        print(f"User '{username}' registered successfully!")
        return True
    def login_user():
        username = input("Enter username: ")
        password = input("Enter password: ")
        if username in users and users[username] == password:
            print(f"Welcome back, {username}!")
            return True
        else:
            print("Invalid username or password!")
            return False
    while True:
        print("\n=== USER AUTHENTICATION SYSTEM ===")
        print("1. Register")
        print("2. Login")
        print("3. Exit")
        choice = input("Enter your choice (1-3): ")
        if choice == '1':
            register_user()
        elif choice == '2':
            login_user()
        elif choice == '3':
            print("Goodbye!")
            break
        else:
            print("Invalid choice! Please try again.")
if __name__ == "__main__":
    user_auth_system()
```

OUTPUT:

```
Problems  Output  Debug Console  Terminal  Ports
3. Exit
Enter your choice (1-3): 2
Enter username: MEGHANA
Enter password: MEGHA
Invalid username or password!

=== USER AUTHENTICATION SYSTEM ===
1. Register
2. Login
3. Exit
Enter your choice (1-3): 1
Enter username to register: MEGHA
Username already exists!

=== USER AUTHENTICATION SYSTEM ===
1. Register
2. Login
3. Exit
Enter your choice (1-3): 2
Enter username: MEGHA
Enter password: MEGHANA
Welcome back, MEGHA!
```

Expected Output#4

- Consistent functions with shared logic

Task Description#5

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

PROMPT: GENERATE A PYTHON CODE TO CALCULATE TEMPERATURE CONVERSION.

```python
def temp_converter():
    print("Temperature Converter")
    print("1. Celsius to Fahrenheit")
    print("2. Fahrenheit to Celsius")
    print("3. Celsius to Kelvin")
    print("4. Kelvin to Celsius")
    print("5. Fahrenheit to Kelvin")
    print("6. Kelvin to Fahrenheit")

    choice = int(input("Enter choice (1-6): "))
    temp = float(input("Enter temperature: "))

    if choice == 1:
        result = (temp * 9/5) + 32
        print(f"{temp}°C = {result}°F")
    elif choice == 2:
        result = (temp - 32) * 5/9
        print(f"{temp}°F = {result}°C")
    elif choice == 3:
        result = temp + 273.15
        print(f"{temp}°C = {result}K")
    elif choice == 4:
        result = temp - 273.15
        print(f"{temp}K = {result}°C")
    elif choice == 5:
        result = (temp - 32) * 5/9 + 273.15
        print(f"{temp}°F = {result}K")
    elif choice == 6:
        result = (temp - 273.15) * 9/5 + 32
        print(f"{temp}K = {result}°F")
    else:
        print("Invalid choice!")

# Run the converter
if __name__ == "__main__":
    temp_converter()
```

OUTPUT:

```
Temperature Converter
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
3. Celsius to Kelvin
4. Kelvin to Celsius
5. Fahrenheit to Kelvin
6. Kelvin to Fahrenheit
Enter choice (1-6): 1
Enter temperature: 123
123.0°C = 253.4°F
```

Expected Output#5

- Code quality difference analysis for various prompts

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
Factorial Function (Task#1)	0.5
Sorting Function (Task#2)	0.5
Vogue Vs. Specific Prompting (Task #3)	0.5
Linked Functions (Task #4)	0.5
Temperature Conversion Function (Task #5)	0.5
Total	**2.5 Marks**