Course Code:AIPP

Assignment:4

Done by:Revathi.D(CSC)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** M.Tech. and MCA | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **Course Coordinator Name** | | | | Venkataramana Veeramsetty | | | | | |
| **Course Code** | | |  | **Course Title** | | AI Assisted Problem Solving Using Python | | | |
| **Year/Sem** | | | I/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week3 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | M.Tech. and MCA | | | |
| **AssignmentNumber:4.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques  **Lab Objectives:**   * To explore and apply different levels of prompt examples in AI-assisted code generation. * To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality. * To evaluate the impact of context richness and example quantity on AI performance. * To build awareness of prompt strategy effectiveness for different problem types.   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Use zero-shot prompting to instruct AI with minimal context. * Use one-shot prompting with a single example to guide AI code generation. * Apply few-shot prompting using multiple examples to improve AI responses. * Compare AI outputs across the three prompting strategies.   **Task Description#1**   * Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.   **Prompt:**   * “Write a Python function to check whether a given year is a leap year.”   **Code:**  def is\_leap\_year(year):      """Check if a given year is a leap year."""      if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):          return True      return False  # Example usage  year = 2024  print(f"{year} is a leap year: {is\_leap\_year(year)}")  **Expected Output#1**   * AI-generated function with no examples provided   Output:  PS C:\revathi-sru> python -u "c:\revathi-sru\task1.py"  2024 is a leap year: True  **Task Description#2**   * One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.   Prompt:  “Write a Python function to check whether a given year is a leap year.”  Code:  # One-shot example to guide the function:  # Input: 10  # centimeters  # Output: 3.937007874015748  # inches  (10 / 2.54)  **Expected Output#2**   * Function with correct conversion logic   Output:  Output: 3.937007874015748  # inches  (10 / 2.54)  **Task Description#3**   * Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”.   Prompt:  “Format names as ‘Last, First’. Examples:   * Input: ‘John Doe’ → Output: ‘Doe, John’ * Input: ‘Mary Ann Smith’ → Output: ‘Smith, Mary Ann’ * Input: ‘Rahul Kumar’ → Output: ‘Kumar, Rahul’   Code:  # formats full names as "Last, First"  def format\_name(full\_name: str) -> str:      s = full\_name.strip()      if not s:          return ""      parts = s.split()      if len(parts) == 1:          return parts[0]      first, last = parts[0], parts[-1]      return f"{last}, {first}"  # Examples  print(format\_name("Ada Lovelace"))             # -> "Lovelace, Ada"  print(format\_name("Alan Mathison Turing"))     # -> "Turing, Alan"  print(format\_name("  marie   skłodowska curie "))  # -> "curie, marie"  **Expected Output#3**   * Well-structured function respecting the examples   Output:  PS C:\revathi-sru> python -u "c:\revathi-sru\task3"  Lovelace, Ada  Turing, Alan  curie, marie  **Task Description#4**   * Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.   Prompt:   * “Write a Python function to count the number of vowels in a given string.”   Count the number of vowels in a given string. Examples: Input: ‘Hello’ → Output: 2 Input: ‘ChatGPT’ → Output: 1 Input: ‘OpenAI’ → Output: 4  Write a Python function that does this.”  Code:  def count\_vowels\_zero\_shot(text):      """Zero-shot approach: Direct implementation without examples"""      vowels = 'aeiouAEIOU'      return sum(1 for char in text if char in vowels)  def count\_vowels\_few\_shot(text):      """Few-shot approach: Implementation with example patterns      Examples used:      'hello' -> 2 vowels (e, o)      'APPLE' -> 2 vowels (A, E)      'why' -> 0 vowels      """      vowels = set('aeiouAEIOU')      count = 0      for char in text:          if char in vowels:              count += 1      return count  # Test both approaches  test\_strings = ['hello', 'APPLE', 'why', 'Beautiful', 'Python']  print("Zero-shot results:")  for text in test\_strings:      print(f"'{text}': {count\_vowels\_zero\_shot(text)} vowels")  print("\nFew-shot results:")  for text in test\_strings:      print(f"'{text}': {count\_vowels\_few\_shot(text)} vowels")  **Expected Output#4**   * Functional output and comparative reflection   Output:  'Beautiful': 5 vowels  'Python': 1 vowels  Few-shot results:  'hello': 2 vowels  'APPLE': 2 vowels  'why': 0 vowels  'Beautiful': 5 vowels  'Python': 1 vowels  **Task Description#5**   * Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.   Prompt: “Write a Python function that reads a .txt file and returns the number of lines.  Code: def count\_lines\_in\_file(file\_path):      """Reads a .txt file and returns the number of lines."""      try:          with open(file\_path, 'r') as file:              lines = file.readlines()              return len(lines)      except FileNotFoundError:          print(f"The file at {file\_path} was not found.")          return 0      except Exception as e:          print(f"An error occurred: {e}")          return 0  **Expected Output#5**   * Working file-processing function with AI-guided logic   **Output:**  **PS C:\revathi-sru> python -u "c:\revathi-sru\taskdup5.py"**  **You are given a file path. Return the number of lines in the file.**  **Examples:**  **- file path: /tmp/one\_line.txt; contents: 'hello\n' -> 1**  **- file path: /tmp/three\_lines.txt; contents: 'a\nb\nc\n' -> 3**  **- file path: /tmp/empty.txt; contents: '' -> 0**  **Now provide a Python function count\_lines(path: str) -> int that reads the file and returns the number of lines.**  **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** | | --- | --- | | Zero Shot (Task #1) | 2.5 | | One Shot (Task#2) | 2.5 | | Few Shot (Task#3 & Task #5) | 2.5 | | Comparison (Task#4) | 2.5 | | **Total** | **10 Marks** | | | | | | | Week3 - Monday |  |