|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** M. Tech/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** | | | Week1 - TUESDAY | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | M. Tech/MCA | | | |
| **AssignmentNumber:2.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | Lab 2: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI  **Lab Objectives:**   * To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab. * To understand and use Cursor AI for code generation, explanation, and refactoring. * To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI. * To perform code optimization and documentation using AI tools.   **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**   * Use Google Gemini in Colab to write a function that reads a CSV file and calculates mean, min, max.   **Expected Output#1**   * Functional code with output and screenshot   **Prompt:** Write a python function that reads a CSV file and calculates mean, min and max.          **Output:**    **Task Description#2**   * Compare Gemini and Copilot outputs for a palindrome check function.   **Expected Output#2**   * Side-by-side comparison and observations   **Prompt:** Write a python function using Gemini and Copilot for Palindrome check.  **Gemini Code:**        **Gemini Code Output:**    **Code Explanation:**   1. **Input guard:** check for None and handle empty input to avoid errors. 2. **Normalize:** remove all non-alphanumeric characters using a regex and convert to lowercase (e.g., "A man, a plan" -> "amanaplan"). 3. **Compare:** test equality between the normalized string and its reverse (cleaned == cleaned[::-1]). 4. **Return useful result:** return a tuple (is\_palindrome, normalized\_string) so callers can both see the result and the exact string used for the check.   **Copilot Code:**      **Copilot Code Output:**    **Code Explanation:**   1. **Compact normalization:** build the cleaned string with a comprehension or generator: cleaned = ''.join(ch.lower() for ch in s if ch.isalnum()). 2. **Palindrome test:** check cleaned == cleaned[::-1] in one line (often returned directly). 3. **Minimal API:** function typically returns only a boolean (concise and easy to use inline). 4. **Fast and idiomatic:** favors brevity and Python idioms; slightly less verbose about edge cases (e.g., may not return the normalized string).   **Observations:**  1. Both Gemini and Copilot correctly suggested logic to reverse the string and compare it with the original.  2. Copilot’s code was concise, using slicing (str[::-1]), while Gemini provided a more descriptive approach with loop-based reversal.  3. Gemini emphasized input validation and case normalization, improving robustness.  4. Copilot focused on quick implementation, making it efficient for simple palindrome checks.  **Side by Side Differences table:**    **Task Description#3**   * Ask Gemini to explain a Python function (to calculate area of various shapes) line by line..   **Expected Output#3**   * Detailed explanation with code snippet   **Promt: Write a Python function to calculate area of various shapes and allow user to give user input.**  **Code:**      **Output:**    **Explanation:**  **Circle**  **Code Logic:**  radius = float(input("Enter radius: "))  area = 3.14159 \* radius \*\* 2  print(f"Area of circle = π \* r² = {area:.2f}")  **Explanation:**   * The user is prompted to enter the radius of the circle. * The formula for the area of a circle is , where π (pi) ≈ 3.14159. * The program computes 3.14159 \* radius \*\* 2 — the \*\* operator squares the radius. * Finally, it displays the result with two decimal places.   **Rectangle**  **Code Logic:**  length = float(input("Enter length: "))  width = float(input("Enter width: "))  area = length \* width  print(f"Area of rectangle = length \* width = {area:.2f}")  **Explanation:**   * The user inputs the length and width. * The area is simply the product of the two dimensions. * This formula works for all rectangles regardless of orientation.   **Triangle**  **Code Logic:**  base = float(input("Enter base: "))  height = float(input("Enter height: "))  area = 0.5 \* base \* height  print(f"Area of triangle = ½ \* base \* height = {area:.2f}")  **Explanation:**   * The user enters base and height. * The area of a triangle formula is . * The code multiplies 0.5 \* base \* height to calculate and display the result.   **Square**  **Code Logic:**  side = float(input("Enter side length: "))  area = side \*\* 2  print(f"Area of square = side² = {area:.2f}")  **Explanation:**   * The user provides the side length. * Since all sides are equal, area = side × side or side \*\* 2. * The result is printed with the proper formula format.   **Trapezoid**  **Code Logic:**  base1 = float(input("Enter base 1: "))  base2 = float(input("Enter base 2: "))  height = float(input("Enter height: "))  area = 0.5 \* (base1 + base2) \* height  print(f"Area of trapezoid = ½ \* (base1 + base2) \* height = {area:.2f}")  **Explanation:**   * A trapezoid has two parallel bases and a height. * Formula: . * The code takes both bases and height from the user, adds the bases, multiplies by height, and halves the result.   **Ellipse**  **Code Logic:**  a = float(input("Enter semi-major axis (a): "))  b = float(input("Enter semi-minor axis (b): "))  area = 3.14159 \* a \* b  print(f"Area of ellipse = π \* a \* b = {area:.2f}")  **Explanation:**   * An ellipse has two axes — a (semi-major) and b (semi-minor). * Formula: . * The program multiplies π by both axes and prints the calculated area.   Task Description#4   * Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of squares).   **Expected Output#4**   * Screenshots of working environments with few prompts to generate python code   **Prompt 1:** Write a python function to find sum of squares  **Prompt 2:** Allow user to give user input.      **Output:**    **Explanation:**   * Cursor AI was installed and configured as an intelligent coding assistant within the development environment. * It was used to automatically generate a Python function that calculates the sum of squares of given numbers. * The tool suggested clean, optimized, and readable code with proper error handling for user input. * Cursor AI reduced manual coding effort by providing instant suggestions and improving code quality. * The generated function was tested successfully by taking user input and displaying the computed result.   **Task Description#5**   * Student need to write code to calculate sum of add number and even numbers in the list   **Expected Output#5**   * Refactored code written by student with improved logic   **Prompt:** Generate a python code to find sum of even and odd numbers and ask user to give user input.  **Code:**    **Output:**    **Explanation:**  1. The function sum\_odd\_even() separates odd and even numbers using list comprehensions and computes their sums individually.  2. User inputs a list of numbers separated by spaces.  3. The program then displays the total sum of odd numbers and even numbers separately.  4. The code is simple, efficient, and clean.  **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** | | --- | --- | | Successful Use of Gemini in Colab (Task#1 & #2) | 2.5 | | Code Explanation Accuracy (Gemini) (Task#3) | 2.5 | | Cursor AI Setup and Usage (Task#4) | 2.5 | | Refactoring and Improvement Analysis (Task#5) | 2.5 | | **Total** | **10 Marks** | | | | | | | Week1 - TuesDay |  |