|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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, 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:** **Give a python function for a palindrome check function by dealing with input and giving output.**  **Gemini :**  **A screenshot of a computer**  **Output:**  **A black background with white text  AI-generated content may be incorrect.**  **Explanation:**  1. This function checks if a given string reads the same forwards and backwards.  2. It removes spaces and converts all letters to lowercase before comparison.  3. The cleaned text is compared with its reverse to decide if it’s a palindrome.  4. The script directly takes user input and prints a formatted message using f-strings.  5. It’s simpler and easy to read but less comprehensive in handling punctuation.  **Copilot:**  **A computer screen shot of a program code  AI-generated content may be incorrect.**  **Output:**  **A black background with yellow and blue text  AI-generated content may be incorrect.**  Explanation:  1. The program defines a function that checks if a string is a palindrome, ignoring punctuation, spaces, and case.  2. It cleans the input by keeping only alphanumeric characters and converting them to lowercase.  3. The cleaned string is compared with its reverse to check for palindrome status.  4. A command-line interface lets users input text and get a clear “Yes” or “No” output.  5. It’s a robust and well-documented implementation for general use.   | **Aspect** | **Copilot** | **Gemini** | | --- | --- | --- | | **Palindrome logic** | Cleans input by removing all non-alphanumeric characters and ignores case. | Only removes spaces and converts text to lowercase. | | **Code robustness** | More accurate — handles punctuation and symbols properly. | Simpler, but fails for strings with punctuation (e.g., “A man, a plan, a canal: Panama”). | | **Docstring & examples** | Contains detailed examples showing usage and expected outputs. | Uses structured docstring (Args, Returns) for clarity. | | **CLI usage** | Uses a clear if \_\_name\_\_ == "\_\_main\_\_": block. | Runs input/output directly in the script body (less modular). | | **Output format** | Simple printed responses. | Uses f-strings for formatted output. |   **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   **Prompt:** Explain a Python function (to calculate area of various shapes) line by line.. Give a detailed explanation with code snippet. It should take input and give output. The lines of code should be as less as possible.  **A screenshot of a computer program  AI-generated content may be incorrect.**  **A computer screen shot of a program  AI-generated content may be incorrect.**  **Output:**  **A black background with white text  AI-generated content may be incorrect.**  **Explanation:**   * Line-by-line explanation of the calculate\_area function: * import math: Imports the math module for pi. * def calculate\_area(shape, dimensions): Defines the function with shape (str) and dimensions (dict). * """Docstring explaining the function, args, and returns.""" * shape = shape.lower(): Converts shape input to lowercase for case-insensitivity. * if shape == 'circle' and 'radius' in dimensions:: Checks if shape is circle and radius is provided. * return math.pi \* (dimensions['radius'] \*\* 2): Calculates and returns circle area. * elif shape == 'square' and 'side' in dimensions:: Checks if shape is square and side is provided. * return dimensions['side'] \* dimensions['side']: Calculates and returns square area. * elif shape == 'rectangle' and 'length' in dimensions and 'width' in dimensions:: Checks if shape is rectangle and length/width are provided. * return dimensions['length'] \* dimensions['width']: Calculates and returns rectangle area. * elif shape == 'triangle' and 'base' in dimensions and 'height' in dimensions:: Checks if shape is triangle and base/height are provided. * return 0.5 \* dimensions['base'] \* dimensions['height']: Calculates and returns triangle area. * else:: If shape is unsupported or dimensions are missing. * print(f"Error: Unsupported shape '{shape}' or missing dimensions."): Prints an error message. * return None: Returns None as area cannot be calculated. * Example usage with input and output: * The following lines handle user input and display output. * shape\_input = input(...).lower(): Gets shape input from user and converts to lowercase. * dimensions\_input = {}: Initializes an empty dictionary to store dimensions. * The following 'if/elif' blocks handle getting specific dimensions based on the shape: * if shape\_input == 'circle': Prompts for radius. * elif shape\_input == 'square': Prompts for side. * elif shape\_input == 'rectangle': Prompts for length and width. * elif shape\_input == 'triangle': Prompts for base and height. * try...except ValueError: Each input section uses a try-except block to handle non-numeric input. * dimensions\_input[...] = ...: Adds the valid input dimensions to the dictionary. * area = calculate\_area(shape\_input, dimensions\_input): Calls the function to calculate the area. * if area is not None:: Checks if the area was successfully calculated. * print(f"The area of the {shape\_input} is: {area}"): 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:** Generate a python code to find sum of squares after taking input from user.  **A screenshot of a computer  AI-generated content may be incorrect.**  **Output:**  **A screenshot of a computer  AI-generated content may be incorrect.**  **Task Description#5**   * Student need to write code to calculate sum of odd number and even numbers in the list   **Expected Output#5**   * Refactored code written by student with improved logic   **Prompt:** Generate a code to calculate the sum of odd numbers and even numbers in the list and take input from user      **Output:**  **A computer screen with numbers and letters  AI-generated content may be incorrect.**  **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 |  |