|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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         **Task Description#2**   * Compare Gemini and Copilot outputs for a palindrome check function.   **Expected Output#2**   * Side-by-side comparison and observations       **Expected Output:**     |  |  |  | | --- | --- | --- | | **Feature** | **Gemini Output (Image 1)** | **Copilot Output (Image 2)** | | **Function Name** | Removes all special characters and makes everything lowercase. | Removes spaces and makes everything lowercase. | | **Approach** | Cleans the text by keeping only letters and numbers, then converts to lowercase. | Removes spaces only, then converts to lowercase. | | **Cleaning Step** | Uses a loop to remove unwanted symbols and make text lowercase. | Uses split and join to remove spaces and then lowercase. | | **Reversal Check** | Compares the cleaned text with its reverse using slicing. | Compares the cleaned text with its reverse using slicing. | | **Docstring / Comments** | Has a proper explanation of what the function does and what it returns. | Has short comments explaining the code logic. | | **Test Cases** | Examples: “Racecar”, “Madam, I’m Adam”, “Hello world”. | Example: “A man a plan a canal Panama”. | | **Output** | Gives True, True, and False for the examples. | Says “A man a plan a canal Panama” is a palindrome. | | **Focus / Accuracy** | Works well even when there are punctuations or symbols. | Works only when there are spaces (fails if symbols are present). |     **IMAGE**    **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   **Expected Output:**        **Code snippet:**  import math # Imports the math module to use mathematical functions like pi.  def calculate\_area(shape, \*\*kwargs): # Defines a function named calculate\_area that takes 'shape' as a string and a variable number of keyword arguments (\*\*kwargs) for dimensions.    """Calculates the area of various shapes. # Docstring: explains what the function does.    Args: # Docstring section: describes the arguments the function takes.      shape: The name of the shape (e.g., "circle", "rectangle", "triangle"). # Describes the 'shape' argument.      \*\*kwargs: Keyword arguments for the shape's dimensions. # Describes the \*\*kwargs argument.    Returns: # Docstring section: describes what the function returns.      The area of the shape, or None if the shape is not supported or # Describes the return value.      dimensions are missing.    """    if shape == "circle": # Checks if the shape is "circle".      radius = kwargs.get("radius") # Gets the value for the 'radius' keyword argument. Returns None if 'radius' is not provided.      if radius is not None: # Checks if 'radius' was provided (is not None).        return math.pi \* radius\*\*2 # Calculates and returns the area of a circle (pi \* radius squared).      else: # If 'radius' was not provided.        print("Error: Radius is required for a circle.") # Prints an error message.        return None # Returns None indicating an error.    elif shape == "rectangle": # Checks if the shape is "rectangle".      length = kwargs.get("length") # Gets the value for the 'length' keyword argument.      width = kwargs.get("width") # Gets the value for the 'width' keyword argument.      if length is not None and width is not None: # Checks if both 'length' and 'width' were provided.        return length \* width # Calculates and returns the area of a rectangle (length \* width).      else: # If 'length' or 'width' was not provided.        print("Error: Length and width are required for a rectangle.") # Prints an error message.        return None # Returns None indicating an error.    elif shape == "triangle": # Checks if the shape is "triangle".      base = kwargs.get("base") # Gets the value for the 'base' keyword argument.      height = kwargs.get("height") # Gets the value for the 'height' keyword argument.      if base is not None and height is not None: # Checks if both 'base' and 'height' were provided.        return 0.5 \* base \* height # Calculates and returns the area of a triangle (0.5 \* base \* height).      else: # If 'base' or 'height' was not provided.        print("Error: Base and height are required for a triangle.") # Prints an error message.        return None # Returns None indicating an error.    else: # If the shape is not "circle", "rectangle", or "triangle".      print(f"Error: Shape '{shape}' is not supported.") # Prints an error message indicating the shape is not supported.      return None # Returns None indicating an error.  # Example Usage: # Comment indicating the start of example usage.  print(f"Area of circle with radius 5: {calculate\_area('circle', radius=5)}") # Calls the function for a circle with radius 5 and prints the result.  print(f"Area of rectangle with length 4 and width 6: {calculate\_area('rectangle', length=4, width=6)}") # Calls the function for a rectangle and prints the result.  print(f"Area of triangle with base 3 and height 7: {calculate\_area('triangle', base=3, height=7)}") # Calls the function for a triangle and prints the result.  print(f"Area of square with side 4: {calculate\_area('square', side=4)}") # Calls the function for an unsupported shape ("square") and prints the result (will print an error message and None).  print(f"Area of circle without radius: {calculate\_area('circle')}") # Calls the function for a circle without providing the required 'radius' and prints the result (will print an error message and None).  **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             **Sample Program:**    **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       **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 |  |