**Name: K. Saketh Varma Roll No.: 2403A510B6 Batch: 04**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | | |  | | --- | | Dr. V. Venkataramana (Co-ordinator) | | Dr. T. Sampath Kumar | | Dr. Pramoda Patro | | Dr. Brij Kishor Tiwari | | Dr.J.Ravichander | | Dr. Mohammand Ali Shaik | | Dr. Anirodh Kumar | | Mr. S.Naresh Kumar | | Dr. RAJESH VELPULA | | Mr. Kundhan Kumar | | Ms. Ch.Rajitha | | Mr. M Prakash | | Mr. B.Raju | | Intern 1 (Dharma teja) | | Intern 2 (Sai Prasad) | | Intern 3 (Sowmya) | | NS\_2 ( Mounika) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week1 - Tuesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | | 24CSBTB01 To 24CSBTB39 | | | |
| **AssignmentNumber:1.2**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 1: Environment Setup – GitHub Copilot and VS Code Integration  **Lab Objectives:**   * To install and configure GitHub Copilot in Visual Studio Code. * To explore AI-assisted code generation using GitHub Copilot. * To analyze the accuracy and effectiveness of Copilot's code suggestions. * To understand prompt-based programming using comments and code context   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Set up GitHub Copilot in VS Code successfully. * Use inline comments and context to generate code with Copilot. * Evaluate AI-generated code for correctness and readability. * Compare code suggestions based on different prompts and programming styles.   **Task Description#1**   * Write a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and allow Copilot to complete it.   **Expected Output#1**   * A function that correctly returns True for phrases like "A man a plan a canal Panama"   **Prompt#1**  🡪The task is that if we give the input to check the condition whether the string is a Palindrome or not if the string is palindrome then print true else false so here if I give “A Man a plan a canal a panama” if the string is palindrome such that the output is true else false.  **Output#1**    **Observation#1**  🡪This Python code defines a function is\_palindrome(s) that checks whether a given string is a palindrome, ignoring spaces, punctuation, and letter case. Inside the function, it first removes all non-alphanumeric characters and converts the remaining characters to lowercase using a list comprehension and isalnum() check. The cleaned string is then compared to its reverse (cleaned[::-1]), and the function returns True if they match or False otherwise. The program prompts the user to enter a string, stores it in input\_str, and prints the result of is\_palindrome(input\_str). In the example shown, the user enters "madam", and the output is True, indicating that the input is indeed a palindrome. This approach ensures that palindromes with mixed case and punctuation, such as “A man, a plan, a canal: Panama,” are correctly identified.  **Task Description#2**   * Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring   **Expected Output#2**   * AI completes the function logic using loop or recursion with accurate output   **Prompt#2**  🡪write a python code to print the fibonnaci series such that the output will be in list.  **Output#2**    **Observation#2**  **🡪**This Python code generates and prints the first n terms of the Fibonacci series using recursion. The function fibonacci\_recursive(n) calculates the nth Fibonacci number by recursively summing the two preceding numbers, with base cases for n <= 1. The function fibonacci\_series\_recursive(n) uses a list comprehension to call fibonacci\_recursive(i) for all values from 0 to n-1, creating a list of the series. The variable n is set to 10, so the program prints the first 10 Fibonacci numbers: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]. While the code correctly produces the result, it is computationally inefficient for large n due to repeated recursive calculations without memoization.  **Task Description#3**   * Write a comment like # Function to reverse a string and use Copilot to generate the function.   **Expected Output#3**   * Auto-completed reverse function   **Prompt#3**  🡪Write a python code such that the input string is to be reversed and it will display the reversed string.  **Output#3**    **Observation#3**  **🡪**This Python code reverses a given string entered by the user using a loop and dynamic string concatenation. The program first prompts the user to input a string and initializes an empty string reversed\_str to store the reversed result. It then iterates over each character in the input string, placing each character before the existing reversed str content, effectively building the reversed string one character at a time. Finally, it prints the reversed string with a descriptive message. In the provided example, the user enters "copilot", and the program outputs "tolipoc", correctly reversing the input. This method is simple and does not rely on Python’s slicing features, making it a good demonstration of manual string reversal using loops.  **Task Description#4**   * Generate a program that simulates a basic calculator (add, subtract, multiply, divide). Write the comment: # Simple calculator with 4 operations and let AI complete it.   **Expected Output#4**   * Fully working calculator with input/output and operator selection logic   **Prompt#4**  **🡪**Write a python code that works as a basic calculator such that the output will ask multiple choices like add , subtract ,multiply, divide and closing calculator.  **Output#4**    **Observation#4**  **🡪**This Python code implements a simple command-line calculator that performs addition, subtraction, multiplication, and division based on user input. It defines four separate functions—add(), subtract(), multiply(), and divide()—to handle each operation, with the division function including a check to prevent division by zero. The main() function runs an infinite loop that displays a menu of operations, takes the user’s choice, and either performs the calculation or exits the program if the choice is 5. For valid operation choices (1–4), it prompts the user for two numbers, handles invalid numeric inputs using try-except, and uses a dictionary (ops) to map the choice to the corresponding function. The result is then displayed. In the given example, the user selects addition (choice 1), enters 12 and 34, and receives the correct output 46.0. The program continues running until the user selects the exit option. This structure makes the calculator reusable, user-friendly, and resilient to incorrect inputs.  **Task Description#5**   * Use a comment to instruct AI to write a function that reads a file and returns the number of lines..   **Expected Output#5**   * Functional implementation using open() or with open() and readlines()   **Prompt#5**  **🡪** write a python code that reads a file and returns the number of line using dynamic input.  **Output#5**    **Observation#5**  **🡪**This Python code defines a function count\_lines\_with\_string() that counts how many lines in a given file contain a specified search string, with an option for case-insensitive matching. It opens the file in read mode using UTF-8 encoding, iterates through each line, and if case\_insensitive is True, both the line and search string are converted to lowercase before comparison. If the search string is found in the line, the count is incremented. The code includes error handling for missing files (FileNotFoundError) and general exceptions, printing appropriate error messages. In the main block, the program takes user inputs for the filename, the string to search, and whether the search should be case-insensitive. It then calls the function and prints the result. In the example execution, the user searches for "html" in a non-existent file named "html", resulting in an error message indicating the file was not found, and the count remains zero. This program demonstrates file reading, string searching, optional case-insensitivity, and robust exception handling.  **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** | | --- | --- | | Task #1 | 0.5 | | Task #2 | 0.5 | | Task #3 | 0.5 | | Task #4 | 0.5 | | Task #5 | 0.5 | | **Total** | **2.5 Marks** | | | | | | | Week1 - wednesday |  |