|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** B. Tech | | | | **Assignment Type: Lab** | | | **Academic Year:**2025-2026 | | |
| **Course Coordinator Name** | | | | 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 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | **Week1 - Monday** | **Time(s)** | | 24CSBTB01 To 24CSBTB39 | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All batches | | | |
| **Assignment Number:1.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***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 0   * Install and configure GitHub Copilot in VS Code. Take screenshots of each step.   Expected Output        Task 1: Factorial without Functions   * Description: Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions (using loops directly in the main code). * Expected Output: * Code:       Task 2: Improving Efficiency   * Description: Examine the Copilot-generated code from Task 1 and demonstrate how its efficiency can be improved (e.g., removing unnecessary variables, optimizing loops). * Expected Output:   Original:    Here's how you can improve the efficiency and clarity of your factorial calculation:   * You can use Python's built-in math.factorial() for optimal performance, but if you want to stick to a loop: * The current code is already quite efficient for small values of [n](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html). * You can remove the initialization of [factorial](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) outside the loop by using a generator expression with reduce from functools. * If you want to keep it simple and readable, you can use a single line with prod from math (Python 3.8+).   Task 3: Factorial with Functions   * Description: Use GitHub Copilot to generate a Python program that calculates the factorial of a number using a user-defined function. * Expected Output:     **Steps to Generate the**[factorial](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**Function**   1. **Function Definition** Defined a function named [factorial](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) that takes a single argument [n](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html). 2. **Input Validation** Checked if [n](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) is negative. If so, raised a [ValueError](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) because factorial is not defined for negative numbers. 3. **Initialization** Initialized a variable [result](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) to 1. This will hold the computed factorial value. 4. **Loop for Calculation** Used a for loop starting from 2 up to and including [n](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html). Multiplied [result](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) by each integer in this range to compute the factorial. 5. **Return Statement** Returned the final value of [result](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) as the factorial of [n](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html). 6. **Main Block for User Input** Used the [if \_\_name\_\_ == "\_\_main\_\_":](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) block to allow the script to be run directly. 7. **Error Handling** Wrapped user input and function call in a try-except block to catch and display errors (e.g., invalid input or negative numbers). 8. **Output** Printed the result in a formatted string if no error occurred.   This approach ensures correctness, handles invalid input, and provides clear output for the user.  Task 4: Comparative Analysis – With vs Without Functions   * Description: Differentiate between the Copilot-generated factorial program with functions and without functions in terms of logic, reusability, and execution. * Expected Output:   **1. Logic**   * **Without Functions:** The logic is written directly in the script. The loop and calculation are performed in the global scope, making the code linear and less modular. * **With Functions:** The logic is encapsulated in a [factorial(n)](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) function. Input validation and calculation are separated, making the code more organized and easier to understand.   **2. Reusability**   * **Without Functions:** The code cannot be reused easily. To calculate another factorial, you must copy-paste or rewrite the loop. * **With Functions:** The [factorial(n)](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) function can be called multiple times with different arguments, making it reusable in other scripts, modules, or larger projects.   **3. Execution**   * **Without Functions:** Executes sequentially from top to bottom. Only calculates the factorial for the user input once. * **With Functions:** Uses a main block ([if \_\_name\_\_ == "\_\_main\_\_":](vscode-file://vscode-app/c:/Users/vemab/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)) for execution. The function can be imported and used elsewhere, and error handling is more robust.   **Summary:** Using functions improves modularity, reusability, and maintainability, while the non-function version is simpler but less flexible and harder to extend.  Task 5: Iterative vs Recursive Factorial   * Description: Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function. * Expected Output:     **Submission Requirements**   1. Generate code for each task with comments. 2. Screenshots of Copilot suggestions. 3. Comparative analysis reports (Task 4 and Task 5). 4. Sample inputs/outputs demonstrating correctness.   **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 Setup of Copilot | 0.5 | | Comparative Analysis – With vs Without Functions | 1 | | Iterative vs Recursive Factorial | 1 | | **Total** | **2.5 Marks** | | | | | | | Week1 - Monday |  |