|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Q.No.** | **Name: H.V.S.Vyshnavi**  **Batch : 06**  **h.no: 2403a51182** | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 10**  **Task 1: Syntax and Error Detection**  **PROMPT:**  I have a Python script with syntax, indentation, and variable errors. Please identify and fix them  # buggy\_code\_task1.py  def add\_numbers(a, b)  result = a + b  return reslt  print(add\_numbers(10 20))  **CODE:**    **Output**:    **Observation:**   Missing Colon in Function Definition:   * Original: def add\_numbers(a, b) * Issue: Python function definitions require a colon (:) at the end of the def line to indicate the start of the function's code block. * Fix: def add\_numbers(a, b):    Incorrect Indentation:   * Original: The lines result = a + b and return reslt were notproperly indented under the function definition. * Issue: Python uses indentation to define code blocks. All statements within a function must be indented consistently.   Fix: The lines result = a + b and return result have been indented to align correctly with the function definition  **Task 2: Logical and Performance Issue Review**  **PROMPT:**  I have a Python script that finds duplicate numbers in a list, but the logic is inefficient because it uses nested loops. Please optimize the code so that it still produces the correct result but runs more efficiently  **CODE**:    **Output**:    **Optimised version:**    **Task 3: Code Refactoring for Readability**  **PROMPT:**  I have a Python script that calculates the factorial of a number, but the code is messy and not PEP 8–compliant. Please refactor it into a clean, well-structured version with:   * Proper indentation and formatting. * A meaningful function name (calculate\_factorial). * Clear variable naming. * A docstring explaining the function.   **CODE:**    **Output:**    **Observation:**   Renamed function to calculate\_factorial (descriptive, PEP 8 friendly).   Added **docstring** with parameters, return type, and error handling.   Used proper **indentation** and spacing around operators.   Used a more descriptive variable name (result instead of x).   Added a **validation check** for negative inputs.  **Task 4: Security and Error Handling Enhancement**  **PROMPT:**  I have a Python script that fetches user data from a SQLite database. The current code is unsafe because it uses string formatting in SQL queries, which makes it vulnerable to SQL injection. Please:   * Use parameterized queries (? placeholders) instead of string concatenation. * Add try–except blocks to handle database errors gracefully. * Include input validation before executing the query.   Refactor the code to follow clean practices.  **CODE:**    **Output:**    **Observation:**  **Parameterized queries (?)** → prevent SQL injection.   * **Exception handling** for database errors (sqlite3.Error) and unexpected issues. * **Input validation** → ensures only numeric IDs are accepted. * **finally block** → always closes DB connection safely.   **Task 5: Automated Code Review Report Generation**  **PROMPT:**  I have a Python script that performs basic arithmetic operations, but it is messy and not PEP 8–compliant. Please generate a review report identifying issues such as:   * Missing docstrings. * Inconsistent formatting and indentation. * Inline return statements without readability. * Missing error handling (division by zero). * Non-descriptive function and variable names. * Suggestions for improving readability and PEP 8 compliance.   After that, provide a refactored version of the code.  **CODE:**      **Output**:    **Observation**:   Rename function and parameters for clarity (calculate, operation).   Add docstrings with parameter, return, and exception details.   Implement **error handling** for invalid operations and division by zero.   Follow **PEP 8 formatting** (spaces, indentation, line breaks).   Use dictionary-based dispatching to simplify logic.   Add proper entry point with if \_\_name\_\_ == "\_\_main\_\_":. |  |  |