|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **AI ASSISTED CODING**  **Name: Cholluri Srikruthi**  **Htno:2403A510H0**  **ASSIGNMENT:10.3**  **BATCH:06** |  |  |
|  | 1 | TASK1#  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:  A computer screen with text  AI-generated content may be incorrect.  OUTPUT:  A black background with blue and green text  AI-generated content may be incorrect.  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 not properly 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  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:    OBSERVATION:   The original code used two nested loops (O(n²) time complexity) to compare every element with every other element.   In the optimized code, we use two sets (seen and duplicates) to track numbers efficiently.   * seen keeps track of elements we’ve already encountered. * If a number is already in seen, it gets added to duplicates.    This reduces the time complexity to O(n) and makes the solution much faster for large input lists.   The output remains the same:  Task 3  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:  A screenshot of a computer program  AI-generated content may be incorrect.  OUTPUT:  A black screen with blue text  AI-generated content may be incorrect.  OBSERVATION:   Function name changed from c → calculate\_factorial for clarity.   Variable x renamed to result, making the purpose more descriptive.   PEP 8 formatting applied: proper indentation, spaces around operators, blank lines for readability.   Docstring added to explain parameters, return type, and purpose.   Loop logic preserved but made more readable with result \*= i.  Task 4:  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:   Exception Handling:   * Added try–except to catch sqlite3.Error. * Ensures the program doesn’t crash on DB errors.    Input Validation:   * Checked user\_input.isdigit() before converting to integer. * Prevents invalid input like "abc" from reaching the query.    Resource Management:   * Used finally to close the DB connection safely.   Task 5: Automated Code Review Report Generation  Task: Generate a review report for this messy code.  # buggy\_code\_task5.py  def calc(x,y,z):  if z=="add":  return x+y  elif z=="sub": return x-y  elif z=="mul":  return x\*y  elif z=="div":  return x/y  else: print("wrong")    print(calc(10,5,"add"))  print(calc(10,0,"div"))  Expected Output:  AI-generated review report should mention:   * + Missing docstrings   + Inconsistent formatting (indentation, inline return)   + Missing error handling for division by zero   + Non-descriptive function/variable names   + Suggestions for readability and PEP 8 compliance   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:  **Issues in Original Code:**   1. **Missing docstrings** – The function has no explanation of purpose, arguments, or return values. 2. **Inconsistent formatting** – Mixed inline and block returns (elif z=="sub": return x-y). 3. **Division by zero** – No error handling, which can cause runtime exceptions. 4. **Non-descriptive names** – Function calc and parameter z are not descriptive; replaced with calculate and operation. 5. **PEP 8 Violations** – Missing spaces after commas, no blank lines between function and calls, inconsistent indentation. |  |  |