ASSIGNMENT 10.4

Task 1: AI-Assisted Syntax and Code Quality Review

Scenario

You join a development team and are asked to review a junior developer's Python script that fails to run correctly due to basic coding mistakes. Before deployment, the code must be corrected and standardized.

Task Description

You are given a Python script containing:

• Syntax errors

• Indentation issues

• Incorrect variable names

• Faulty function calls

Use an AI tool (GitHub Copilot / Cursor AI) to:

• Identify all syntactic and structural errors

• Correct them systematically

• Generate an explanation of each fix made

Expected Outcome

• Fully corrected and executable Python code

• AI-generated explanation describing:

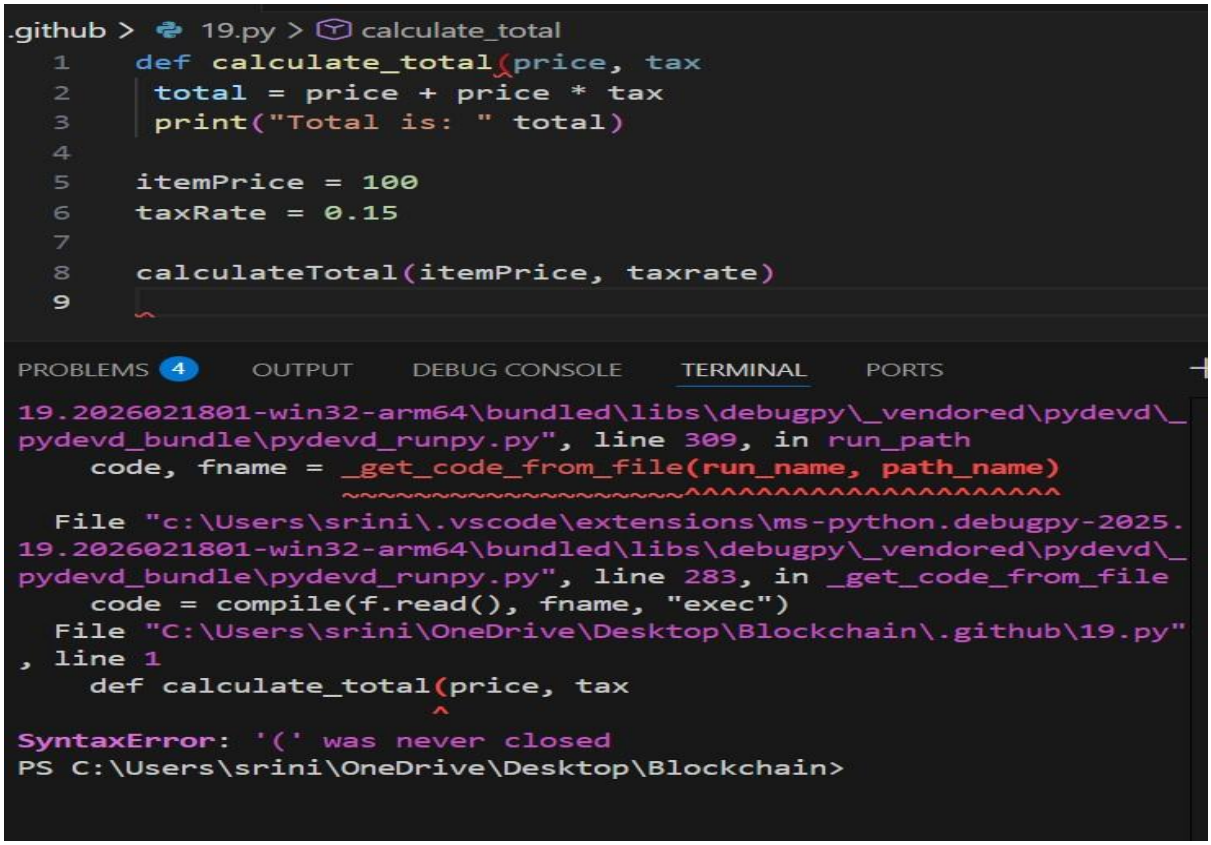
o Syntax fixes

o Naming corrections

o Structural improvements

• Clean, readable version of the script

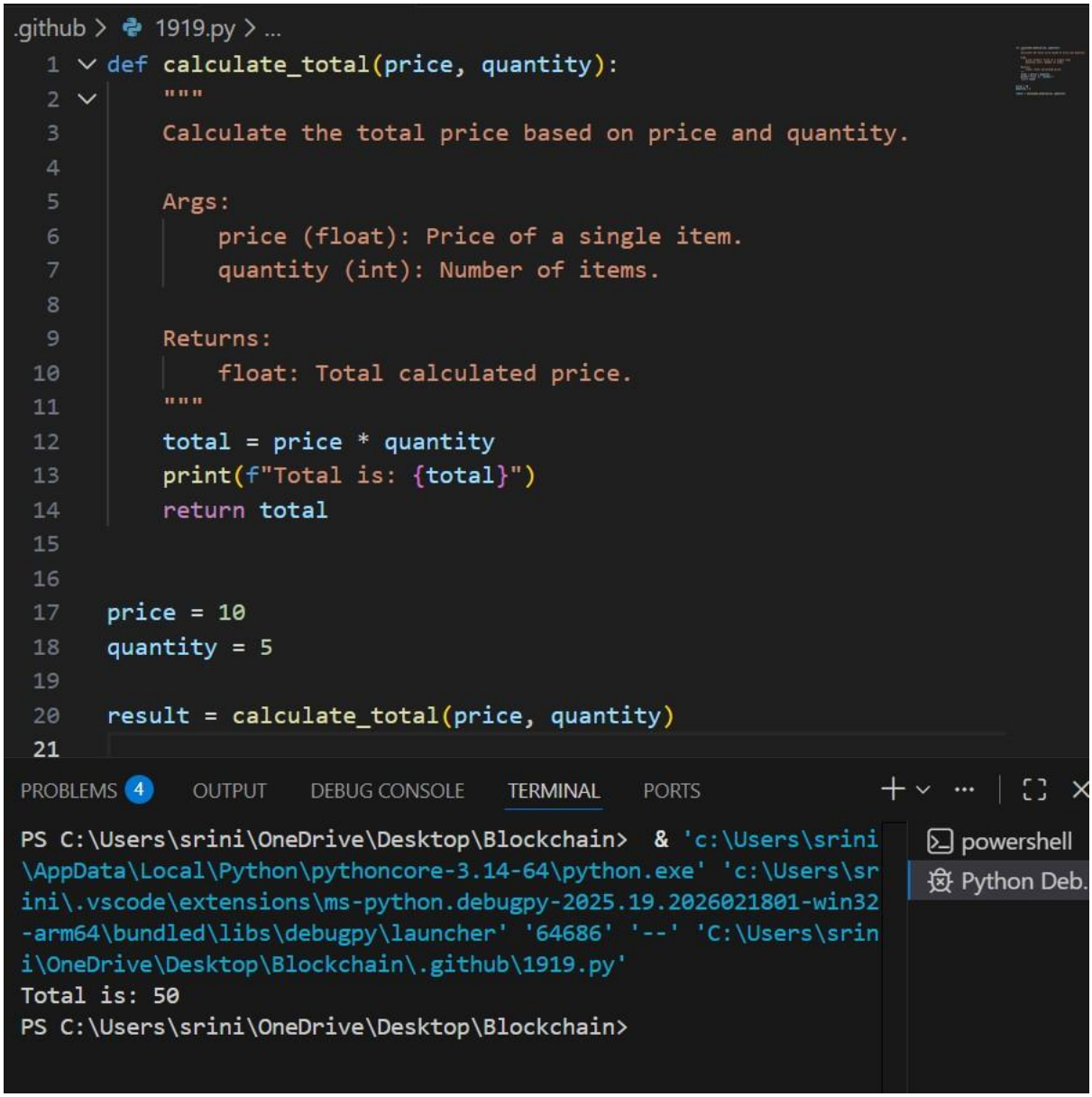
TASK-1:



- Missing colon (:) after function definition
- Incorrect indentation
- Misspelled variable quntity
- Incorrect print syntax
- Function name mismatch (calculateTotal vs calculate_total)
- return outside properly indented block

Prompt Used

“Review the following Python script. Identify syntax errors, indentation issues, naming problems, and structural mistakes. Correct the code and explain each fix according to PEP 8 standards.”



Task 2: Performance-Oriented Code Review

Scenario

A data processing function works correctly but is inefficient and slows

down the system when large datasets are used.

Task Description

You are provided with a function that identifies duplicate values in a list

using inefficient nested loops.

Using AI-assisted code review:

• Analyze the logic for performance bottlenecks

• Refactor the code for better time complexity

• Preserve the correctness of the output

Ask the AI to explain:

• Why the original approach was inefficient

• How the optimized version improves performance

Expected Outcome

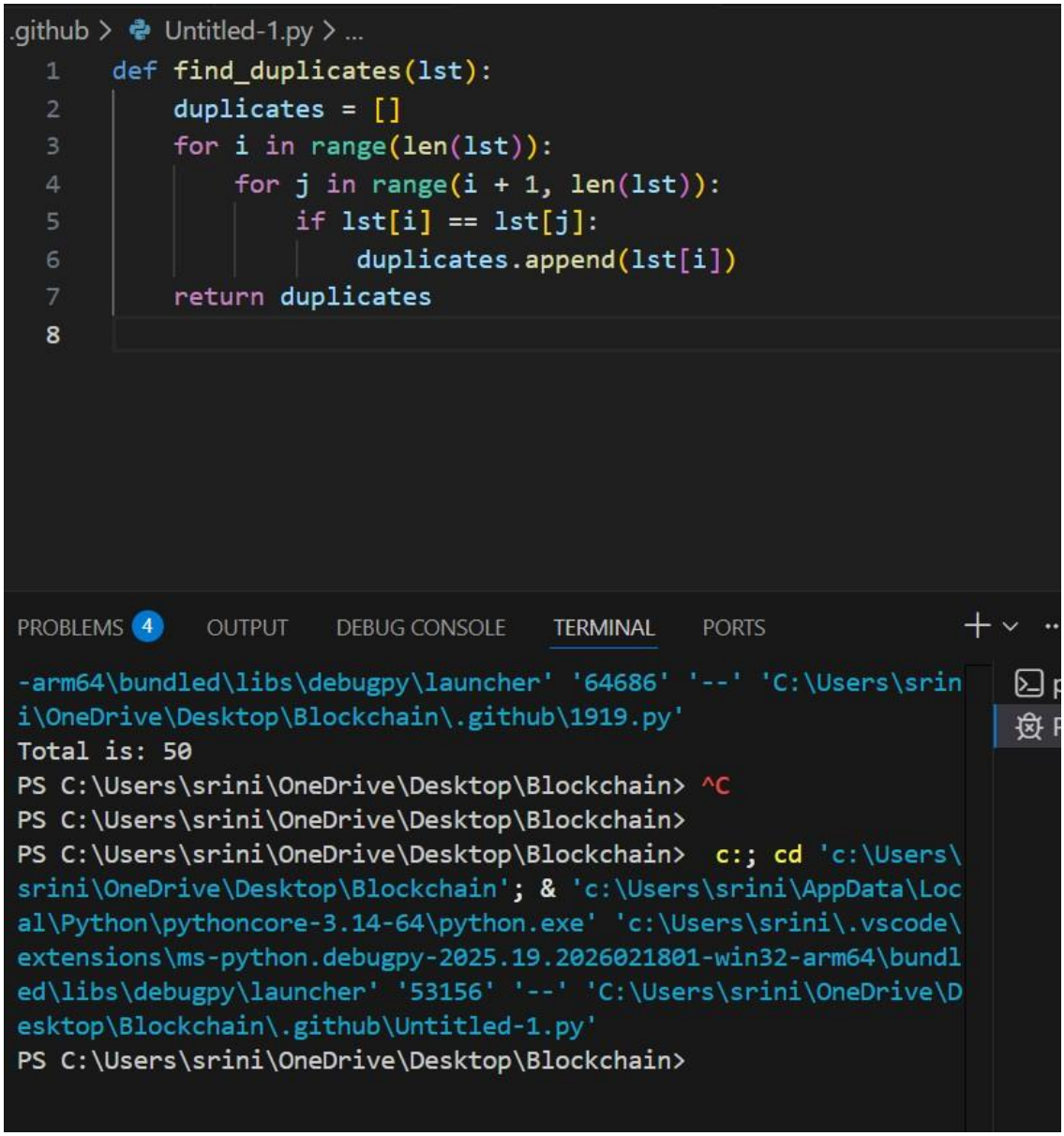
• Optimized duplicate-detection logic (e.g., using sets or hash-based structures)

• Improved time complexity

• AI explanation of performance improvement

• Clean, readable implementation

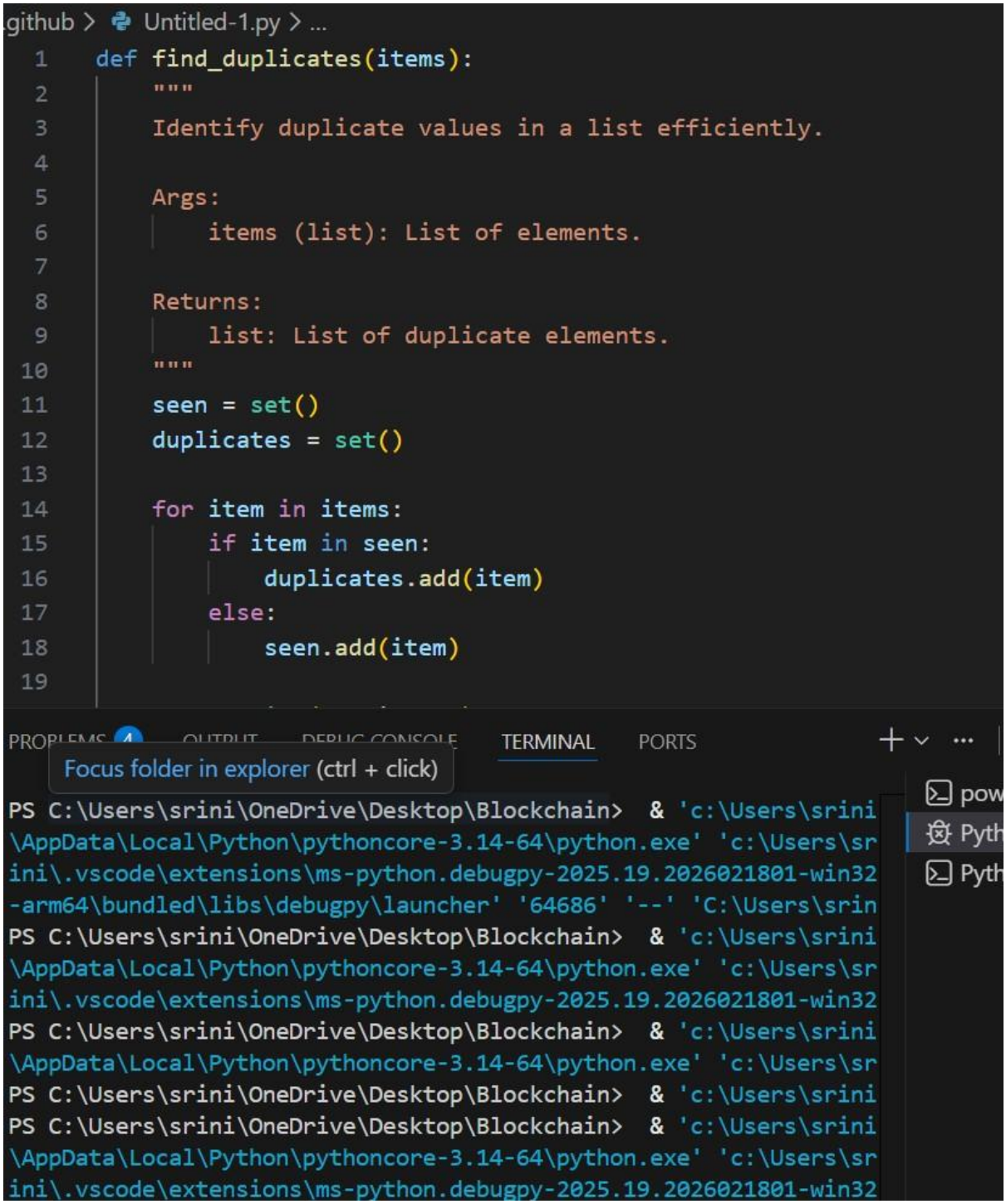
Original Inefficient Code



Performance Issue

- **Uses nested loops → O(n^2) time complexity**
- **Very slow for large datasets**

Optimized Version:

- 

Task 3: Readability and Maintainability Refactoring

Scenario

A working script exists in a project, but it is difficult to understand due to

poor naming, formatting, and structure. The team wants it rewritten for

long-term maintainability.

Task Description

You are given a poorly structured Python function with:

- **Cryptic function names**
- **Poor indentation**
- **Unclear variable naming**
- **No documentation**

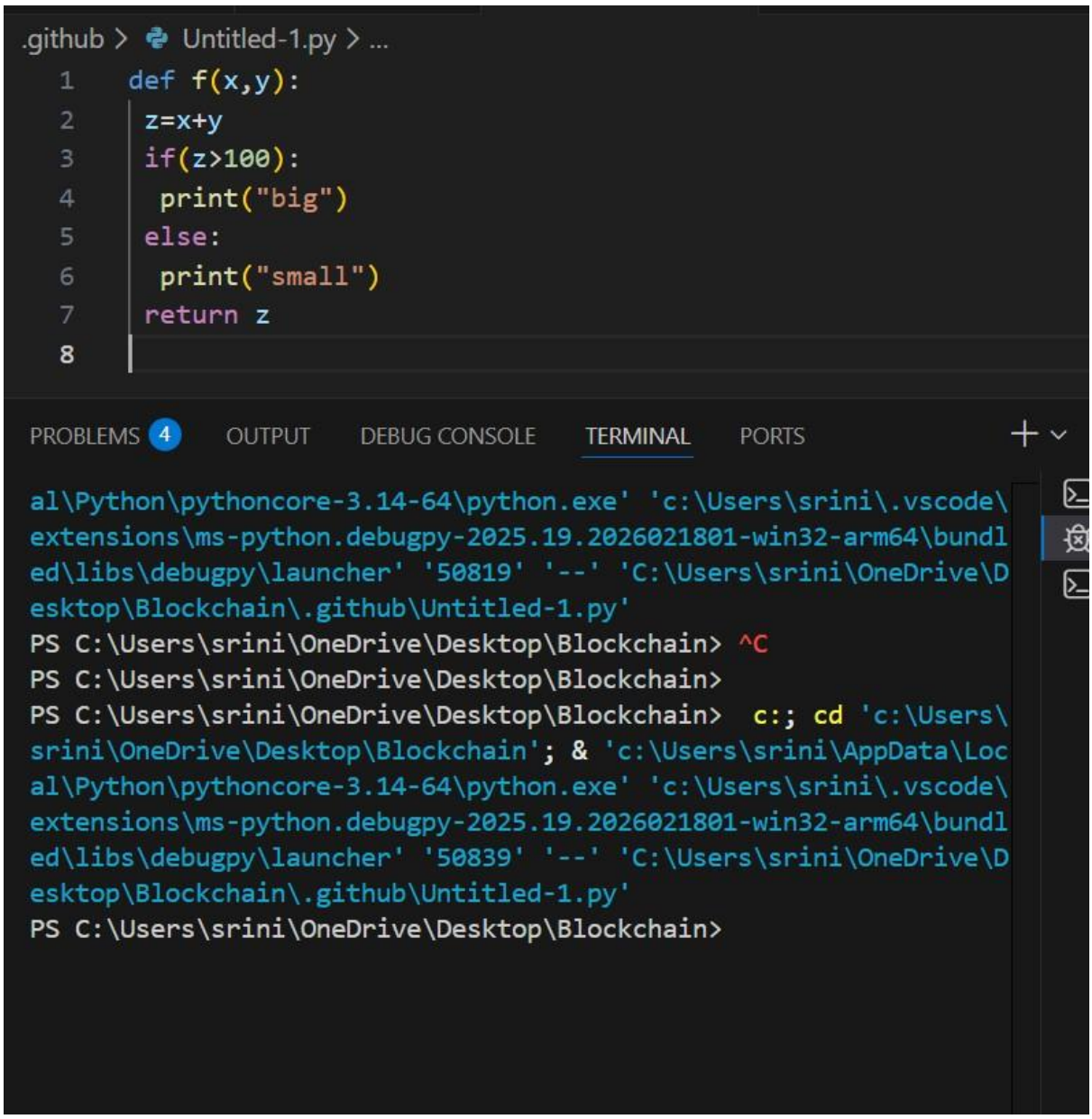
Use AI-assisted review to:

- **Refactor the code for clarity**
- **Apply PEP 8 formatting standards**
- **Improve naming conventions**
- **Add meaningful documentation**

Expected Outcome

- **Clean, well-structured code**
- **Descriptive function and variable names**
- **Proper indentation and formatting**
- **Docstrings explaining the function purpose**
- **AI explanation of readability improvements**

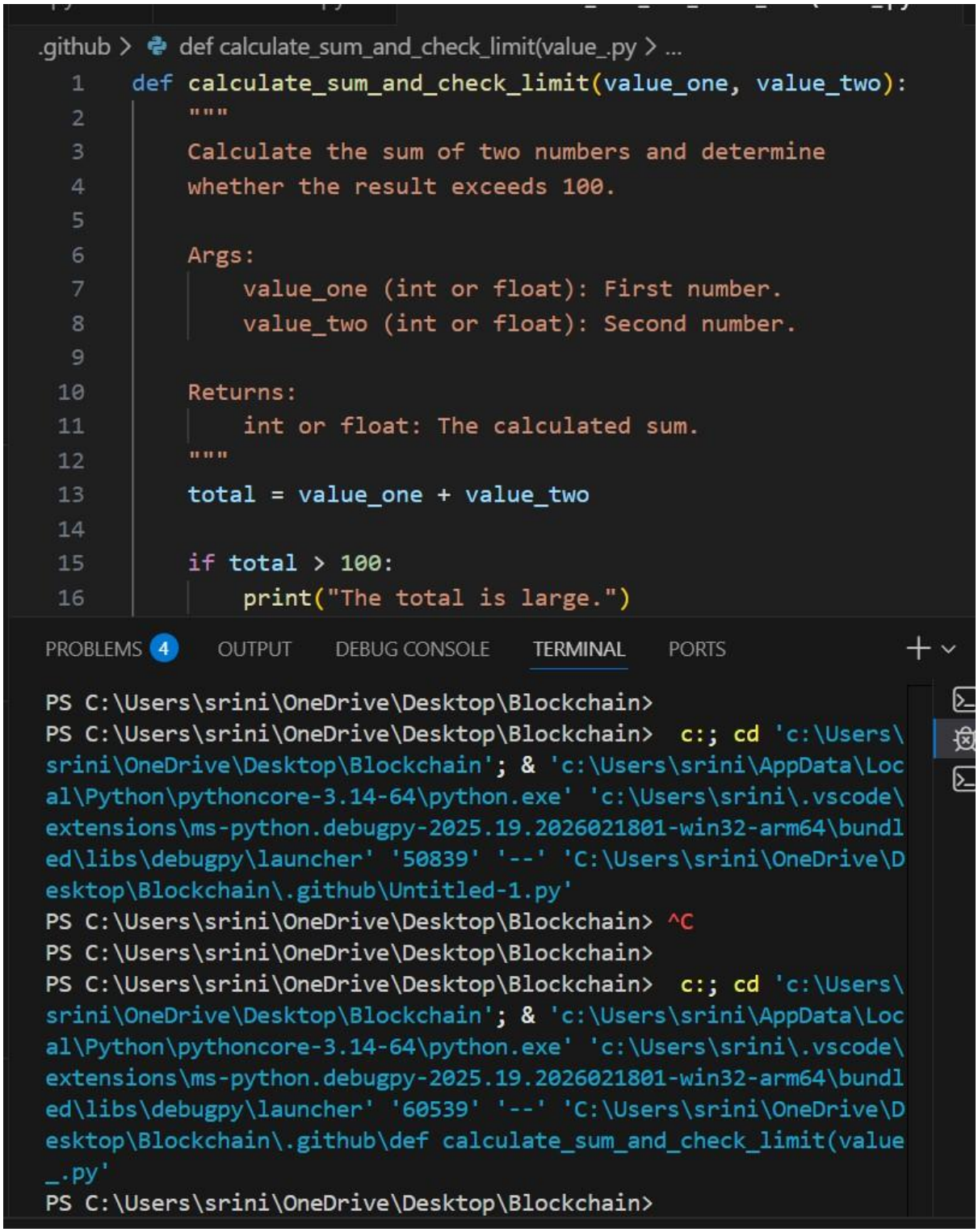
Poorly Written Code:



Issues

- **Cryptic function name**
- **Poor indentation**
- **No documentation**
- **Unclear variable names**

Refactored Version:



Task 4: Secure Coding and Reliability Review

Scenario

A backend function retrieves user data from a database but has security

vulnerabilities and poor error handling, making it unsafe for production

deployment.

Task Description

You are given a Python script that:

- **Uses unsafe SQL query construction**
- **Has no input validation**
- **Lacks exception handling**

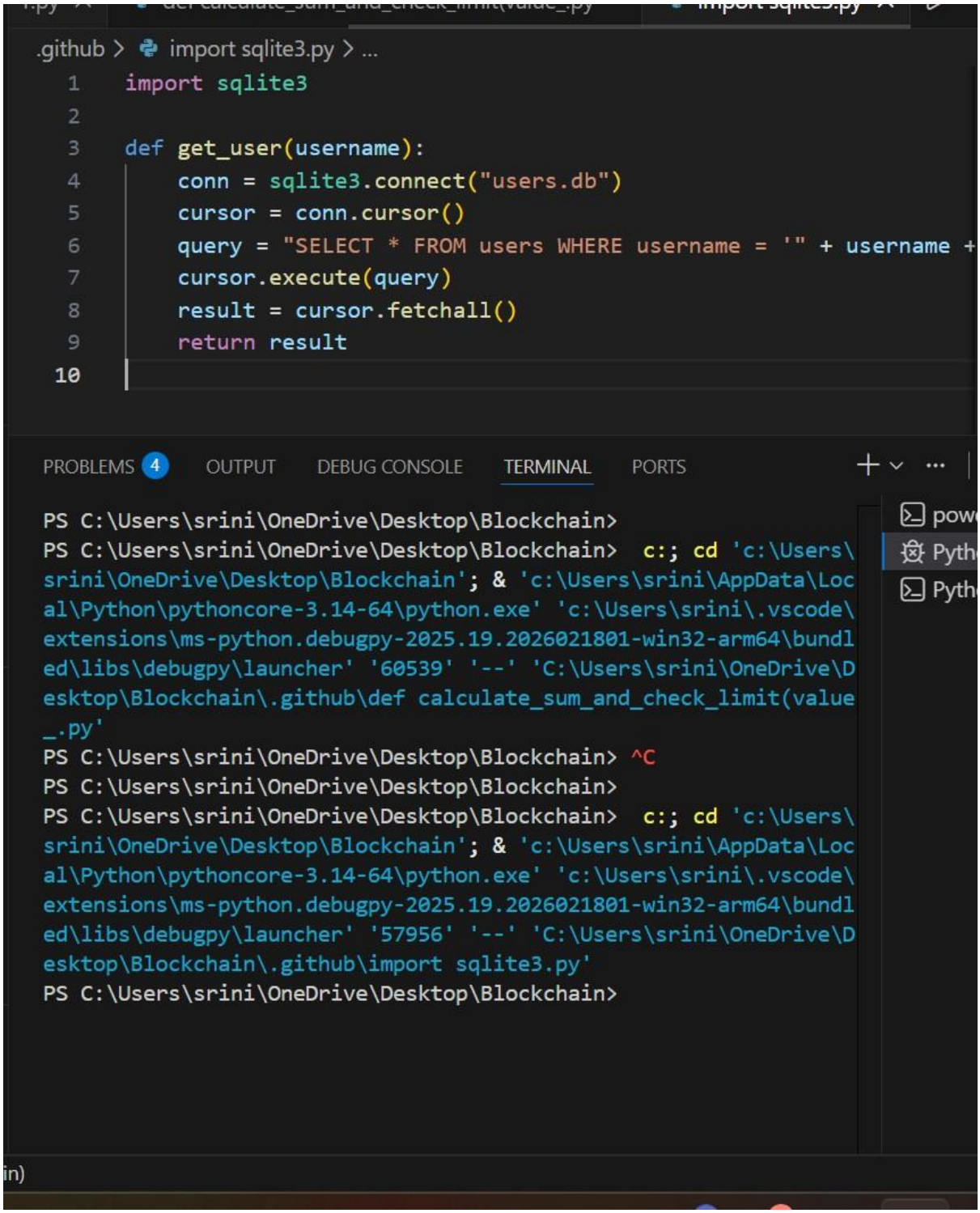
Use AI tools to:

- **Identify security vulnerabilities**
- **Refactor the code using safe coding practices**
- **Add proper exception handling**
- **Improve robustness and reliability**

Expected Outcome

- **Secure SQL queries using parameterized statements**
- **Input validation logic**
- **Try-except blocks for runtime safety**
- **AI-generated explanation of security improvements**
- **Production-ready code structure**

Unsafe Code

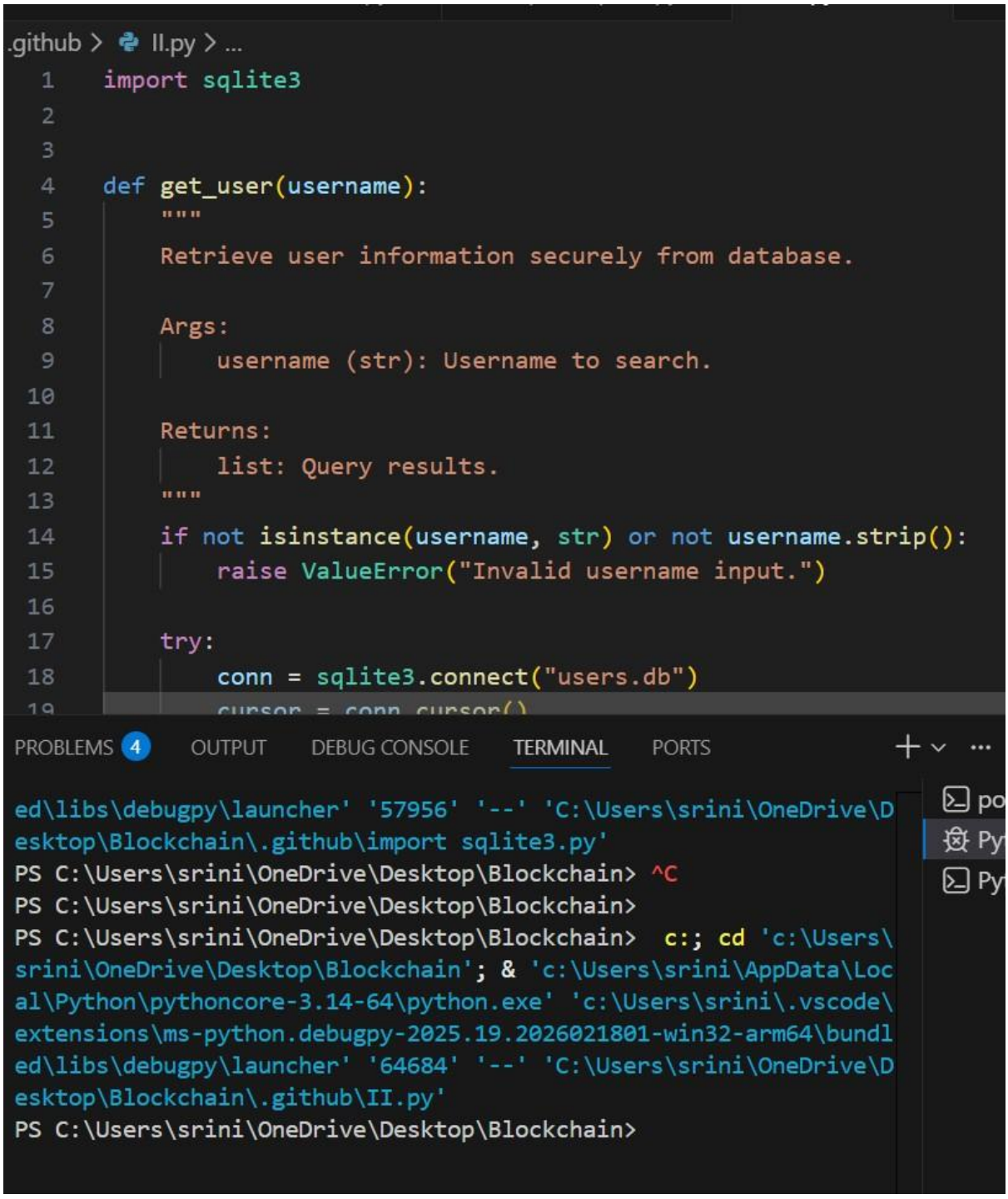


Security Issues

- **SQL Injection vulnerability**

- **No input validation**
- **No exception handling**
- **Connection not properly closed**

Secure Version



Task 5: AI-Based Automated Code Review Report

Scenario

Your team uses AI tools to perform automated preliminary code reviews

before human review, to improve code quality and consistency across

projects.

Task Description

You are provided with a poorly written Python script.

Using AI-assisted review:

• Generate a structured code review report that evaluates:

o Code readability

o Naming conventions

o Formatting and style consistency

o Error handling

o Documentation quality

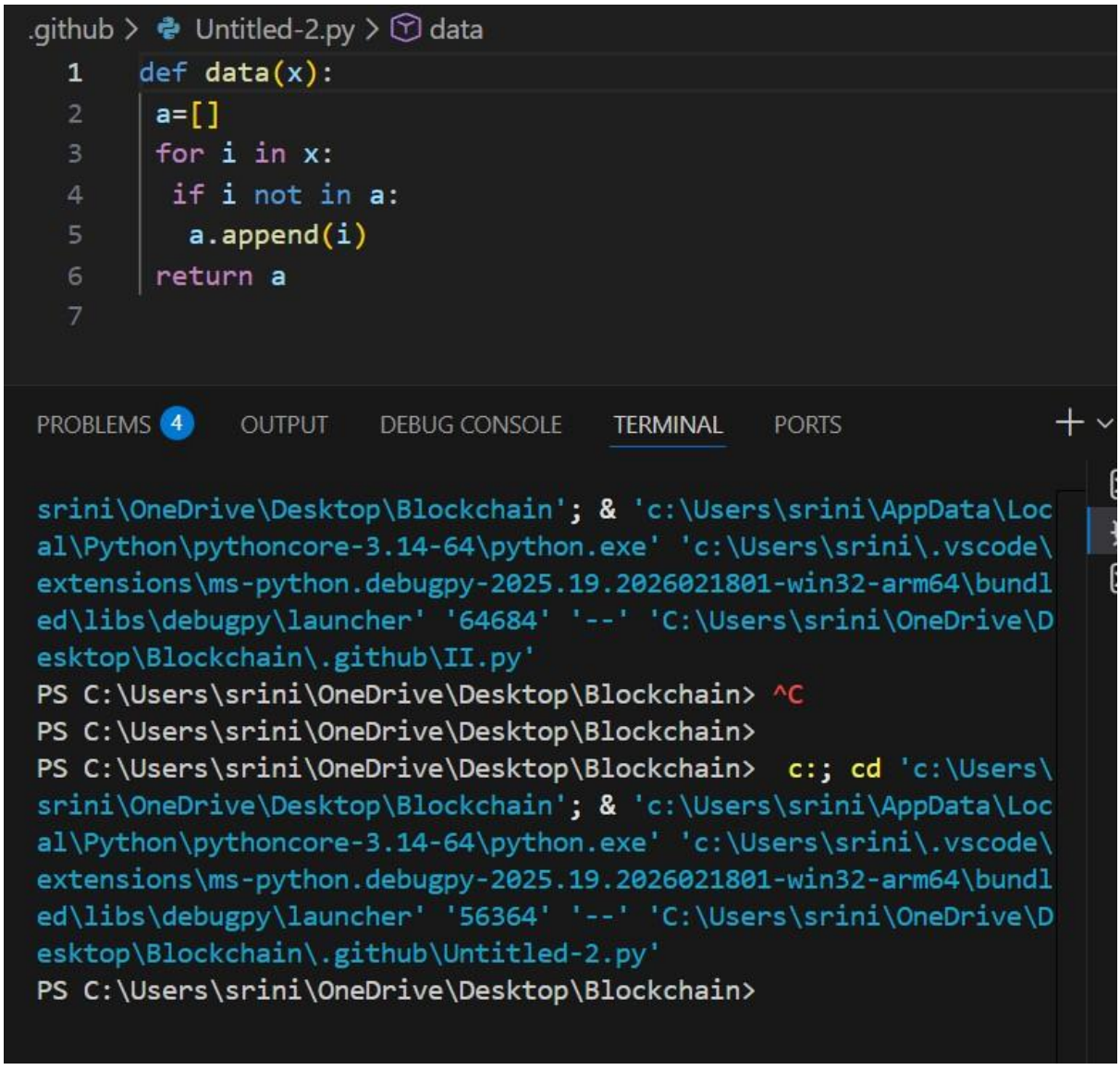
o Maintainability

The task is not just to fix the code, but to analyze and report on quality

issues.

Expected Outcome

- **AI-generated review report including:**
  - **Identified quality issues**
  - **Risk areas**
  - **Code smell detection**
  - **Improvement suggestions**
- **Optional improved version of the code**



AI Code Review Report

1️⃣ Readability

❌ Function name data is vague
❌ Variable a is meaningless
❌ No documentation

2️⃣ Naming Conventions

❌ Does not follow descriptive naming
❌ Violates clarity standards

3️⃣ Formatting & Style

❌ Poor indentation
❌ No spacing
❌ No docstring

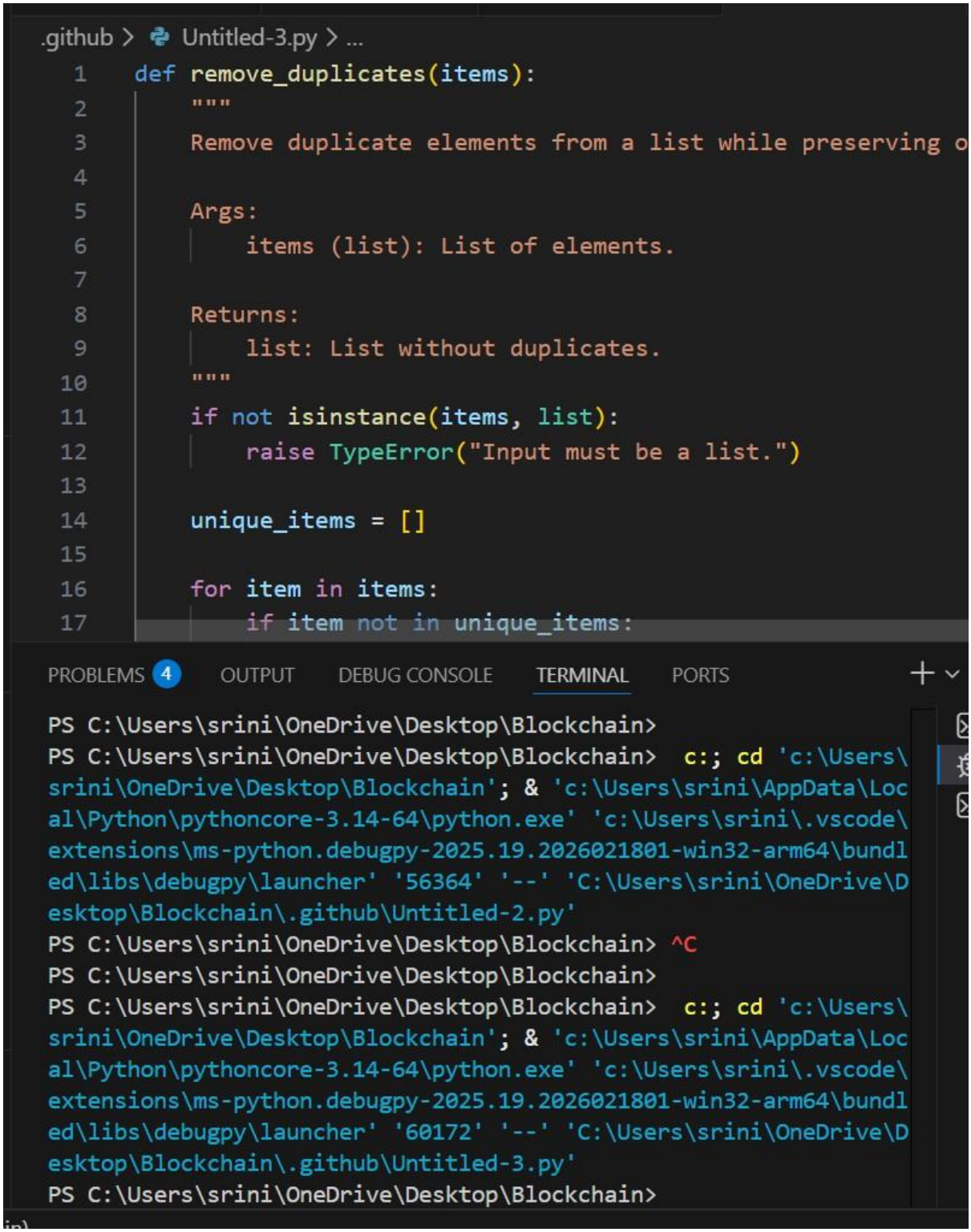
4️⃣ Error Handling

❌ No input validation
❌ Assumes iterable input

5️⃣ Maintainability

⚠ Difficult to understand intent
⚠ No explanation of logic

Improved Version:


.github > Untitled-3.py > ...
def remove_duplicates(items):
"""
Remove duplicate elements from a list while preserving o
Args:
items (list): List of elements.
Returns:
list: List without duplicates.
"""
if not isinstance(items, list):
raise TypeError("Input must be a list.")
unique_items = []
for item in items:
if item not in unique_items:
PROBLEMS 4
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '56364' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\Untitled-2.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain> c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '60172' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\Untitled-3.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>