*Assignment*

**Htno : 2503A52L09**

**Lab 7:** Error Debugging with AI – Systematic Approaches to Finding and Fixing Bugs

**Lab Objectives:**

* To identify and correct syntax, logic, and runtime errors in Python programs using AI tools.
* To understand common programming bugs and AI-assisted debugging suggestions.
* To evaluate how AI explains, detects, and fixes different types of coding errors.
* To build confidence in using AI to perform structured debugging practices.

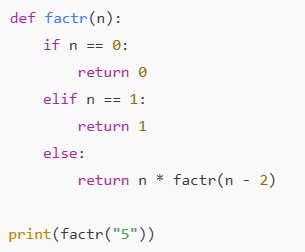
**Lab Outcomes (LOs):**

After completing this lab, students will be able to:

* Use AI tools to detect and correct syntax, logic, and runtime errors.
* Interpret AI-suggested bug fixes and explanations.
* Apply systematic debugging strategies supported by AI-generated insights.
* Refactor buggy code using responsible and reliable programming patterns.

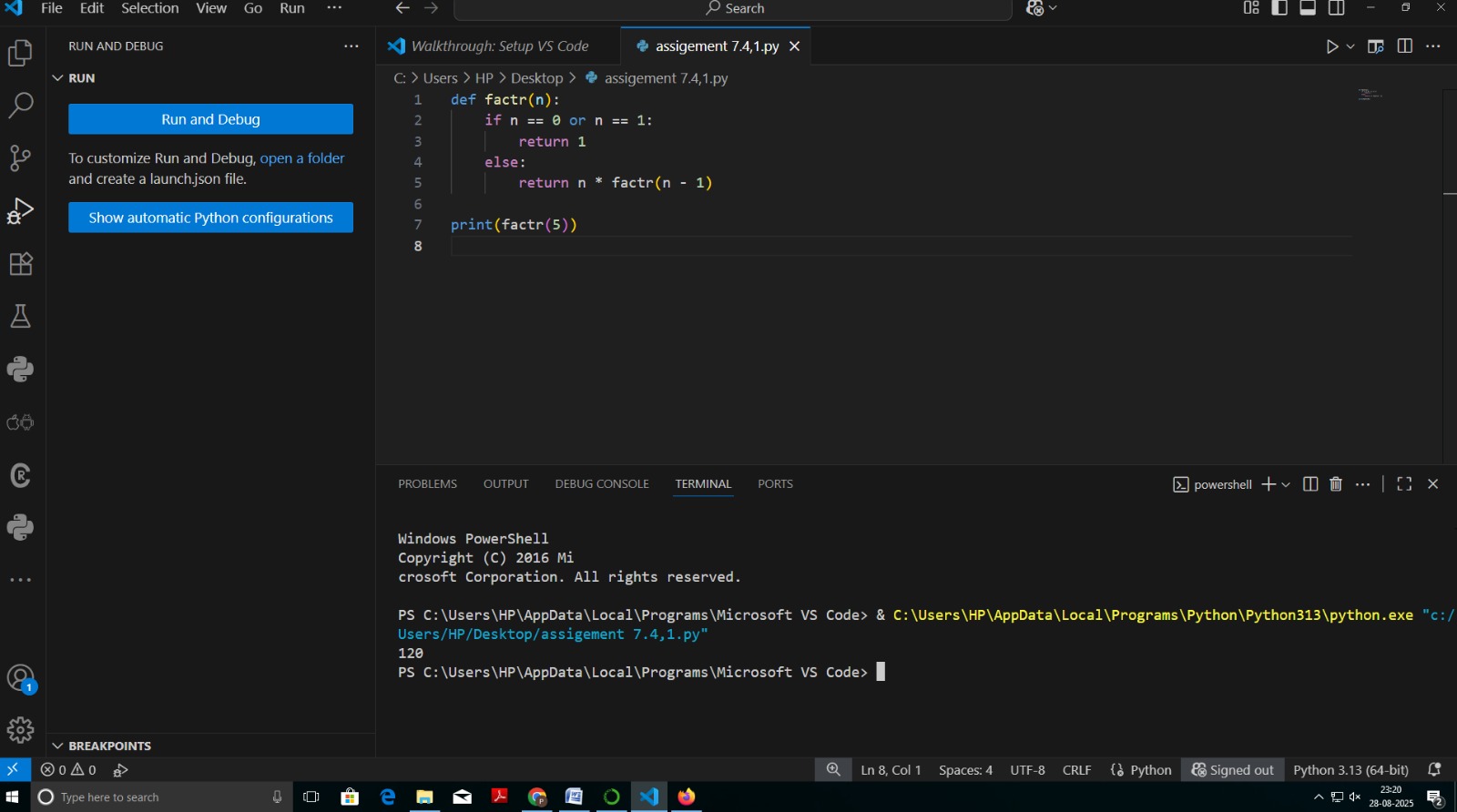
1. **Task Description #1:**

Introduce a buggy Python function that calculates the factorial of a number using recursion. Use Copilot or Cursor AI to detect and fix the logical or syntax errors.



**Expected Outcome #1:**

* Copilot or Cursor AI correctly identifies missing base condition or incorrect recursive call and suggests a functional factorial implementation.



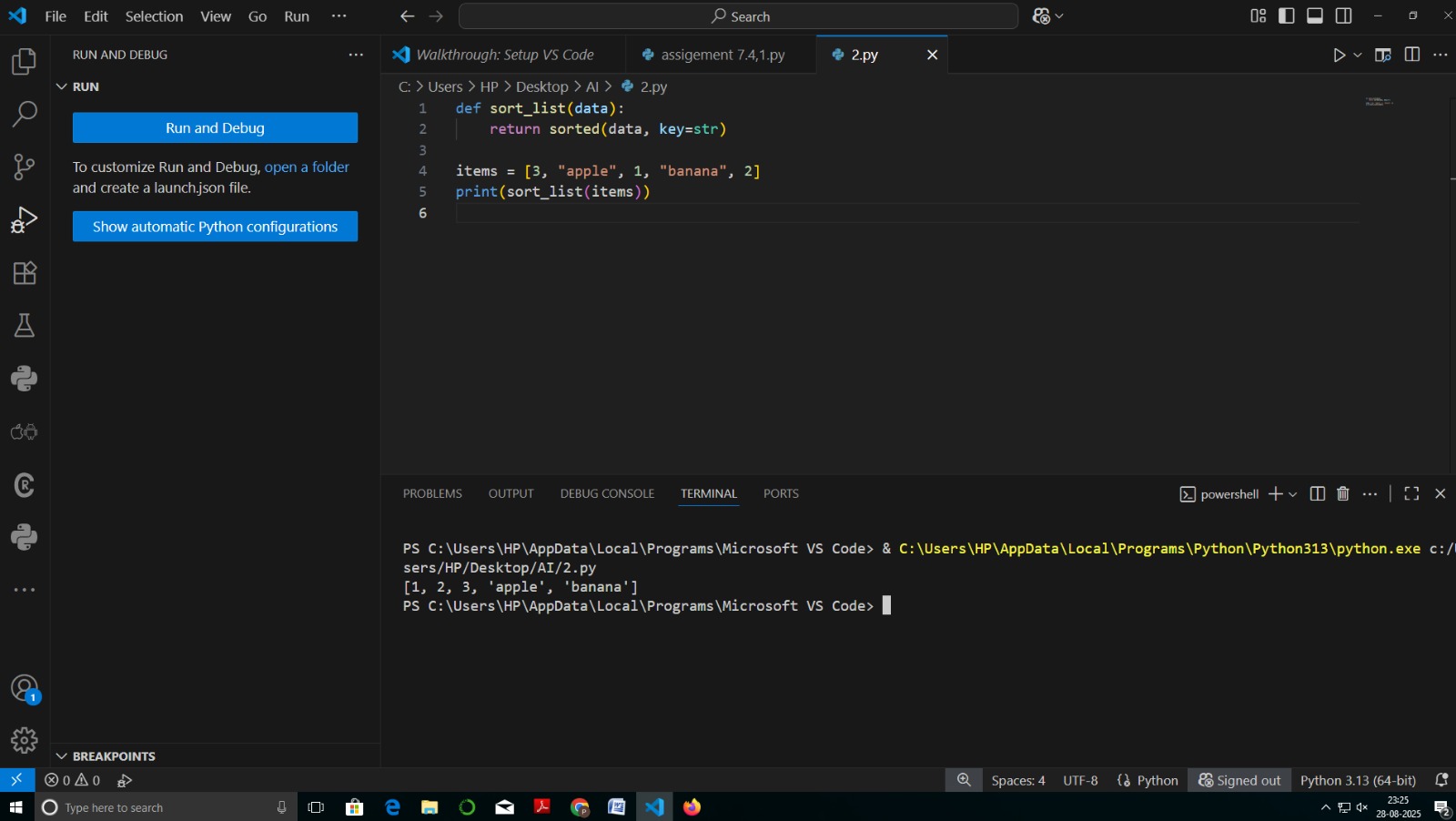
**Task Description #2:**

* Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed integers and strings). Prompt AI to detect the issue and fix the code for consistent sorting**.**

****

**Expected Outcome #2:**

* AI detects the type inconsistency and either filters or converts list elements, ensuring successful sorting without a crash.



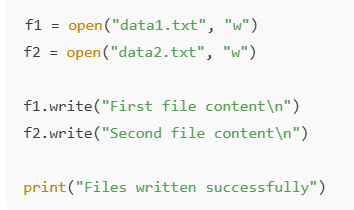
**Task Description #3:**

* Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block).

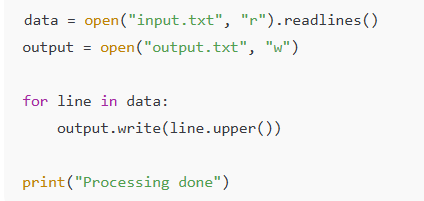
**Code1**

****

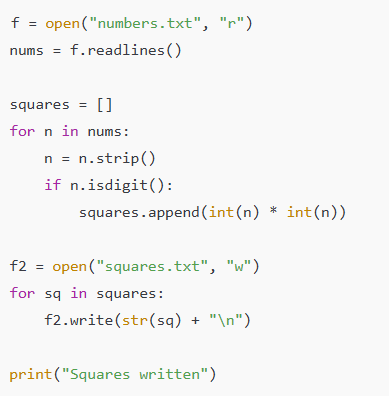
**Code2**

****

**Code3**

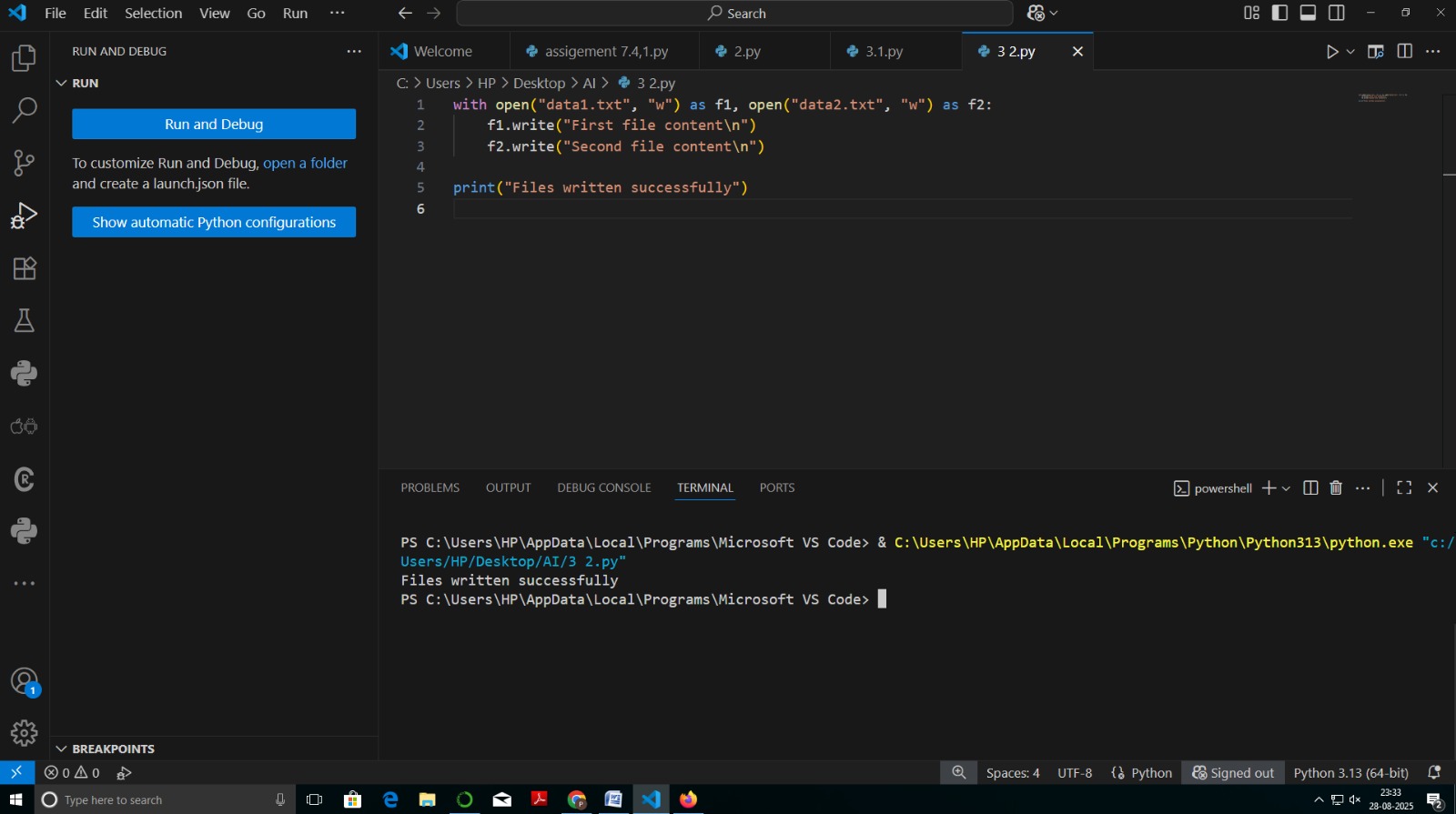
****

**Code4:**

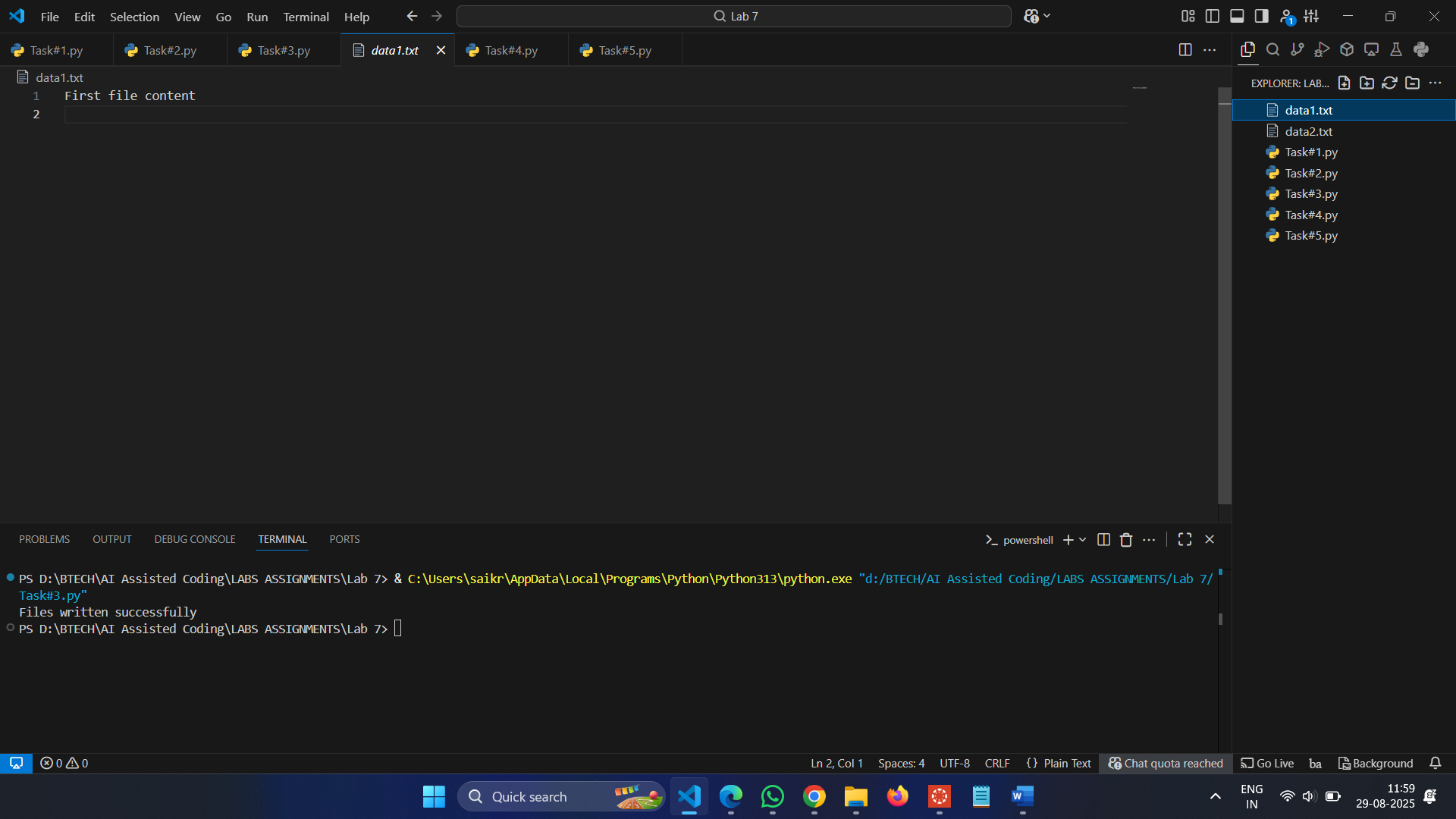
****

**Expected Outcome #3:**

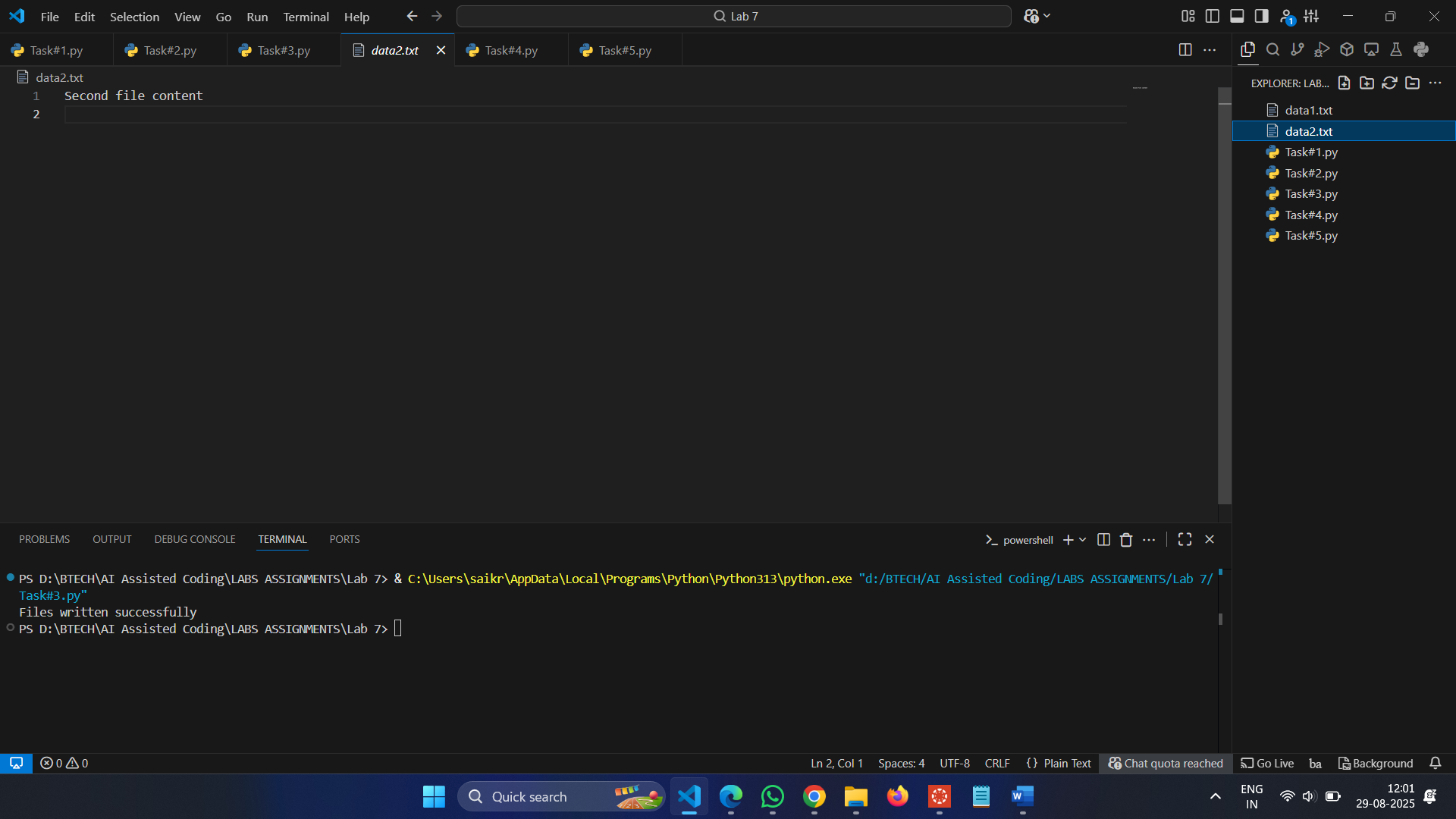
* AI refactors the code to use a context manager, preventing resource leakage and runtime warnings.



* **File one :**

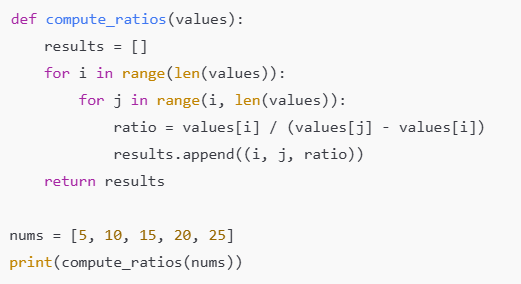


* **File Two :**



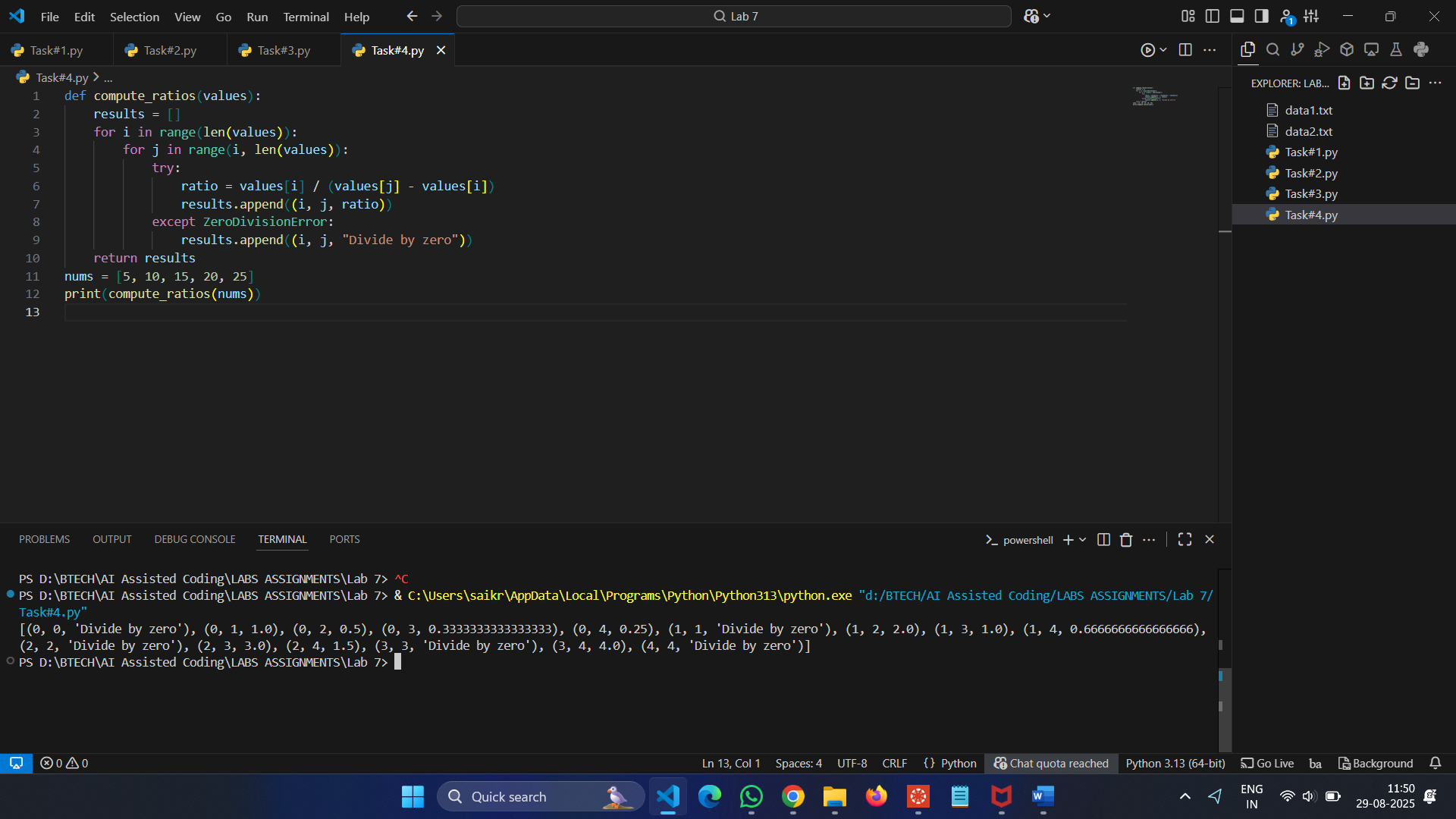
**Task Description #4:**

* Provide a piece of code with a ZeroDivisionError inside a loop. Ask AI to add error handling using try-except and continue execution safely.



**Expected Outcome #4:**

* Copilot adds a try-except block around the risky operation, preventing crashes and printing a meaningful error message.



**Task Description #5:**

* Include a buggy class definition with incorrect \_\_init\_\_ parameters or attribute references. Ask AI to analyze and correct the constructor and attribute usage.

**class StudentRecord:**

**def \_\_init\_\_(self, name, id, courses=[]):**

**self.studentName = names**

**self.student\_id = id**

**self.courses = courseList**

**def add\_course(self, course):**

**self.courses.append(course)**

**def get\_summary(self):**

**return f"Student: {self.studentName}, ID: {self.student\_id}, Courses: {', '.join(self.courses)}"**

**class Department:**

**def \_\_init\_\_(self, deptName, students=None):**

**self.dept\_name = deptName**

**self.students = students**

**def enroll\_student(self, student):**

**self.students.append(student)**

**def department\_summary(self):**

**return f"Department: {self.dept\_name}, Total Students: {len(self.student)}**

**s1 = StudentRecord("Alice", 101, ["Math", "Science"])**

**d1 = Department("Computer Science")**

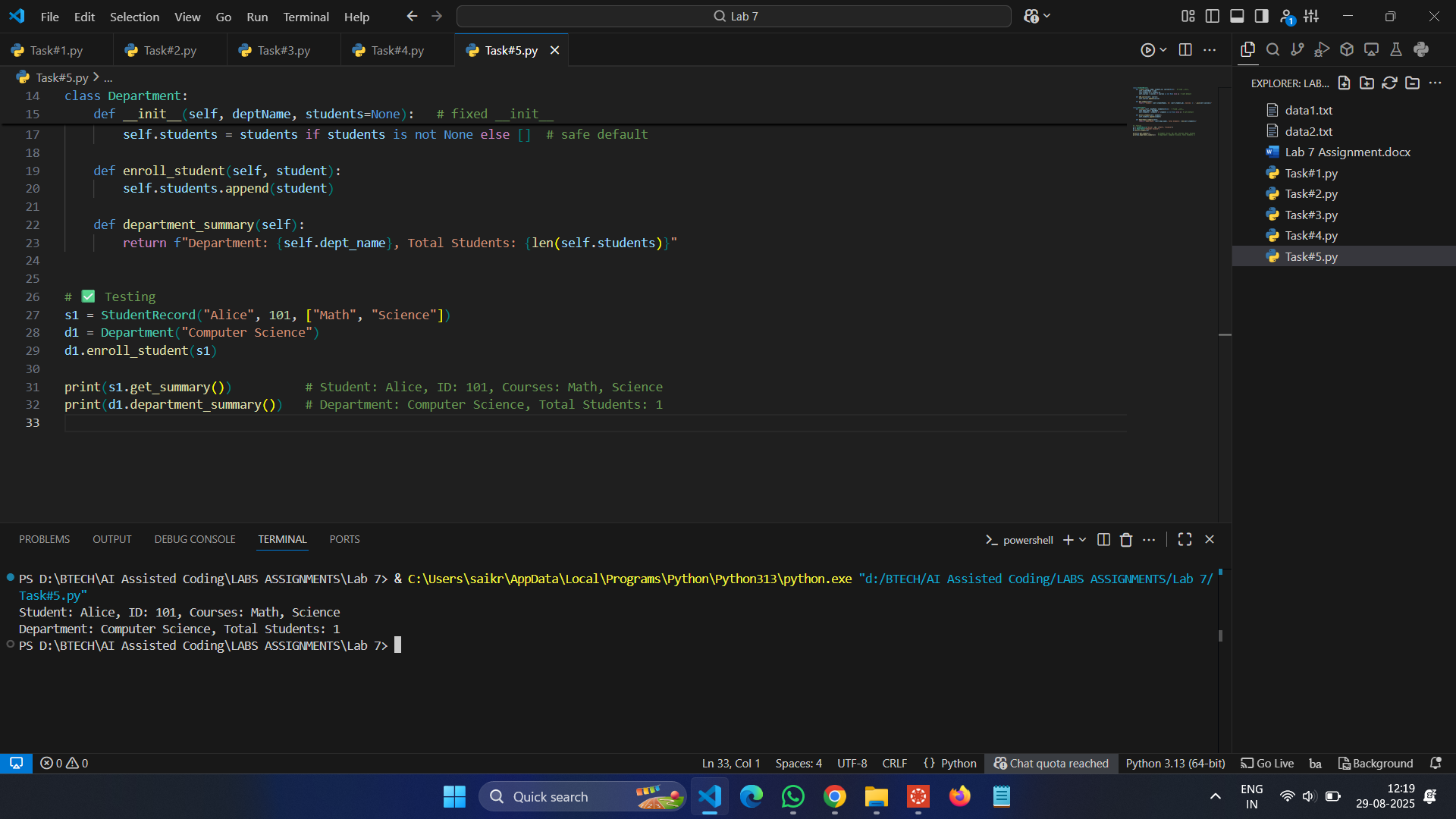
**d1.enroll\_student(s1)**

**print(s1.get\_summary())**

**print(d1.department\_summary())**

**Expected Outcome #5:**

* Copilot identifies mismatched parameters or missing self references and rewrites the class with accurate initialization and usage.



* **Observation**

*Overall, the assignment reinforces core programming principles: Scorrectness in algorithms, strict data handling, managing system resources carefully, robust error handling, safe object-oriented design, and the benefits of AI-supported debugging.*

*This comprehensive approach prepares developers to write reliable, maintainable, and professional Python code.*