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

Assignment Number: 7.4

Enrollment number: 2503A51L36

Name:Suhana Rehan

**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.

**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.

**Prompt:** Detect and fix logical or syntax error in the given code

**Code:**

def factorial(n):

    if n == 0 or n == 1:

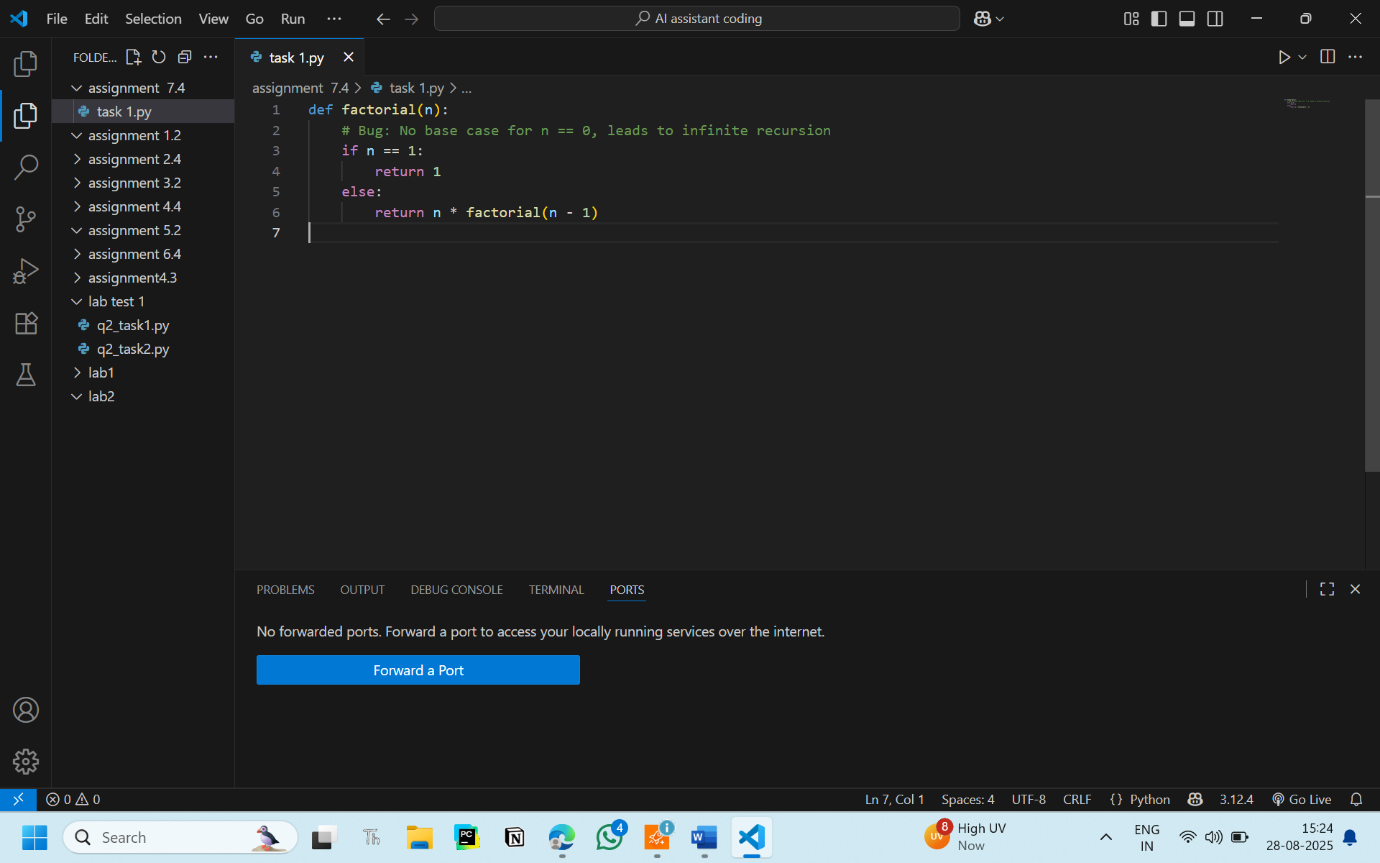
        return 1

    else:

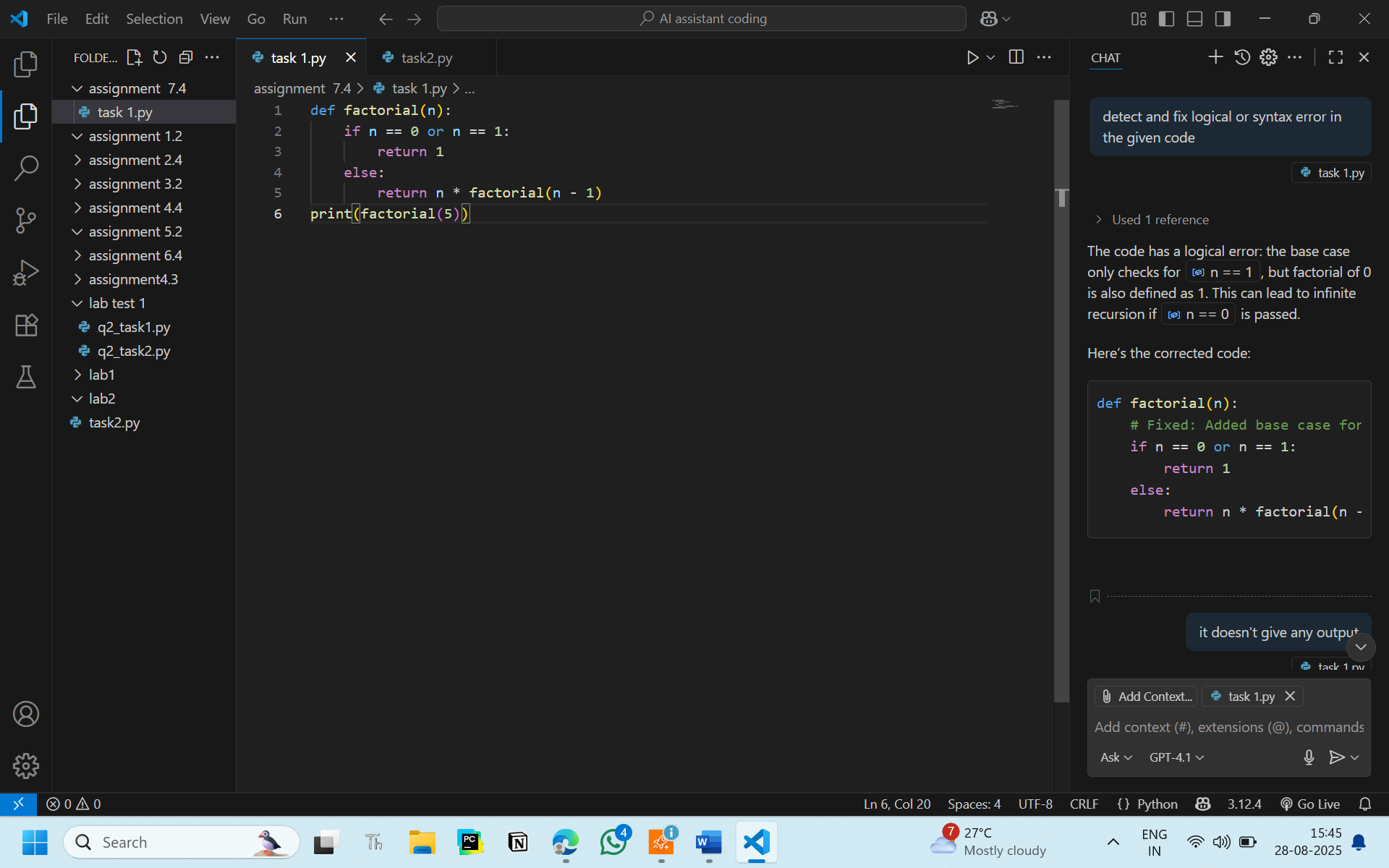
        return n \* factorial(n - 1)

print(factorial(5))

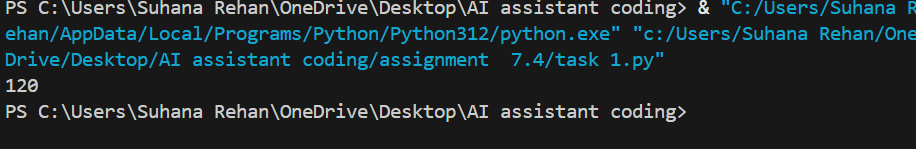
**Code With Bugs:**



**Code Without Bugs:**



**OUTPUT:**



120

**Observation:**

I observed that the AI correct the bugs and syntax error in the given code

**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**.**

**Prompt:** Fix and debug the given code

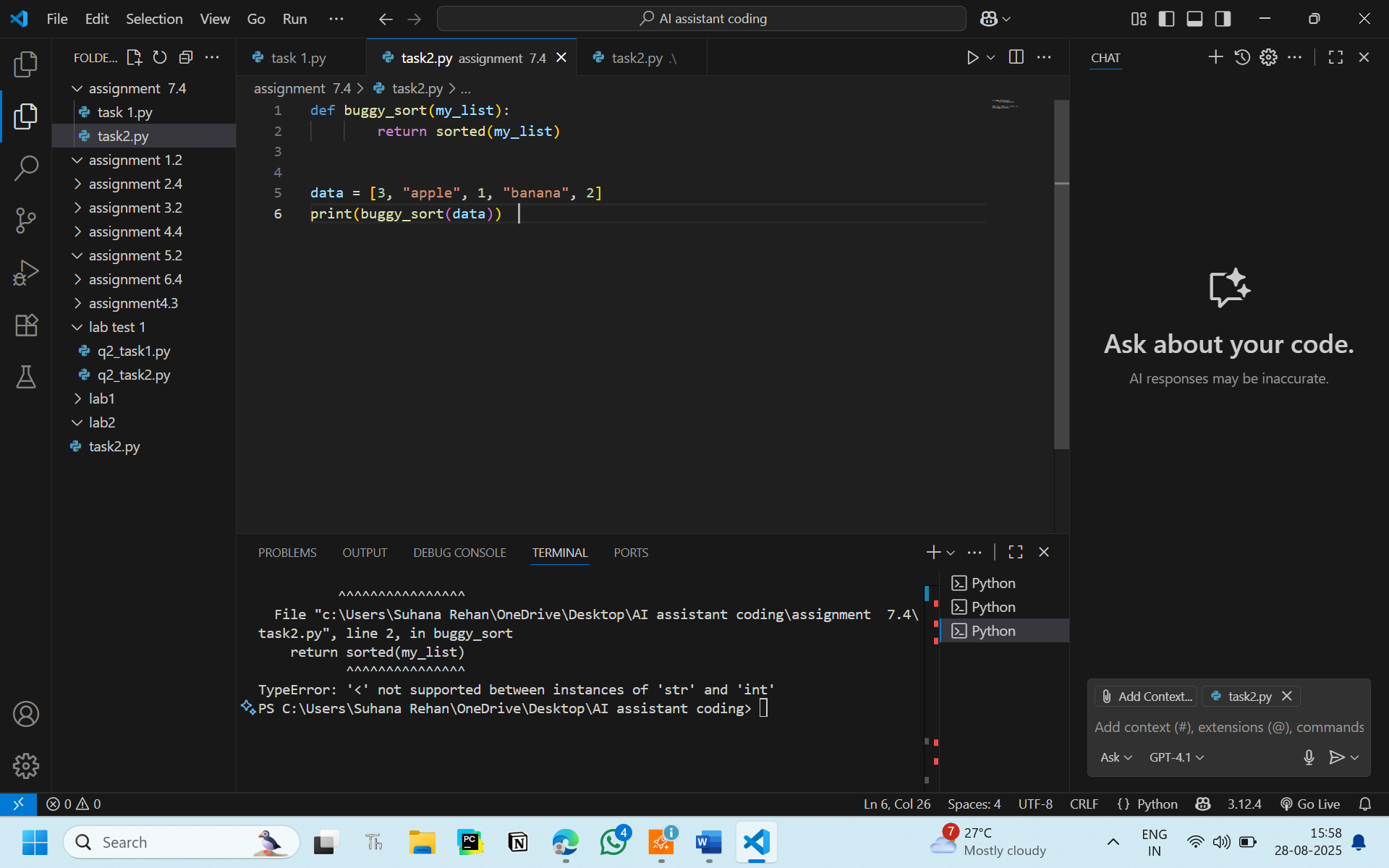
**Code with bug:**

def buggy\_sort(my\_list):

        return sorted(my\_list)

data = [3, "apple", 1, "banana", 2]

print(buggy\_sort(data))



**Code without bugs (After AI removed the bug):**

def buggy\_sort(my\_list):

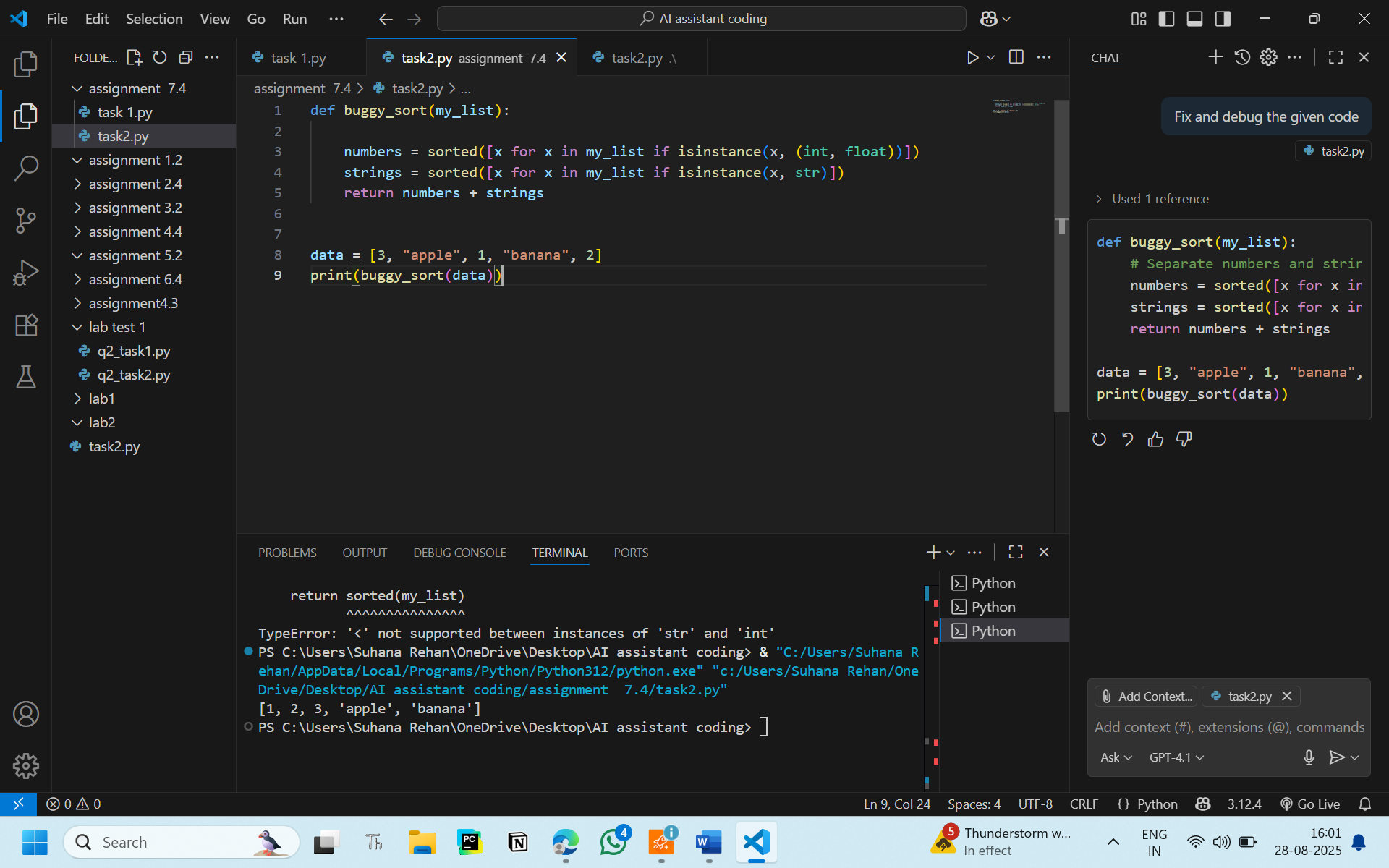
    numbers = sorted([x for x in my\_list if isinstance(x, (int, float))])

    strings = sorted([x for x in my\_list if isinstance(x, str)])

    return numbers + strings

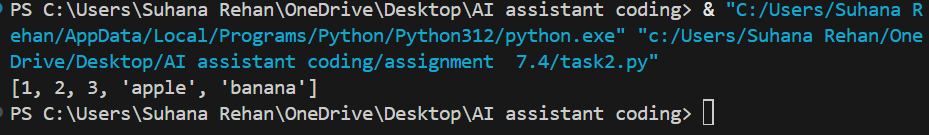
data = [3, "apple", 1, "banana", 2]

print(buggy\_sort(data))



**Output:**

[1, 2, 3, 'apple', 'banana']

****

**Observation:**

The AI removed the bug and sytax error for the given program, efficiently.

**Task 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).

**Prompt:** Fix and debug the given code

**Code with bug:**

def read\_file\_buggy(filename):

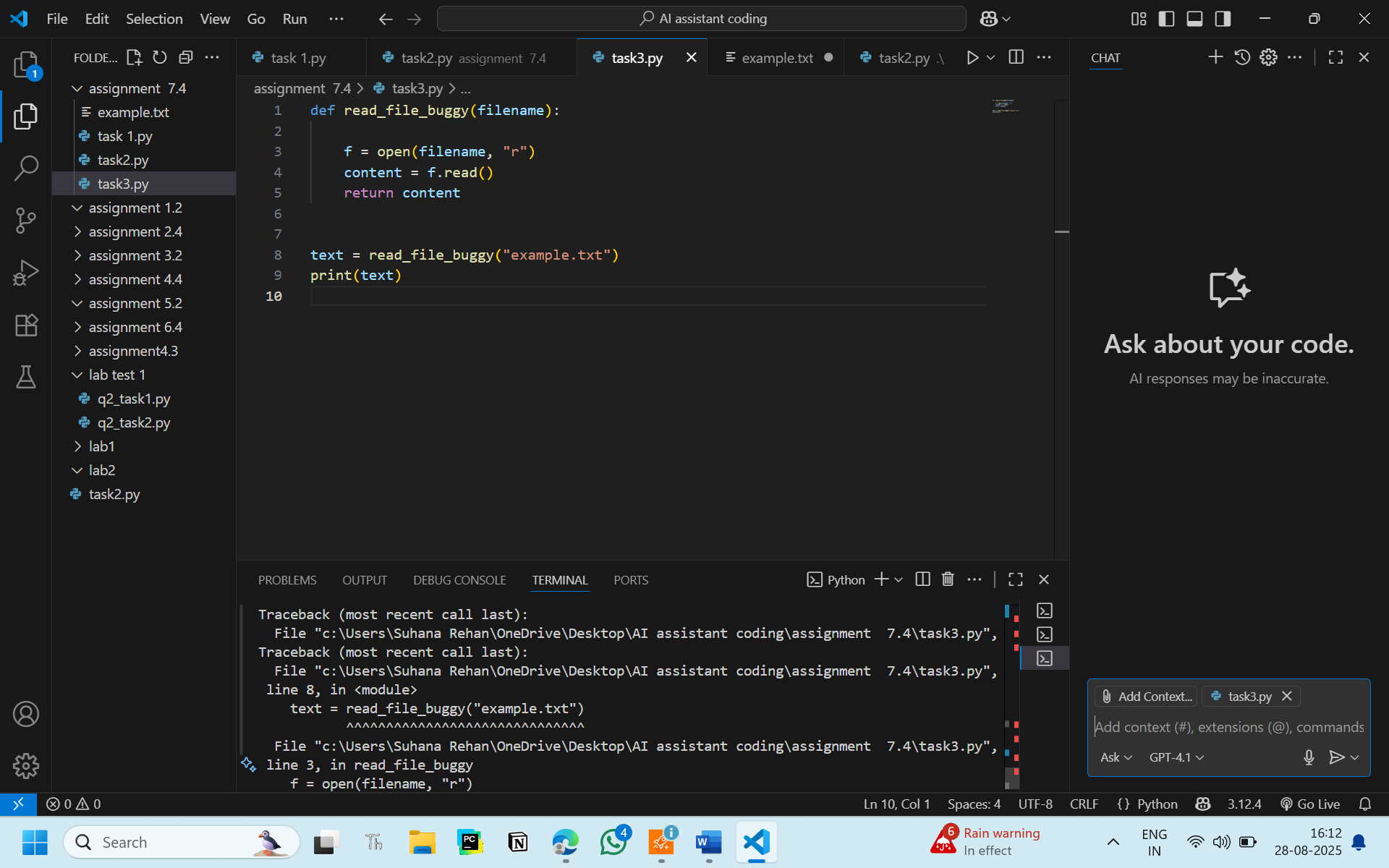
    f = open(filename, "r")

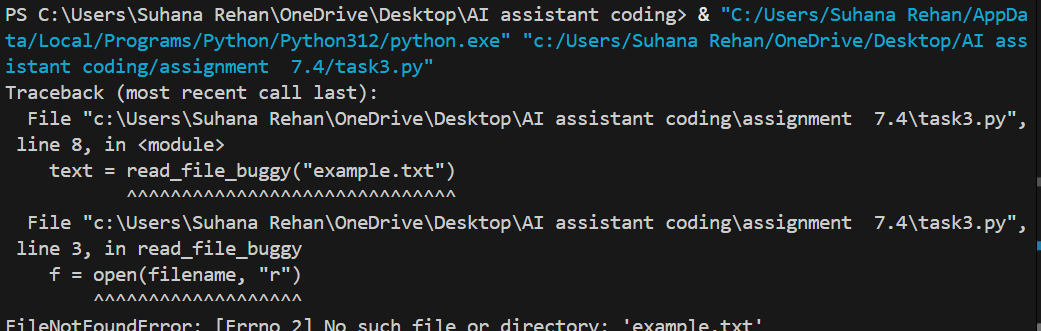
    content = f.read()

    return content

text = read\_file\_buggy("example.txt")

print(text)





**Code without bug:**

def read\_file\_buggy(filename):

    try:

        with open(filename, "r") as f:

            content = f.read()

        print(f"DEBUG: Read content: {content!r}")  # Debug line

        return content

    except FileNotFoundError:

        print(f"File '{filename}' not found.")

        return None

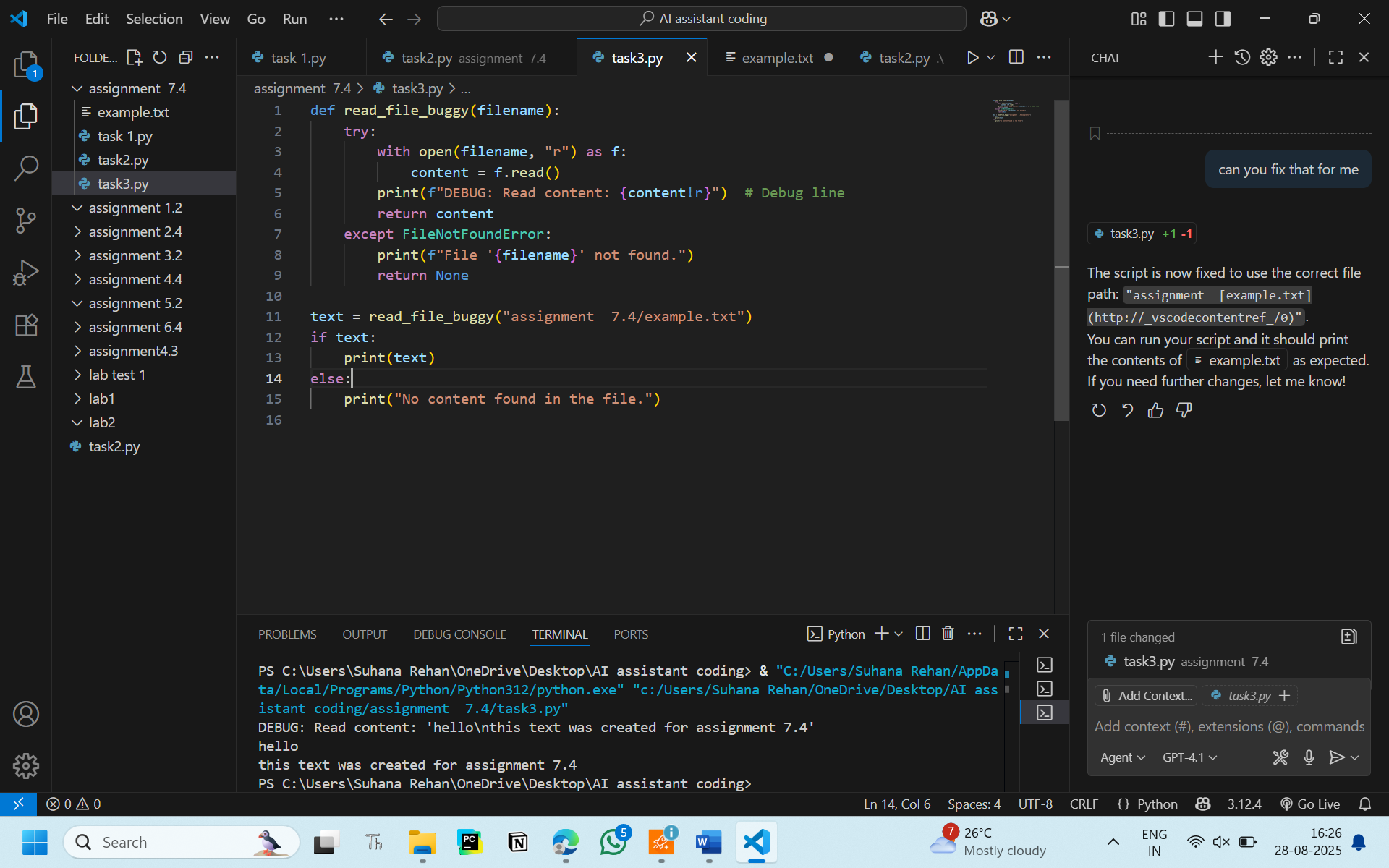
text = read\_file\_buggy("assignment  7.4/example.txt")

if text:

    print(text)

else:

    print("No content found in the file.")

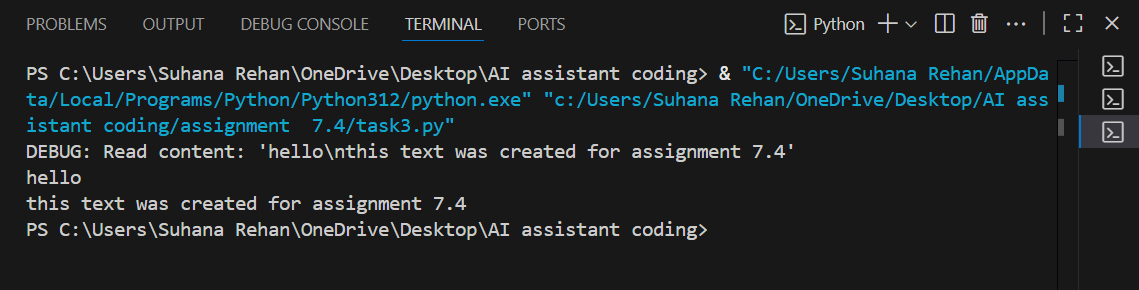


**Output without bug:**

DEBUG: Read content: 'hello\nthis text was created for assignment 7.4'

hello

this text was created for assignment 7.4

****

**Observation:**

Here AI found the file was not close , so it closed the file for me and solved the error for me

**Task 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.

**Prompt:**

Add error handling using try-except and continue execution safely.

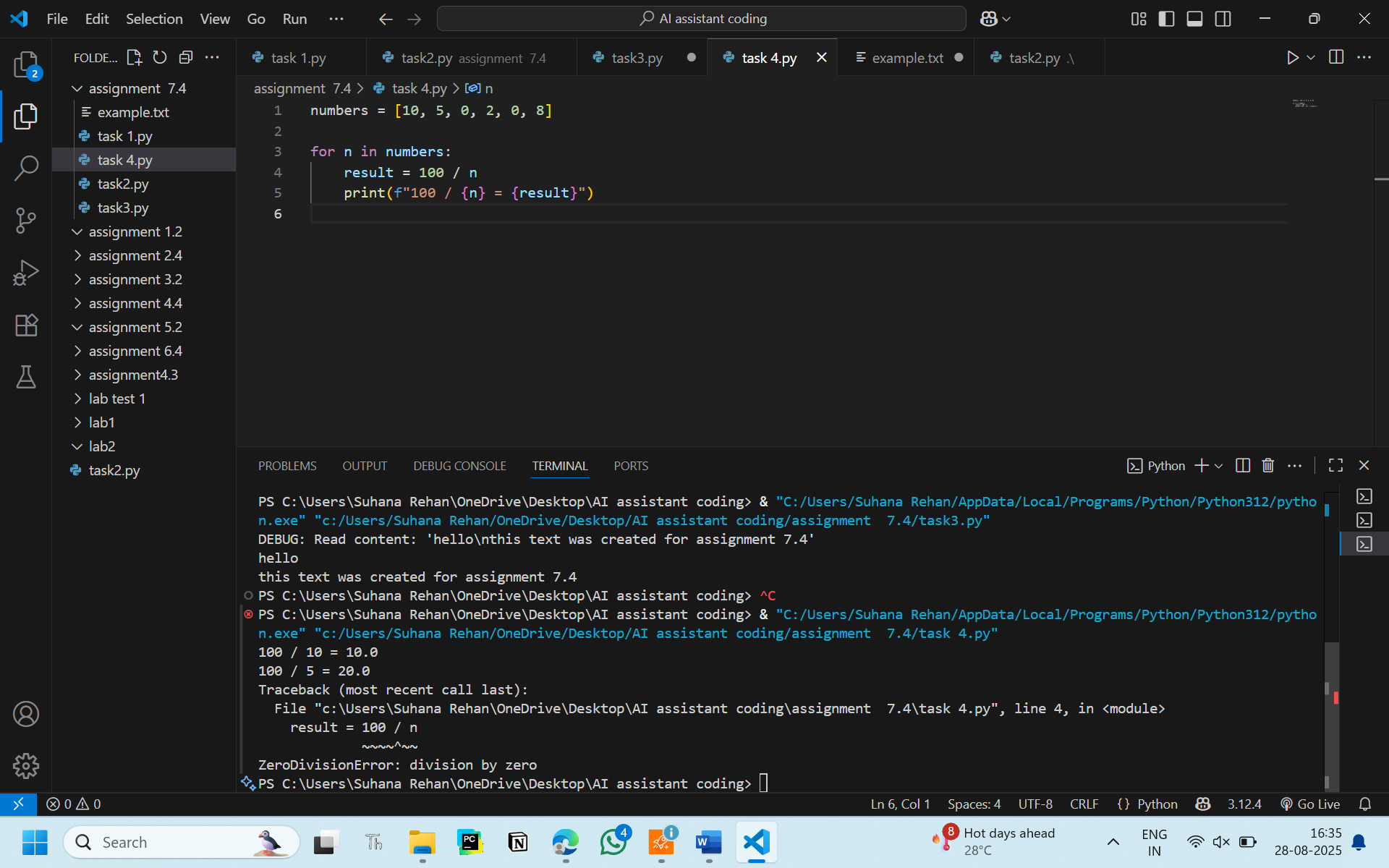
**Code with bug:**

numbers = [10, 5, 0, 2, 0, 8]

for n in numbers:

    result = 100 / n

    print(f"100 / {n} = {result}")





**Code without bug:**

numbers = [10, 5, 0, 2, 0, 8]

for n in numbers:

    try:

        result = 100 / n

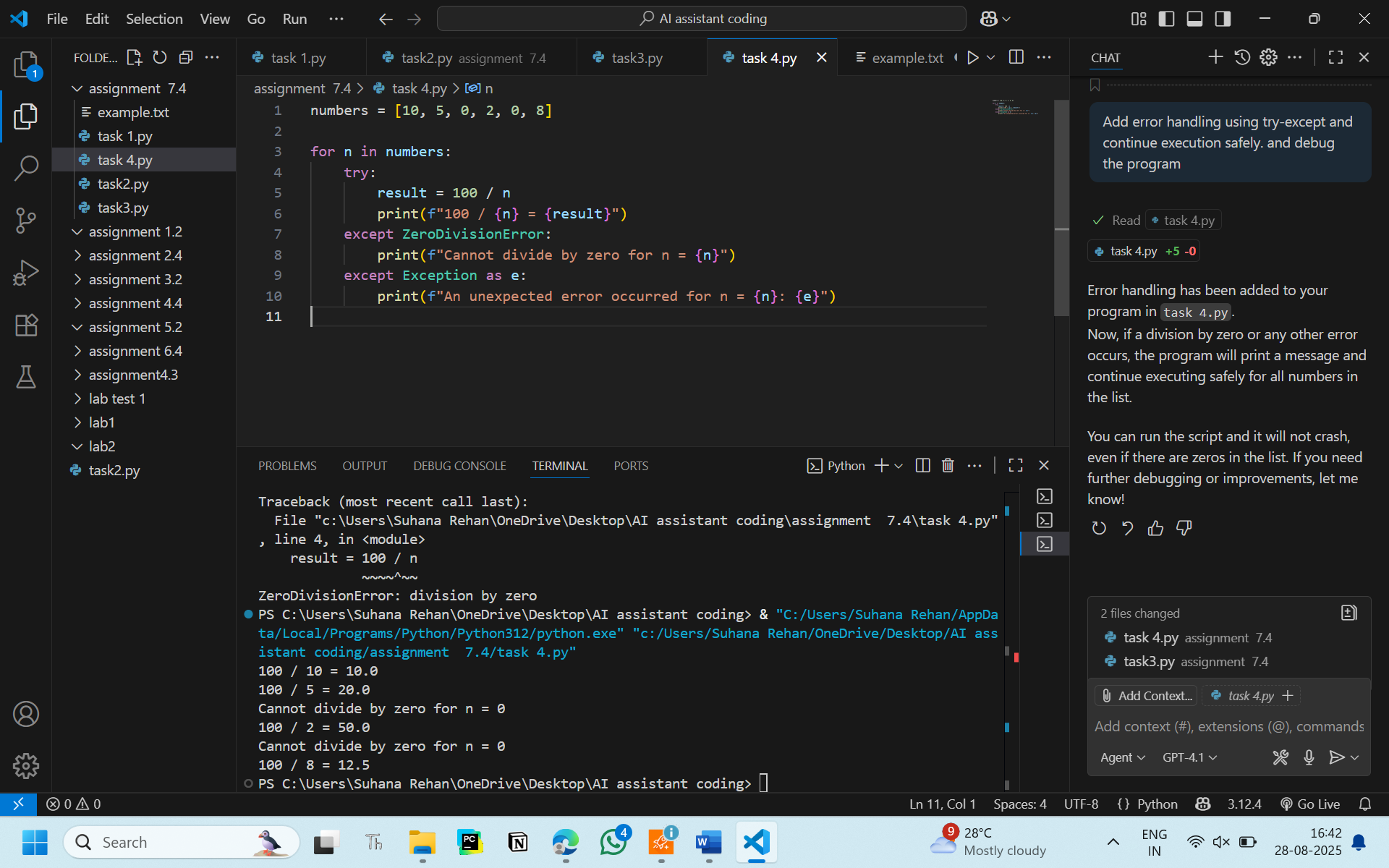
        print(f"100 / {n} = {result}")

    except ZeroDivisionError:

        print(f"Cannot divide by zero for n = {n}")

    except Exception as e:

        print(f"An unexpected error occurred for n = {n}: {e}")



**Output:**

100 / 10 = 10.0

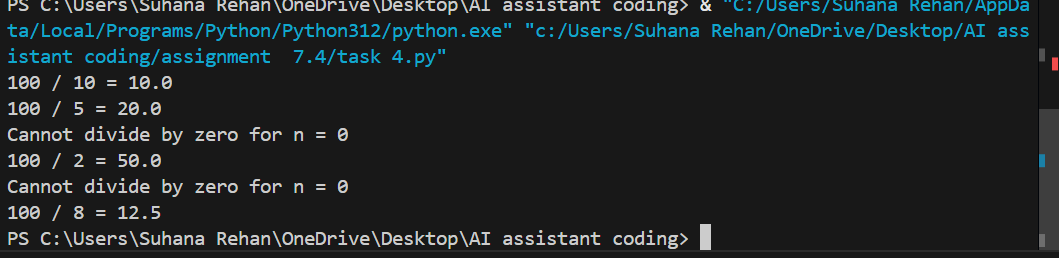
100 / 5 = 20.0

Cannot divide by zero for n = 0

100 / 2 = 50.0

Cannot divide by zero for n = 0

100 / 8 = 12.5

****

**Observation:**

I observed that the AI added error handling and exception block to the given code and solved the error from before

**Task 5:**

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

**Prompt:**

Correct the constructor and attribute usage

**Code with bug:**

class Student:

    def \_\_init\_\_(self, name, age, grade\_level):

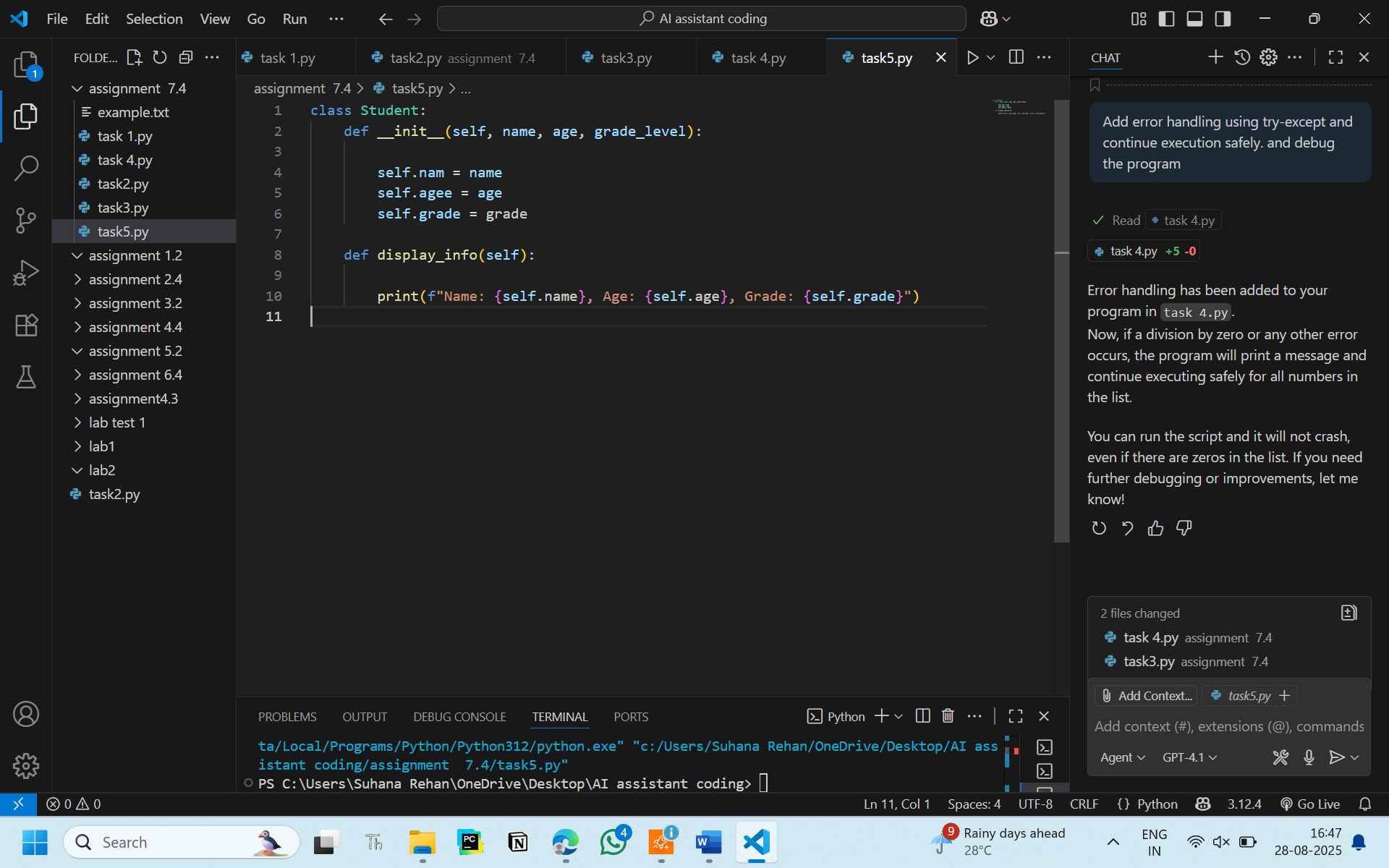
        self.nam = name

        self.agee = age

        self.grade = grade

    def display\_info(self):

        print(f"Name: {self.name}, Age: {self.age}, Grade: {self.grade}")



**Code without bug:**

class Student:

    def \_\_init\_\_(self, name, age, grade\_level):

        self.name = name

        self.age = age

        self.grade = grade\_level

    def display\_info(self):

        print(f"Name: {self.name}, Age: {self.age}, Grade: {self.grade}")

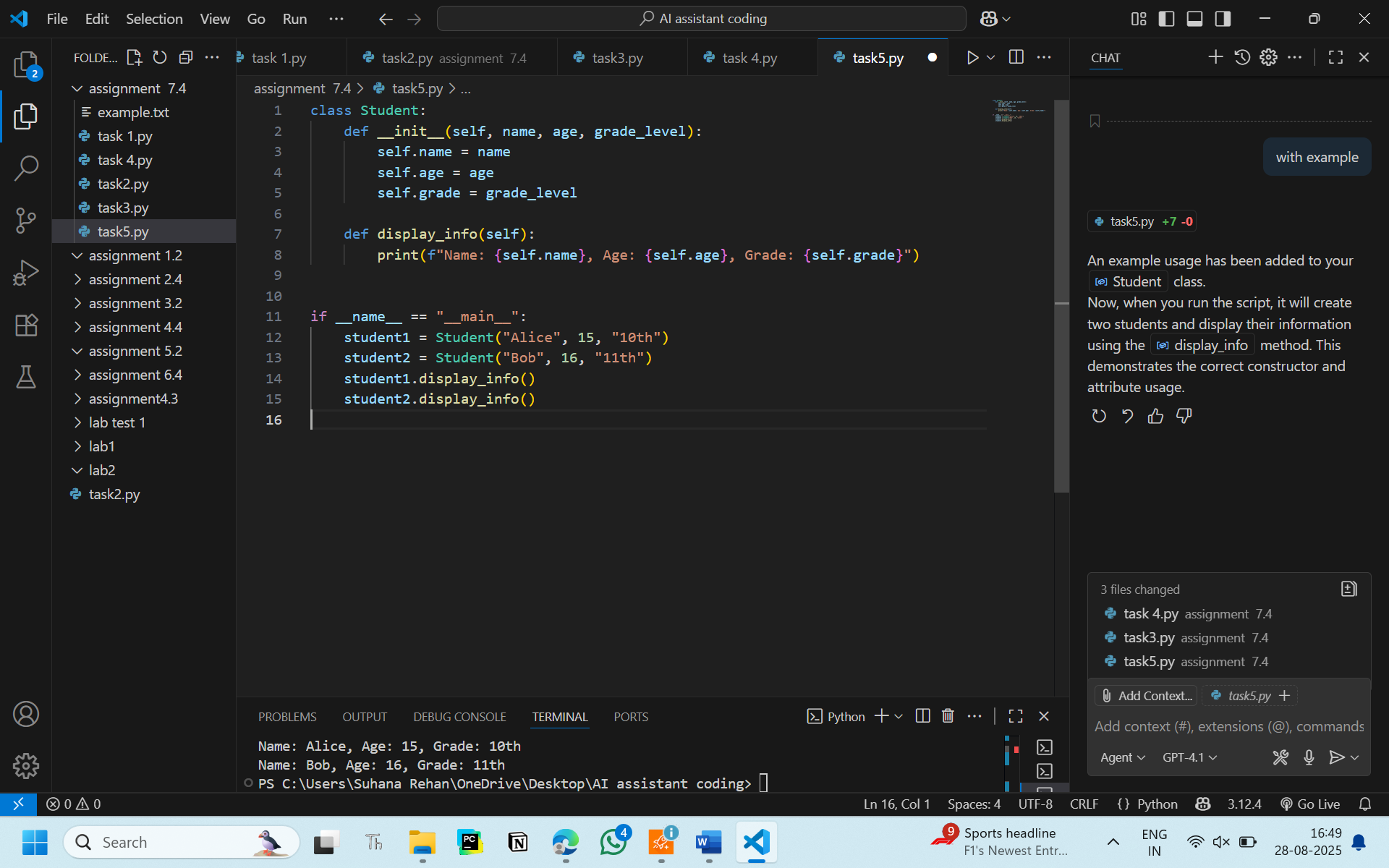
if \_\_name\_\_ == "\_\_main\_\_":

    student1 = Student("Alice", 15, "10th")

    student2 = Student("Bob", 16, "11th")

    student1.display\_info()

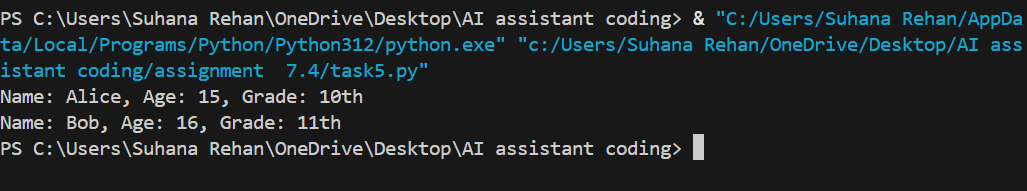
    student2.display\_info()



**OUTPUT:**

Name: Alice, Age: 15, Grade: 10th

Name: Bob, Age: 16, Grade: 11th

****

**Observation:**

I observed that AI corrected the constructor and debugged the program very smoothly