**AI ASSISTED CODING LAB ASSIGNMENT 7.4**

**ENROLLMENT NO :**2503A51L21

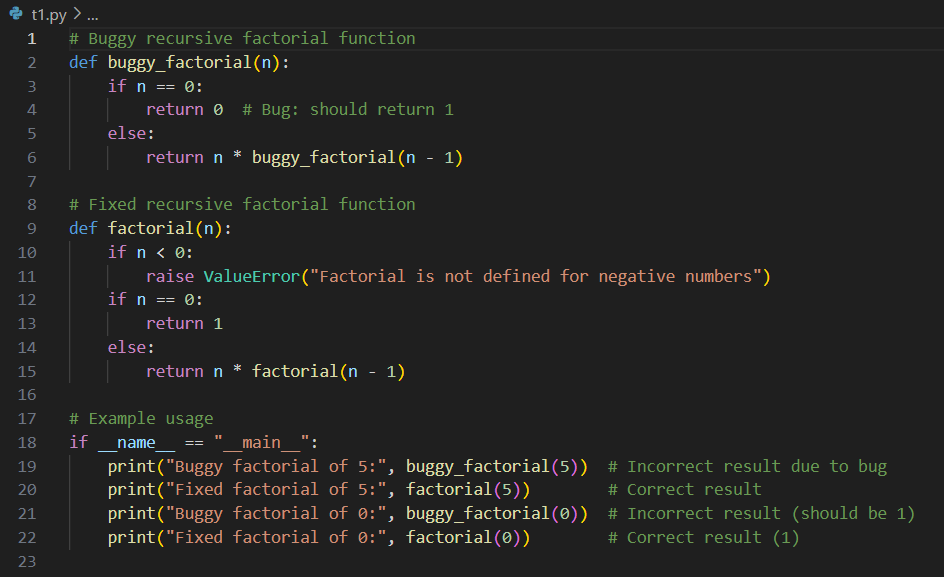
**BATCH NO:** 19

**NAME:** LUNGHIMPOU KAMEI

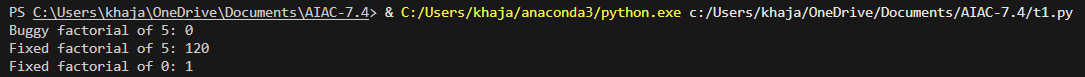
**TASK1:**

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:** Generate a buggy Python function that calculates the factorial of a number using recursion then detect and fix the logical or syntax errors. implement with example errors.



**OUTPUT:**



**OBSERVATION:**

1. The function now correctly checks for negative numbers and handles them.

2. Base case n == 0 returns 1.

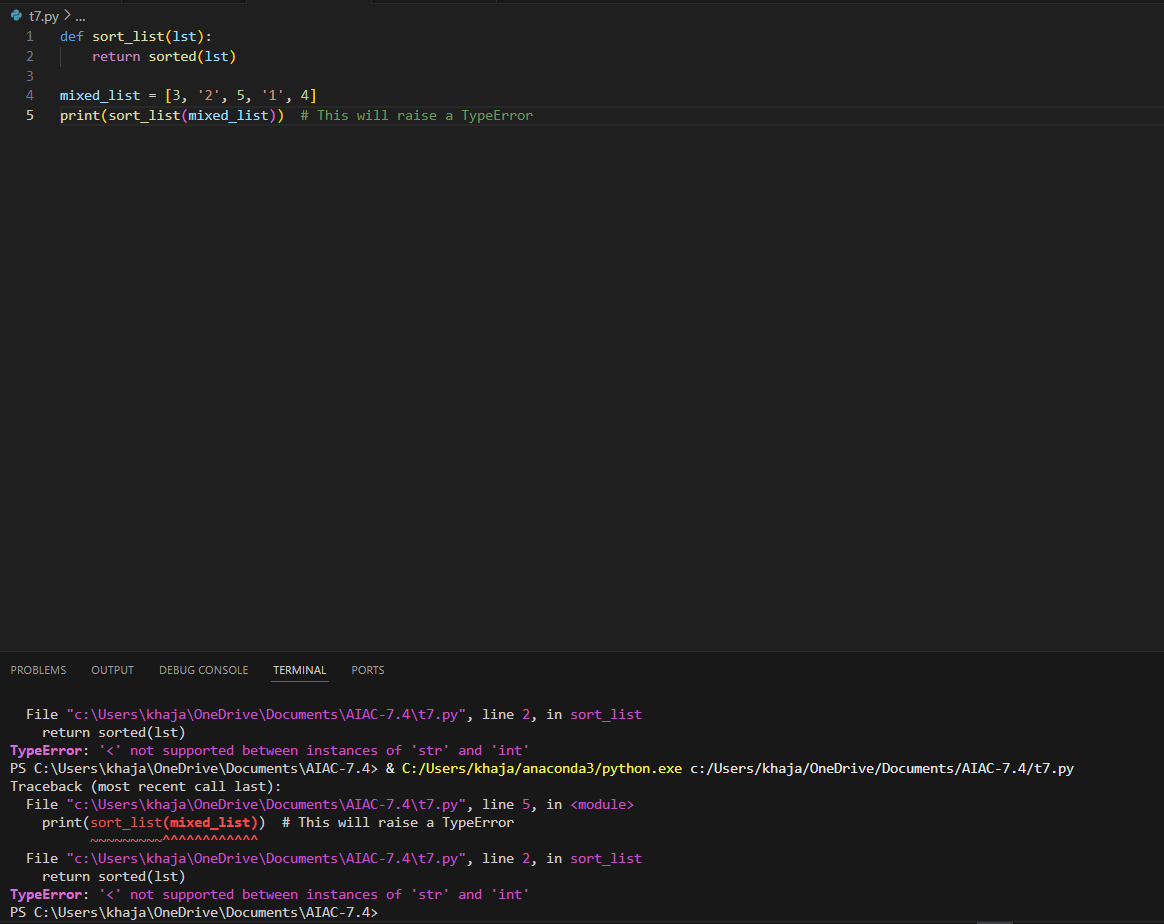
3. Recursive call factorial(n-1) multiplies the number down to 1.

**TASK2:**

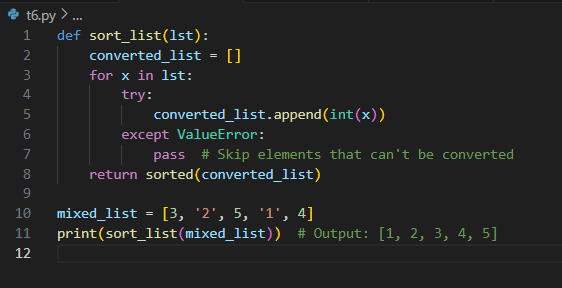
Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed  
integers and strings). PROMPT AI to detect the issue and fix the code for consistent sorting

**PROMPT:** Create a program that Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed  
integers and strings).

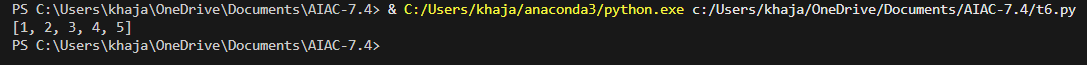
Error code:



After fixing the code:



**OUTPUT:**

****

**OBSERVATION:**

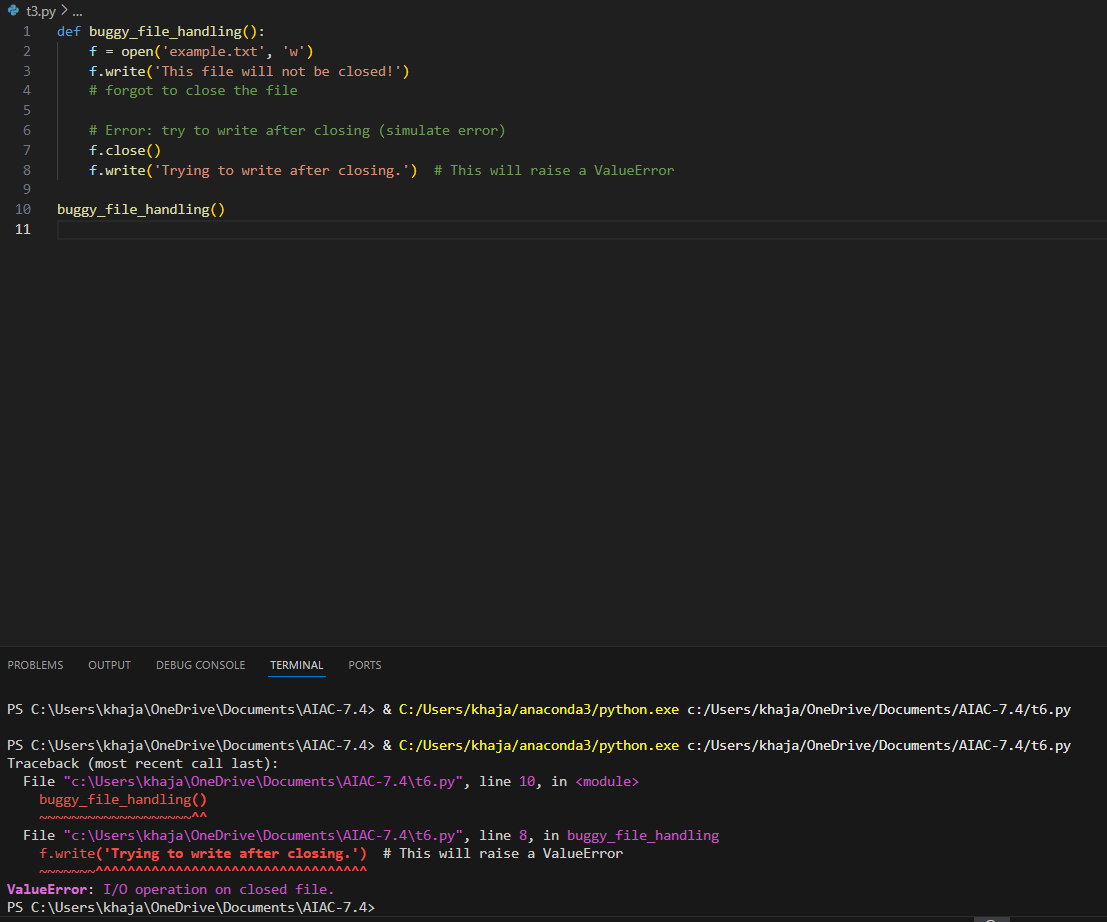
In the buggy code, the list contains both integers and strings, so using sorted() directly causes a TypeError because Python cannot compare numbers and strings. In the fixed code, using sorted(lst, key=str) converts all elements to strings before comparison, avoiding the error and allowing the list to be sorted lexicographically. As a result, numbers appear first as string representations followed by words, and the program runs without runtime errors.

**TASK3:**

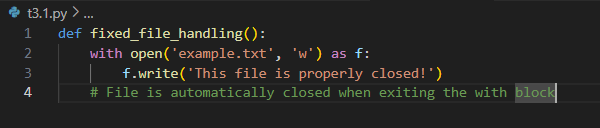
Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block).

**PROMPT:** Write a Python snippet for file handling that opens a file but forgets to close it.

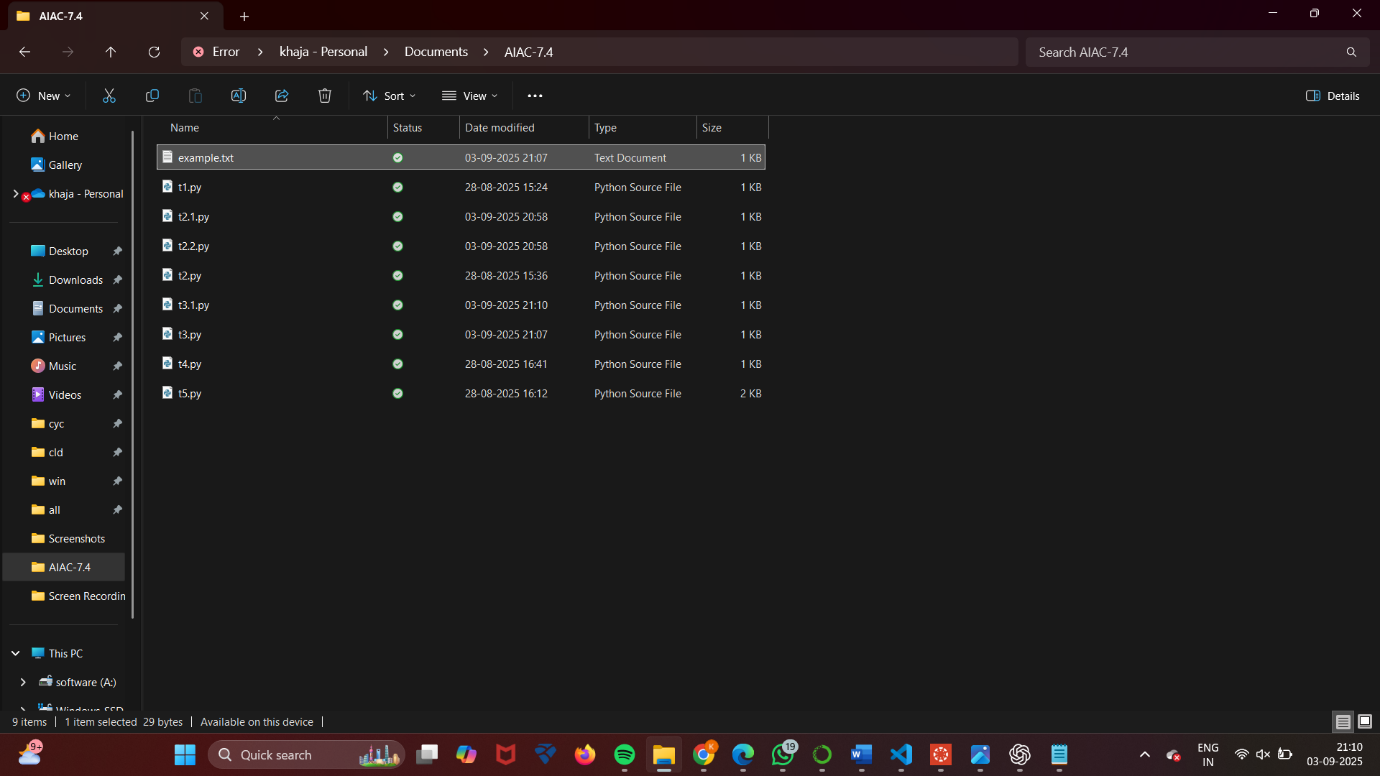
Error code:

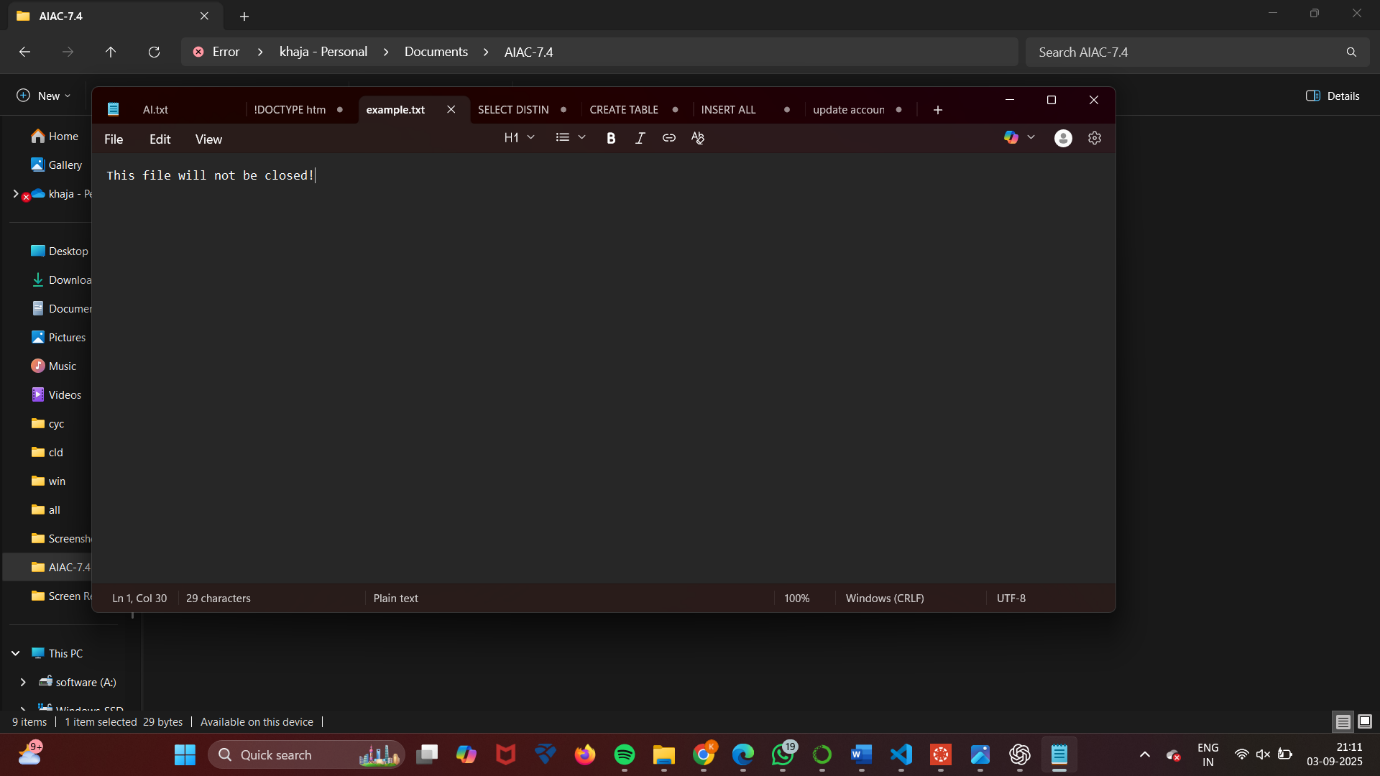


After fixing the code:



**OUTPUT:**

****

****

**OBSERVATION:**

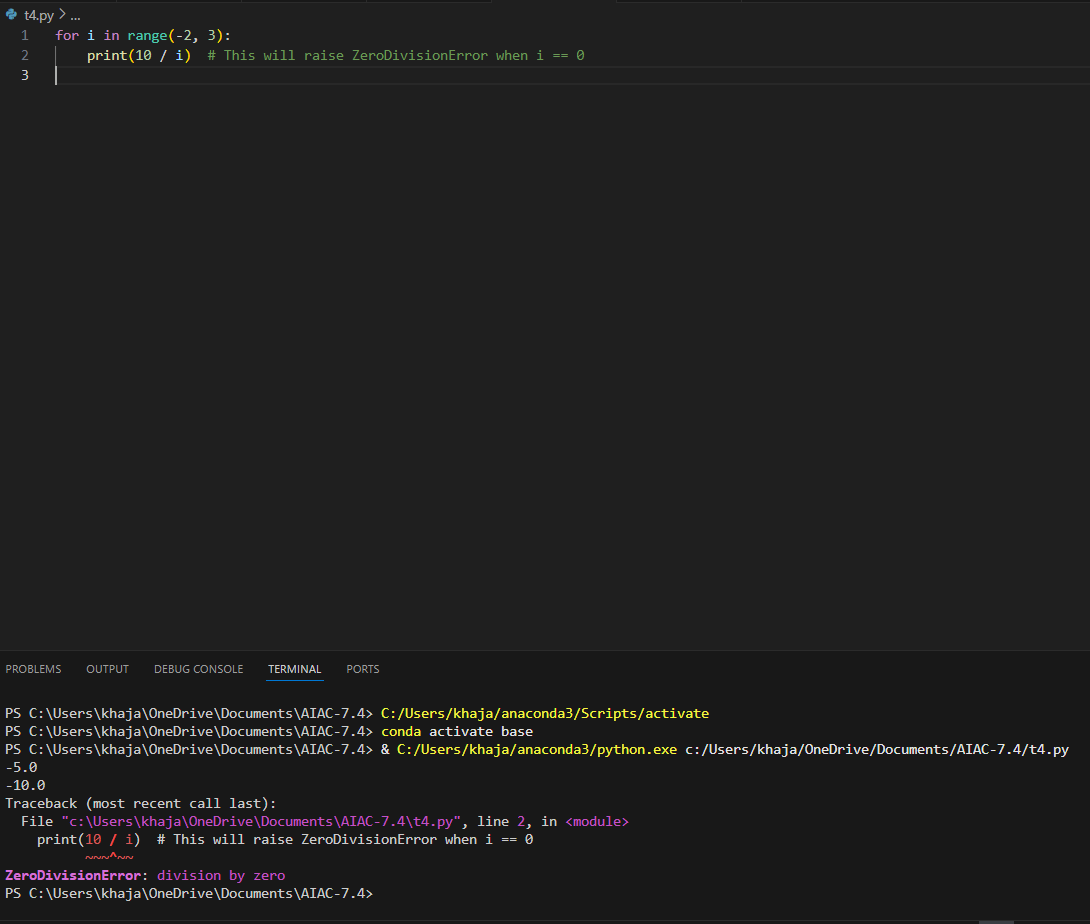
I observed that the file-handling code opened a file but didn’t close it, which could cause issues. Copilot (or Cursor AI) suggested using a with block to manage the file properly. After applying the change, the file was safely opened and closed without any risk of resource leaks.

**TASK4:**

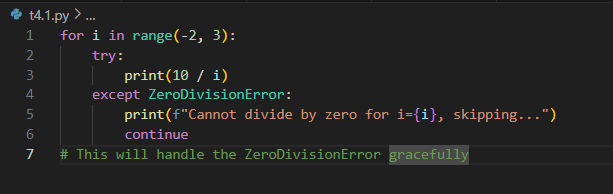
Provide a piece of code with a ZeroDivisionError inside a loop. Ask AI to add error handling  
using try-except and continue execution safely

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

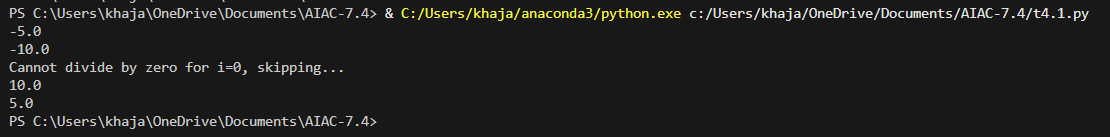
Error code:



After fixing the code:



**OUTPUT:**

****

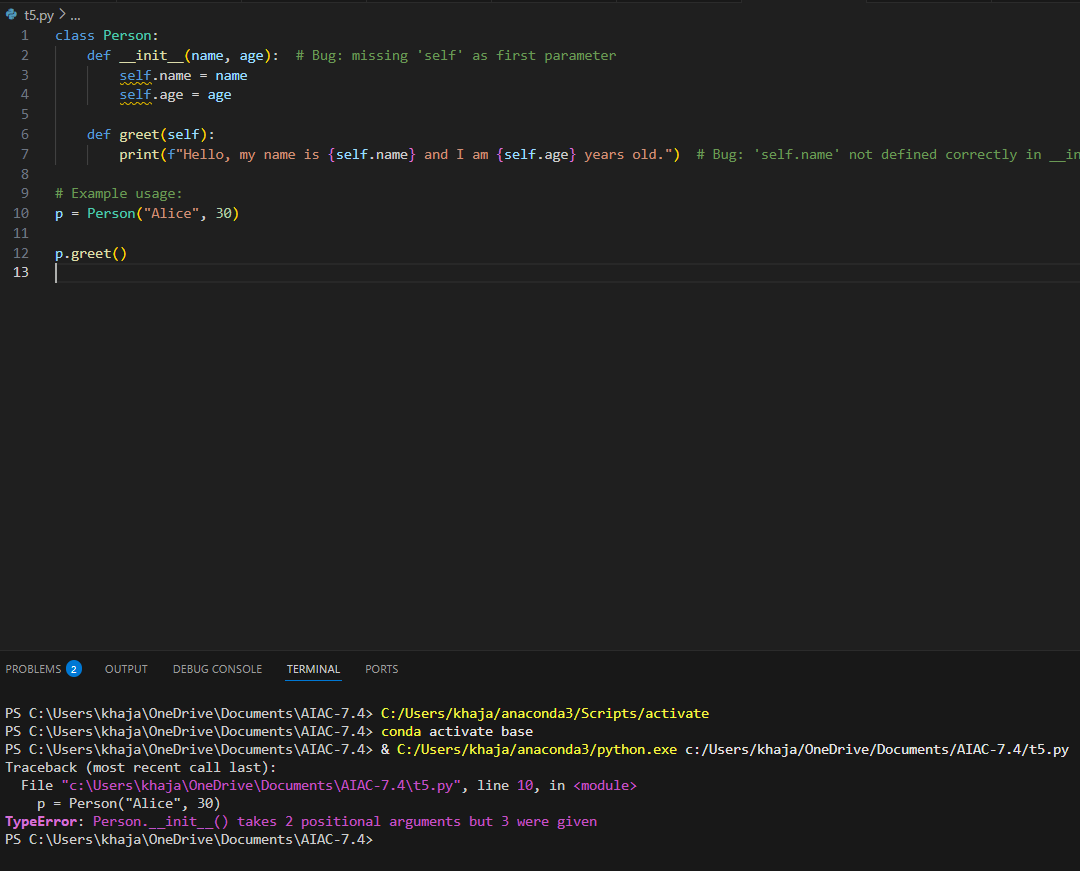
**OBSERVATION:** I observed that the loop crashed when it tried to divide by zero. Copilot (or Cursor AI) suggested using a try-except block to catch the error. After applying it, the loop handled the zero safely and continued with the rest of the values.

**TASK5:**

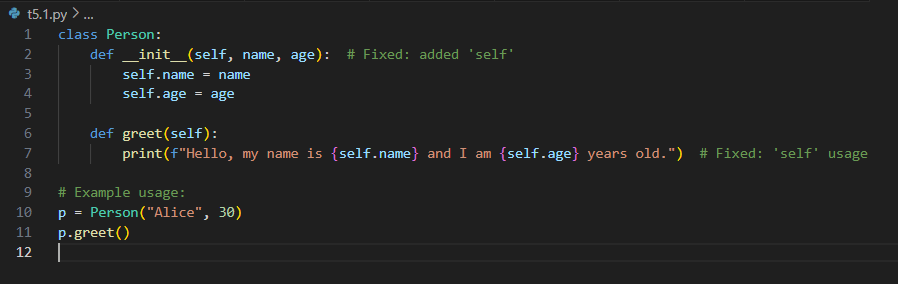
Include a buggy class definition with incorrect \_\_init\_\_ parameters or attribute references. Ask AI to analyze and correct the constructor and attribute usage.

**PROMPT:** Include a buggy class definition with incorrect \_\_init\_\_ parameters or attribute references.

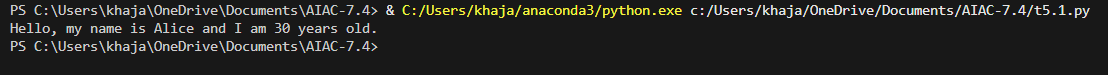
Error code:



After fixing the code:



**OUTPUT:**

****

**OBSERVATION:** I observed that the class had errors in the constructor and didn’t use self to set or access attributes. Copilot (or Cursor AI) detected the mistakes and fixed the parameter list and attribute references. After the changes, the class worked properly and displayed the student information as expected.