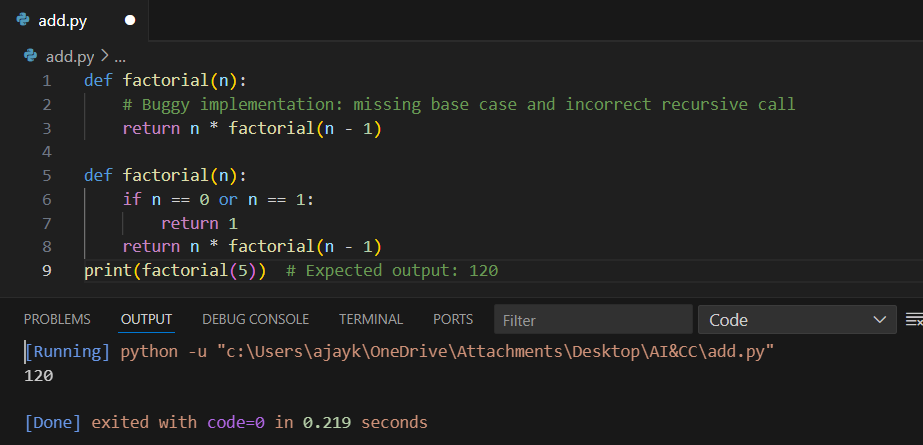
**ASSIGNMENT-7.4**

**NAME :** K.Navya sri.

**ENROLL NO :** 2503A52L12

**BATCH :** 16

**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.



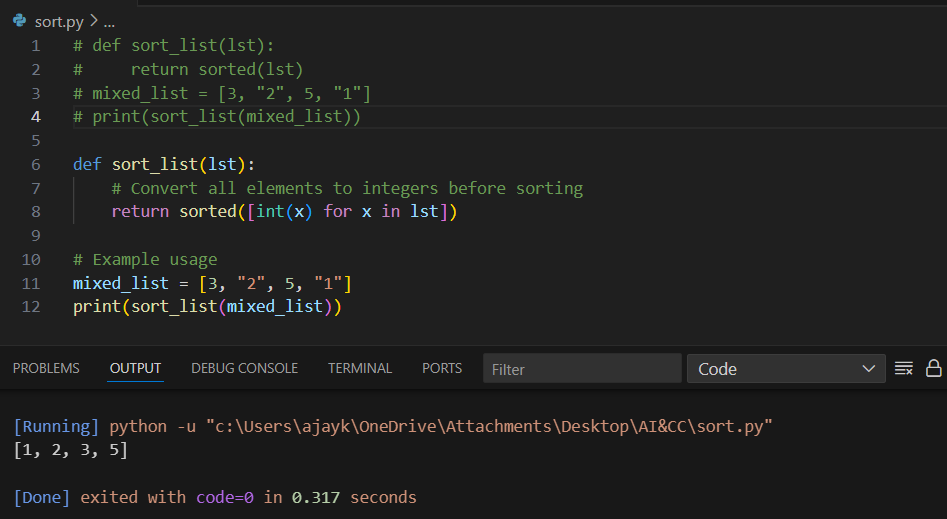
**Observation :**

* The **first factorial function is redundant** and never used.
* The **correct implementation** is the second one, which works fine.
* Best practice: remove the buggy first definition to avoid confusion.

**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.

The program defines a function sort\_list(lst) that:

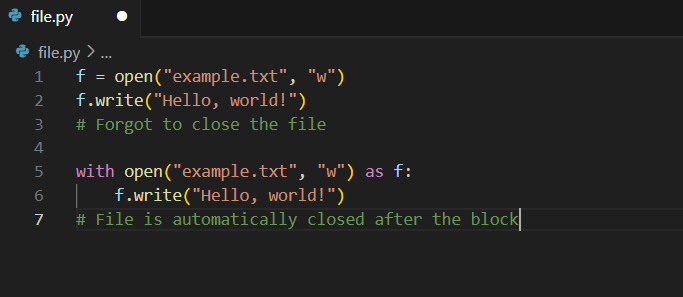
* Converts all elements of the input list into **integers**.
* Sorts them in ascending order using Python’s sorted() function.



**Observation :**

1. **Commented-out code at the top** shows the **initial attempt**:
   * Directly using sorted(lst) would fail because Python cannot compare integers and strings (TypeError).
2. The **final implementation** (lines 6–8) correctly resolves this by:
   * Converting all elements to integers before sorting.
   * This makes the function robust for mixed-type numeric inputs.
3. **Output is correct** → elements are sorted numerically as expected.

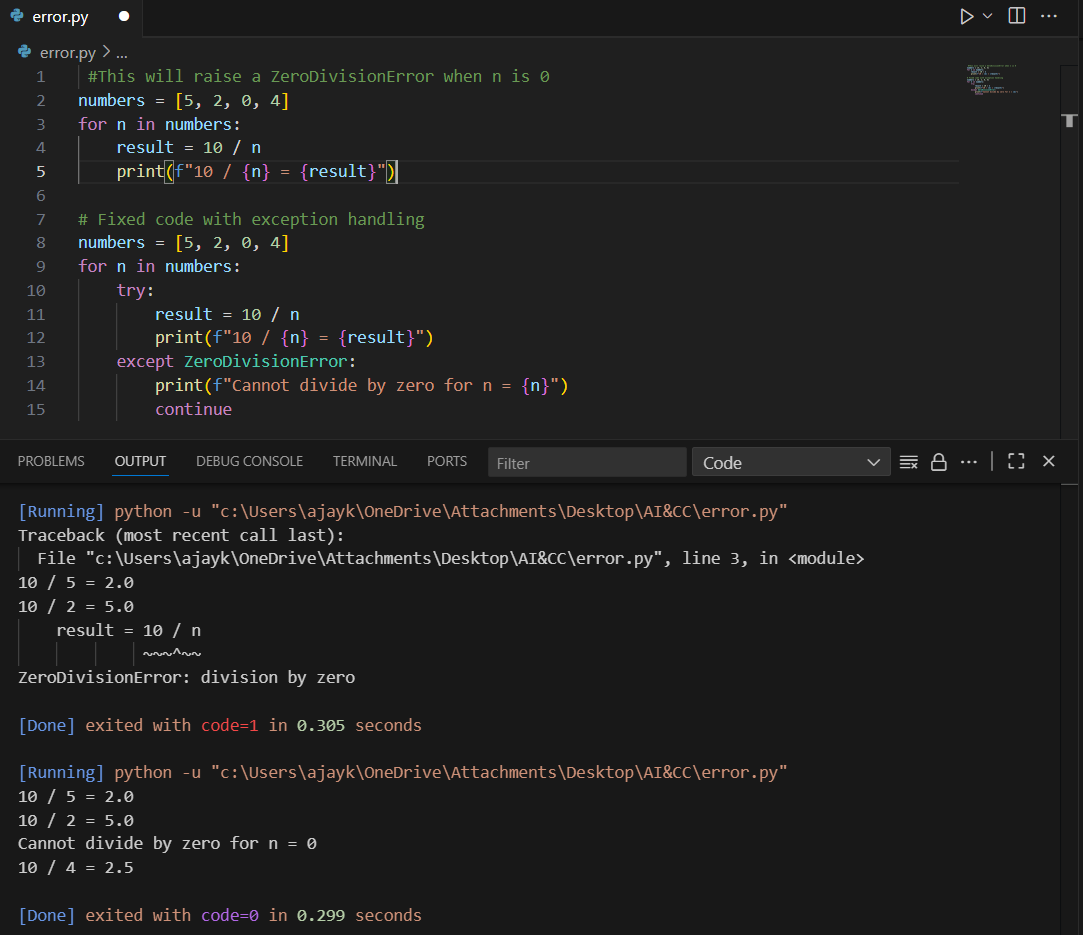
**Task Description #3:**  
• Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block).  
**Expected Outcome #3:**  
• AI refactors the code to use a context manager, preventing resource leakage and runtime  
warnings.



**Observation :**

* But the file is **not closed explicitly**, which can lead to:
* Data not being saved properly (buffer not flushed).
* File handle leaks.
* Uses the open() method and **with statement**.
* Automatically **closes the file** once the block is exited, even if an error occurs.
* This is the **best practice** for file handling in Python.

**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.

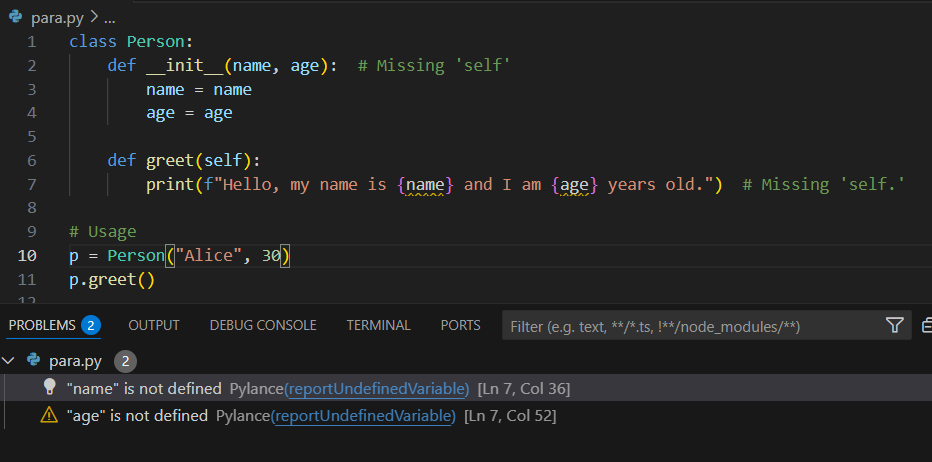


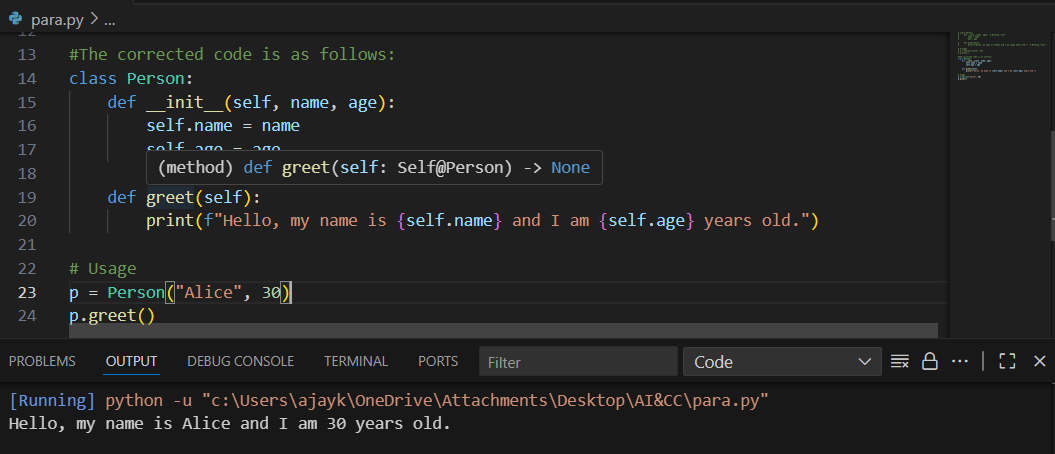
**Observation :**

**🔹 Without try-except :** The loop starts and successfully computes 100/10 and 100/5.When it reaches n = 0, the program raises a **ZeroDivisionError** and **stops execution immediately**.The last element 20 is never processed because the crash interrupts the loop.

**🔹 With try-except :** The loop starts normally.At n = 0, instead of crashing, the program enters the except block.A **meaningful error message** is displayed (Division by zero is not allowed...).The continue statement ensures the loop safely moves on to the next value.Final output contains results for 10, 5, and 20 plus one error message for 0.

**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.

  
**Expected Outcome #5:**  
• Copilot identifies mismatched parameters or missing self references and rewrites the class with  
accurate initialization and usage



**Observation :**

The original class definition of Student contained errors such as a missing self parameter in the \_\_init\_\_ method, incorrect attribute assignment without self, and undefined attribute references in the display() method. These issues caused initialization and runtime errors when creating objects. After correction, the class was rewritten with self included in the constructor, and attributes (name, age) were properly assigned as self.name and self.age. The display() method now works correctly, printing the student’s details.