

Assignment 7.4

Name: Zobiya Fatima

Hall Ticket: 2303A51879

Batch No: 14

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

Task 1: Debugging a Recursive Calculation Module Scenario

You are maintaining a utility module in a software project that performs mathematical computations. One function is meant to calculate the factorial of a number, but users are reporting crashes or incorrect outputs.

Task Description

You are given a Python function intended to calculate the factorial of a number using recursion, but it contains logical or syntactical errors (such as a missing base condition or incorrect recursive call).

Use GitHub Copilot or Cursor AI to:

- Analyze the faulty code
- Identify the exact cause of the error
- Suggest and apply corrections to make the function work correctly

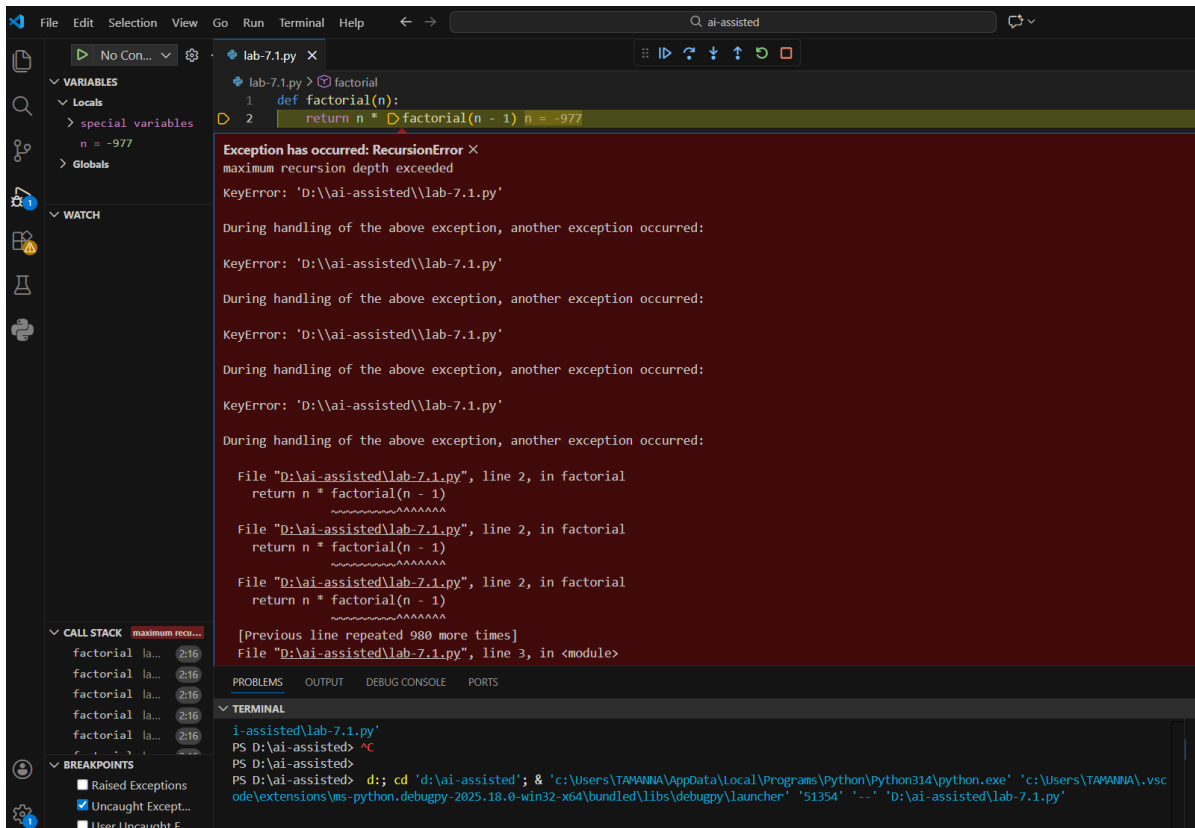
Document how the AI detected the issue and what changes were made.

Expected Outcome

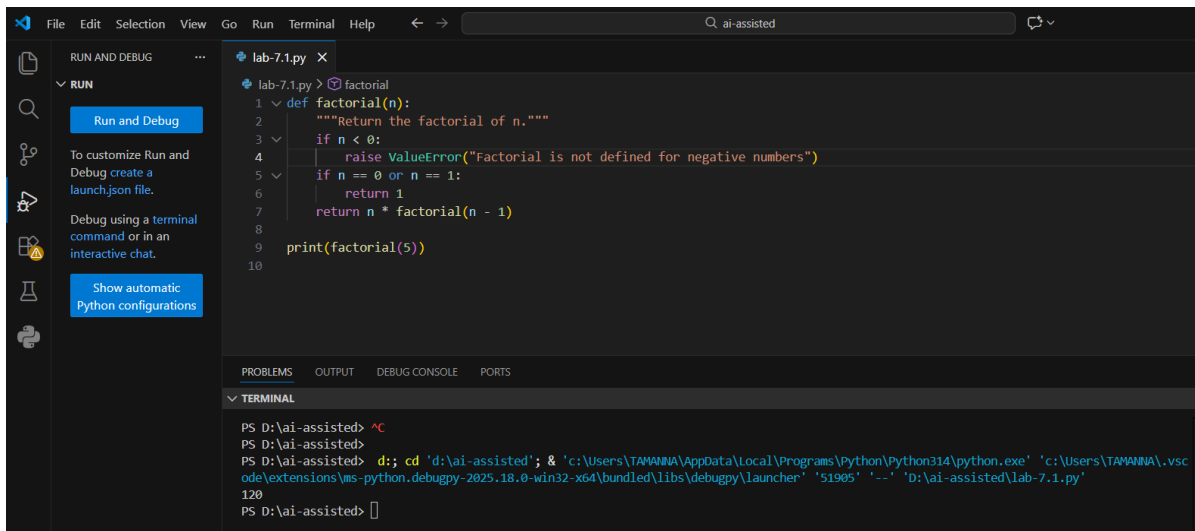
- A corrected recursive factorial function
- AI-generated explanation identifying:
 - The missing or incorrect base case
 - The corrected recursive logic
- Sample input/output demonstrating correct execution

A Faulty Code:

```
lab-7.1.py X
lab-7.1.py > ...
1 def factorial(n):
2     return n * factorial(n - 1)
3 factorial(5)
4
```



Prompt: Rectify the code and give a proper working code.



Summary:

- The faulty code was missing a base case, causing infinite recursive calls and a Recursion Error.
- The corrected code adds proper base conditions (`n == 0` or `n == 1`) to stop recursion safely.
- An additional validation was included to handle negative inputs by raising a clear error.

Task 2: Fixing Data Type Errors in a Sorting Utility

Scenario

You are developing a data processing script that sorts user input values. The program crashes when users enter mixed data types.

Task Description

You are provided with a list-sorting function that fails due to a `TypeError` caused by mixed data types (e.g., integers and strings).

Use GitHub Copilot or Cursor AI to:

- Detect the root cause of the runtime error
- Modify the code to ensure consistent sorting (by filtering or type conversion)
- Prevent the program from crashing

Explain the debugging steps followed by the AI.

Expected Outcome

- A corrected sorting function
- AI-generated solution handling type inconsistencies
- Successful sorting without runtime errors
- Explanation of how the fix improves robustness

Incorrect Code:

```
#Task 2
def sort_numbers(data):
    return sorted(data)

items = [10, "5", 3, "20", 7]
print([sort_numbers(items)])
```

```
0, '5', 3,... 6 | return n * factorial(n - 1)
7
8 | print(factorial(5))"""
9 | #Task 2
10 | def sort_numbers(data):
11 -> | return sorted(data) data = [10, '5', 3, '20', 7]

Exception has occurred: TypeError ×
'<' not supported between instances of 'str' and 'int'

File "D:\ai-assisted\lab-7.1.py", line 11, in sort_numbers
    return sorted(data)
File "D:\ai-assisted\lab-7.1.py", line 14, in <module>
    print(sort_numbers(items))
~~~~~
TypeError: '<' not supported between instances of 'str' and 'int'

12
13 | items = [10, "5", 3, "20", 7]
14 | print(sort_numbers(items))
15

PROBLEMS OUTPUT DEBUG CONSOLE PORTS

▼ TERMINAL

120
PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d;; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNA\AppData\Local\Programs\Python\Py
thon314\python.exe' 'c:\Users\TAMANNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bu
ndled\libs\debugpy\launcher' '63589' '--' 'D:\ai-assisted\lab-7.1.py'
```

Prompt: Rectify the faulty code and give a correct one with user input

```
lab-7.1.py X
lab-7.1.py > sort_numbers

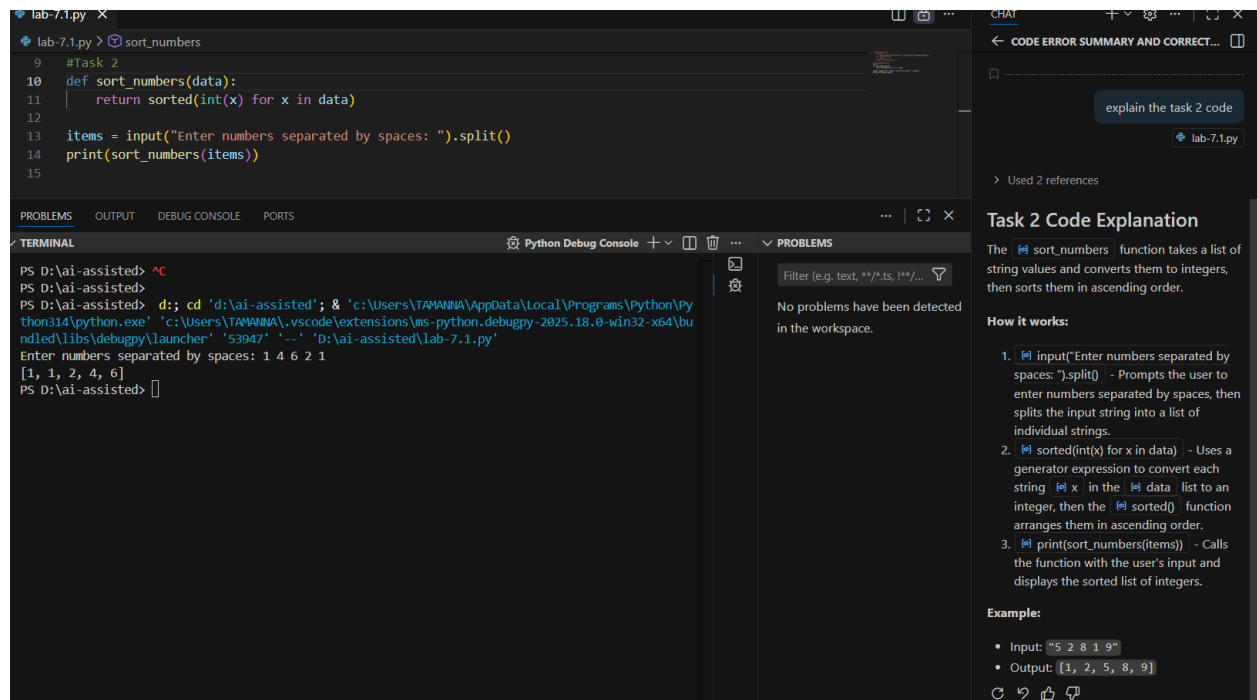
9 | #Task 2
10 | def sort_numbers(data):
11 | | return sorted(int(x) for x in data)
12 |
13 | items = input("Enter numbers separated by spaces: ").split()
14 | print(sort_numbers(items))
15 |

PROBLEMS OUTPUT DEBUG CONSOLE PORTS

▼ TERMINAL Python Debug Console + - [] ...

PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d;; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNA\AppData\Local\Programs\Python\Py
thon314\python.exe' 'c:\Users\TAMANNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bu
ndled\libs\debugpy\launcher' '53947' '--' 'D:\ai-assisted\lab-7.1.py'
Enter numbers separated by spaces: 1 4 6 2 1
[1, 1, 2, 4, 6]
PS D:\ai-assisted> |
```

Error Explanation:



Summary:

- The faulty code failed because it attempted to sort a list containing both strings and integers, which caused a **Type Error** during comparison.
- The corrected code converts every input value to an integer before sorting, ensuring all elements are of the same data type.
- This change prevents runtime crashes and guarantees consistent, correct numerical sorting.

Task 3: Improving File Handling Reliability

Scenario

A backend script reads data from files regularly. Over time, the system shows performance issues due to improper resource management.

Task Description

You are given a Python file-handling snippet that opens a file but does not explicitly close it. Use GitHub Copilot or Cursor AI to:

- Identify the potential problem in the code
- Refactor it using best practices (such as a context manager)
- Ensure safe and reliable file handling

Briefly describe why the revised approach is better.

Expected Outcome

- Refactored code using the with open() statement
- AI explanation highlighting prevention of resource leaks
- Clean execution without warnings or errors

Faulty Code:

```
#Task 3
def read_file(filename):
    file = open(filename, "r")
    data = file.read()
    return data

content = read_file("example.txt")
print(content)
```

The screenshot shows a code editor with a Python script and its execution output. The script defines a function `read_file` that opens a file and returns its content. It then calls `read_file("example.txt")` and prints the result. The execution fails with a `FileNotFoundError` because the file `example.txt` does not exist.

```
17 #Task 3
18 def read_file(filename):
19     file = open(filename, "r")
20     data = file.read()
21     return data
22
23 content = read_file("example.txt")
24 print(content)
25
26
```

Exception has occurred: FileNotFoundError ×
[Errno 2] No such file or directory: 'example.txt'

File "D:\ai-assisted\lab-7.1.py", line 19, in read_file
file = open(filename, "r")
File "D:\ai-assisted\lab-7.1.py", line 23, in <module>
content = read_file("example.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'

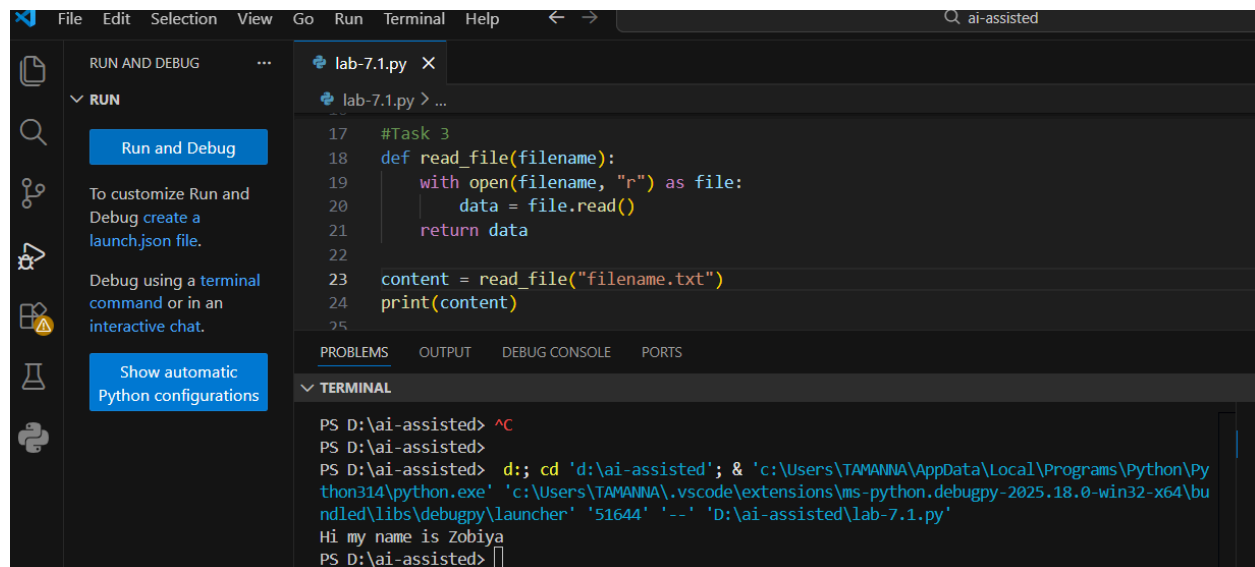
PROBLEMS OUTPUT DEBUG CONSOLE PORTS

TERMINAL

```
PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d:; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNA\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\TAMANNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bu
ndled\libs\debugpy\launcher' '63199' '--' 'D:\ai-assisted\lab-7.1.py'
```

STACK [Errno 2] No su...

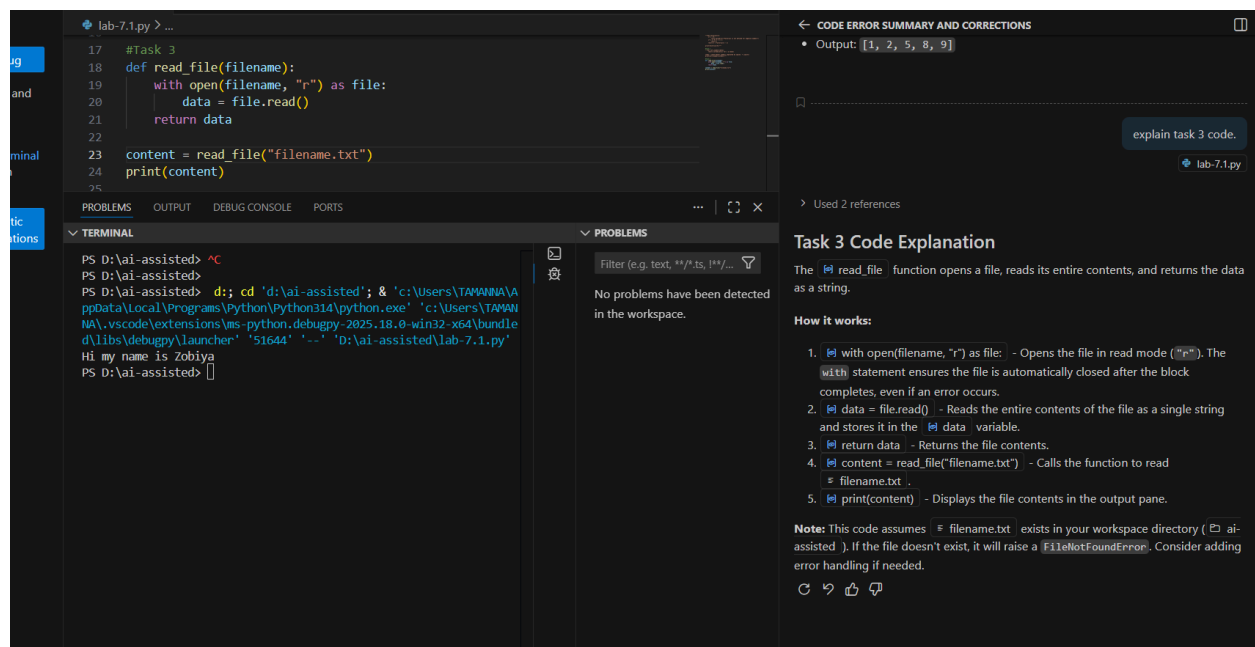
Prompt: Identify the potential problem in the code and refactor it ensuring safe and reliable file handling.



```
17 #Task 3
18 def read_file(filename):
19     with open(filename, "r") as file:
20         data = file.read()
21     return data
22
23 content = read_file("filename.txt")
24 print(content)
25
```

```
PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d;; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNNA\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\TAMANNNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '51644' '--' 'D:\ai-assisted\lab-7.1.py'
Hi my name is Zobiya
PS D:\ai-assisted>
```

Error Explanation:



Task 3 Code Explanation

The `read_file` function opens a file, reads its entire contents, and returns the data as a string.

How it works:

- `with open(filename, "r") as file:` - Opens the file in read mode ("r"). The `with` statement ensures the file is automatically closed after the block completes, even if an error occurs.
- `data = file.read()` - Reads the entire contents of the file as a single string and stores it in the `data` variable.
- `return data` - Returns the file contents.
- `content = read_file("filename.txt")` - Calls the function to read `filename.txt`.
- `print(content)` - Displays the file contents in the output pane.

Note: This code assumes `filename.txt` exists in your workspace directory (D:\ai-assisted). If the file doesn't exist, it will raise a `FileNotFoundError`. Consider adding error handling if needed.

Summary:

- The faulty code opened a file using `open()` but did not close it, which could cause resource leaks if the program runs for a long time or crashes.
- The corrected code uses a **With** context manager to ensure the file is automatically closed after reading.
- This change guarantees safe, reliable file handling even if an error occurs during file operations.

Task 4: Handling Runtime Errors Gracefully in Loops

Scenario

You are working on a data analysis script that processes a list of values. Some values cause runtime errors, but the program should continue processing remaining data.

Task Description

You are provided with a code snippet containing a `ZeroDivisionError` inside a loop.

Use GitHub Copilot or Cursor AI to:

- Detect the exact location of the error
- Add appropriate exception handling using try-except
- Ensure the loop continues executing safely

Document how AI improved the fault tolerance of the program.

Expected Outcome

- Updated code with proper exception handling
- Meaningful error messages instead of program crashes
- Successful execution for all valid inputs

Faulty Code:

```
26 #Task 4
27 numbers = [10, 5, 0, 2, 0]
28
29 for num in numbers:
30     result = 100 / num
31     print(result)
```

```
26 #Task 4
27 numbers = [10, 5, 0, 2, 0]
28
29 for num in numbers: numbers = [10, 5, 0, 2, 0]
30 result = 100 / num result = 20.0, num = 0
```

```
Exception has occurred: ZeroDivisionError X
division by zero

File "D:\ai-assisted\lab-7.1.py", line 30, in <module>
    result = 100 / num
           ~~~~~
ZeroDivisionError: division by zero
```

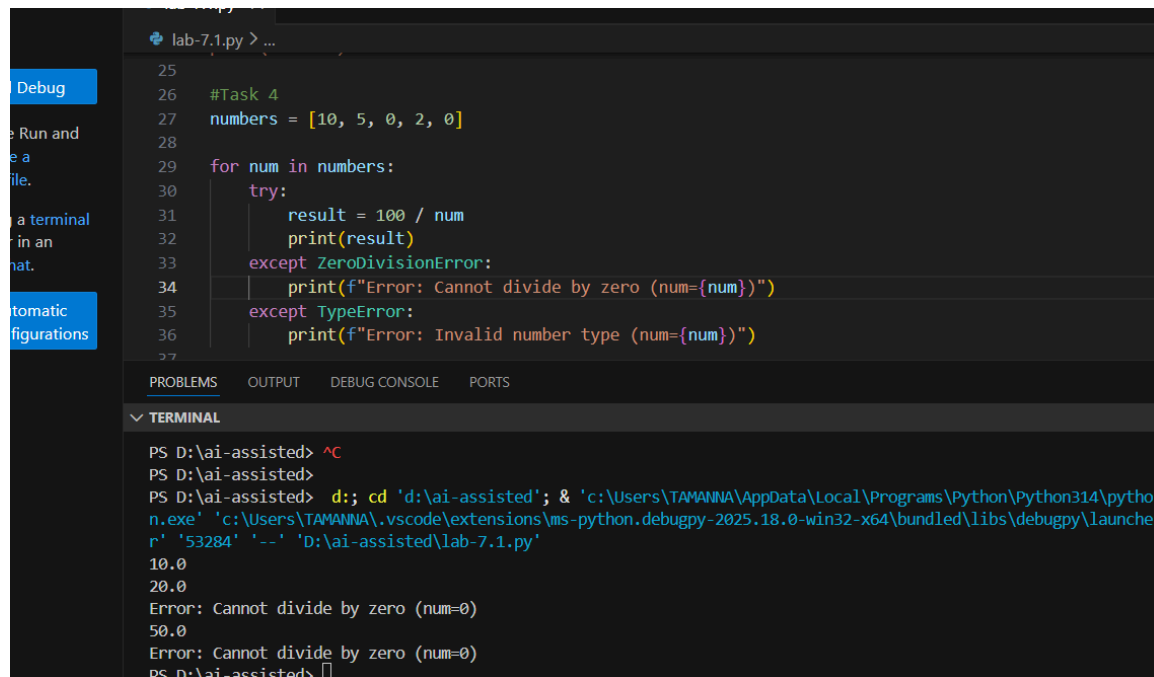
```
31 | print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS

TERMINAL

```
PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d;; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNA\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\TAMANNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51644' '--' 'D:\ai-assisted\lab-7.1.py'
Hi my name is Zobiya
PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d;; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNA\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\TAMANNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57337' '--' 'D:\ai-assisted\lab-7.1.py'
10.0
20.0
```


Prompt: Rectify this code and give a correct version of it.



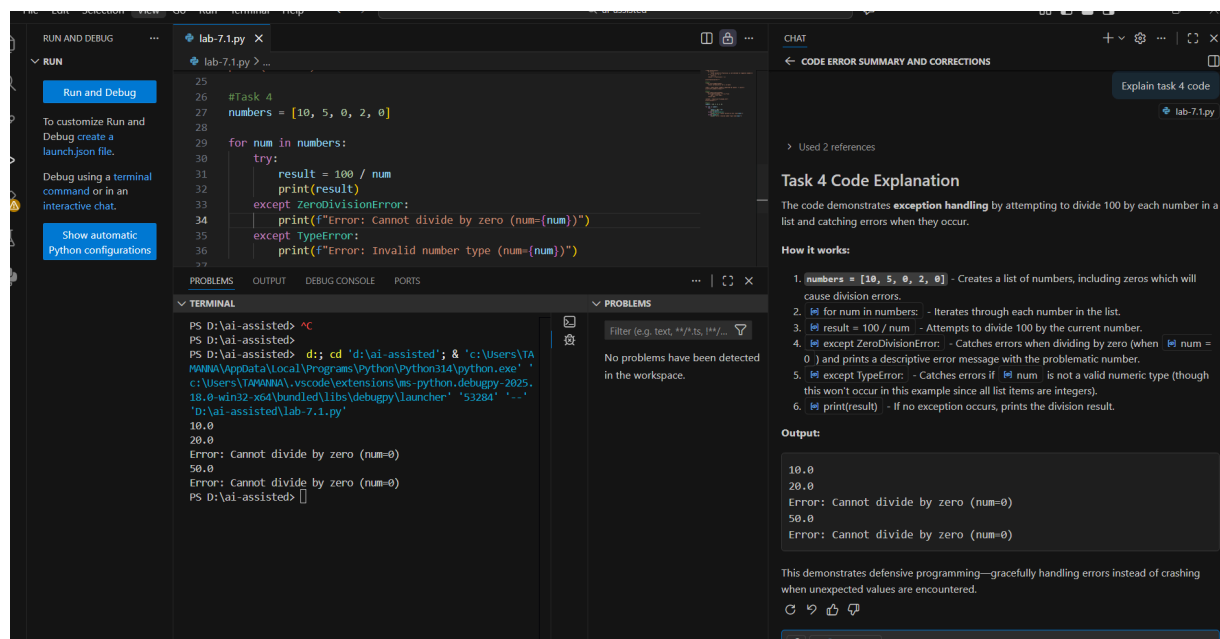
```
lab-7.1.py > ...
25
26 #Task 4
27 numbers = [10, 5, 0, 2, 0]
28
29 for num in numbers:
30     try:
31         result = 100 / num
32         print(result)
33     except ZeroDivisionError:
34         print(f"Error: Cannot divide by zero (num={num})")
35     except TypeError:
36         print(f"Error: Invalid number type (num={num})")
37
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS

TERMINAL

```
PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNA\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\TAMANNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53284' '--' 'D:\ai-assisted\lab-7.1.py'
10.0
20.0
Error: Cannot divide by zero (num=0)
50.0
Error: Cannot divide by zero (num=0)
PS D:\ai-assisted>
```

Error Explanation:



Task 4 Code Explanation

The code demonstrates **exception handling** by attempting to divide 100 by each number in a list and catching errors when they occur.

How it works:

1. `numbers = [10, 5, 0, 2, 0]` - Creates a list of numbers, including zeros which will cause division errors.
2. `for num in numbers:` - Iterates through each number in the list.
3. `result = 100 / num` - Attempts to divide 100 by the current number.
4. `except ZeroDivisionError:` - Catches errors when dividing by zero (when `num = 0`) and prints a descriptive error message with the problematic number.
5. `except TypeError:` - Catches errors if `num` is not a valid numeric type (though this won't occur in this example since all list items are integers).
6. `print(result)` - If no exception occurs, prints the division result.

Output:

```
10.0
20.0
Error: Cannot divide by zero (num=0)
50.0
Error: Cannot divide by zero (num=0)
```

This demonstrates defensive programming—gracefully handling errors instead of crashing when unexpected values are encountered.

Summary:

- The original issue occurred when the loop attempted to divide by zero, causing a **ZeroDivisionError** and stopping execution.
- The updated code uses **try-except** blocks to catch division-by-zero and type-related errors at the exact failure point.
- This ensures the loop continues running safely while reporting errors instead of crashing.

Task 5: Debugging Class Initialization Errors

Scenario

A class written by a junior developer is throwing unexpected errors when objects are created or attributes are accessed.

Task Description

You are given a Python class with:

- Incorrect `__init__` parameters
- Missing or incorrect attribute references (e.g., missing `self`)

Use GitHub Copilot or Cursor AI to:

- Analyze the class definition
- Identify constructor and attribute issues
- Correct the class so objects initialize and behave correctly

Explain the corrections suggested by the AI.

Expected Outcome

- A corrected class definition
- Proper use of `self` and constructor parameters
- AI-assisted explanation of the original errors and fixes
- Sample object creation and method usage

Faulty Code:

```
38 #Task 5
39 class Student:
40     def __init__(name, age):
41         name = name
42         age = age
43
44     def display_info():
45         print("Name:", name)
46         print("Age:", age)
47
48 s1 = Student("Aisha", 20)
49 s1.display_info()
```

```
37
38 #Task 5
39 class Student:
40     def __init__(name, age):
41         name = name
42         age = age
43
44     def display_info():
45         print("Name:", name)
46         print("Age:", age)
47
48 s1 = Student("Aisha", 20)
```

Exception has occurred: TypeError ×
Student.__init__() takes 2 positional arguments but 3 were given
File "D:\ai-assisted\lab-7.1.py", line 48, in <module>
s1 = Student("Aisha", 20)
TypeError: Student.__init__() takes 2 positional arguments but 3 were given

```
49 s1.display_info()
50
```

Prompt: Identify constructor and attribute issues and give a correct class definition with user input.

```
lab-7.1.py > Student > __init__
37
38 #Task 5
39 class Student:
40     def __init__(self, name, age):
41         self.name = name
42         self.age = age
43     def display_info(self):
44         print("Name:", self.name)
45         print("Age:", self.age)
46 name = input("Enter student name: ")
47 age = input("Enter student age: ")
48 s1 = Student(name, int(age))
49 s1.display_info()
50
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS

▼ TERMINAL Python Debug Console + ▾ 🗑️ ⋮

```
PS D:\ai-assisted> ^C
PS D:\ai-assisted>
PS D:\ai-assisted> d;; cd 'd:\ai-assisted'; & 'c:\Users\TAMANNA\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\TAMANNA\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62905' '--' 'D:\ai-assisted\lab-7.1.py'
Enter student name: Zobiya
Enter student age: 20
Name: Zobiya
Age: 20
PS D:\ai-assisted>
```

Error Explanation:

The screenshot displays the Visual Studio Code editor with a Python file named `lab-7.1.py`. The code defines a `Student` class with an `__init__` constructor and a `display_info` method. It takes user input for a name and an age (converted to an integer), creates a `Student` object, and calls `display_info` to print the details.

The terminal window shows the execution of the script, where the user enters 'Zobiya' for the name and '20' for the age, resulting in the output: 'Name: Zobiya' and 'Age: 20'.

The 'Task 5 Code Explanation' sidebar provides a detailed breakdown of the code:

Task 5 Code Explanation

Task 5 demonstrates **object-oriented programming (OOP)** by creating a `Student` class to store and display student information.

How it works:

- `class Student:` - Defines a class named `Student` to represent a student object.
- `def __init__(self, name, age):` - The constructor method that initializes a new `Student` object with two attributes:
 - `self.name` - Stores the student's name
 - `self.age` - Stores the student's age
- `def display_info(self):` - A method that prints the student's name and age in a formatted way.
- `name = input("Enter student name: ")` - Prompts the user to enter the student's name.
- `age = input("Enter student age: ")` - Prompts the user to enter the student's age.
- `s1 = Student(name, int(age))` - Creates a new `Student` object named `s1`, passing the name and age (converted to an integer).
- `s1.display_info()` - Calls the `display_info` method to display the student's information.

Example Output:

```
Enter student name: John
Enter student age: 20
Name: John
Age: 20
```

Summary:

- The class correctly defines the constructor using `self` to initialize object attributes **name** and **age**.
- User input is taken and the age is converted to an integer before creating the **student** object.
- The **display_info** method accesses instance attributes via `self`, ensuring the object behaves as expected.