SR UNIVERSITY

AI ASSIST CODING

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

NAME: Meer Burhan Ali Hashmi

HTNO :2503A51L44

Batch:20

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):

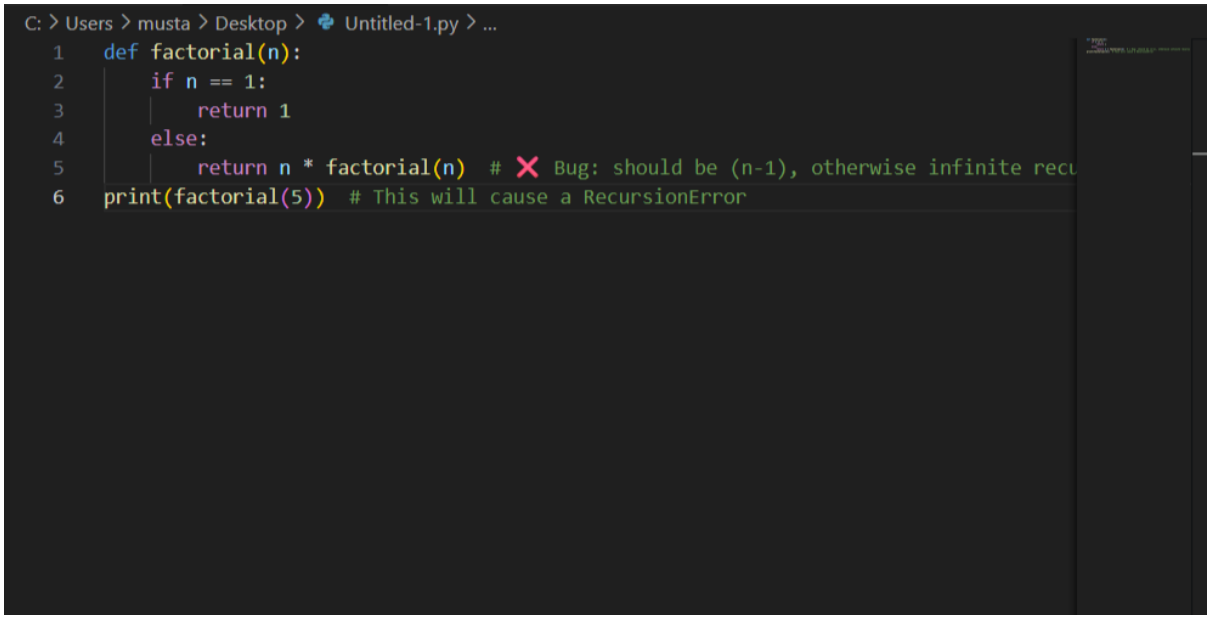
After completing this lab, students will be able to:

- 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.

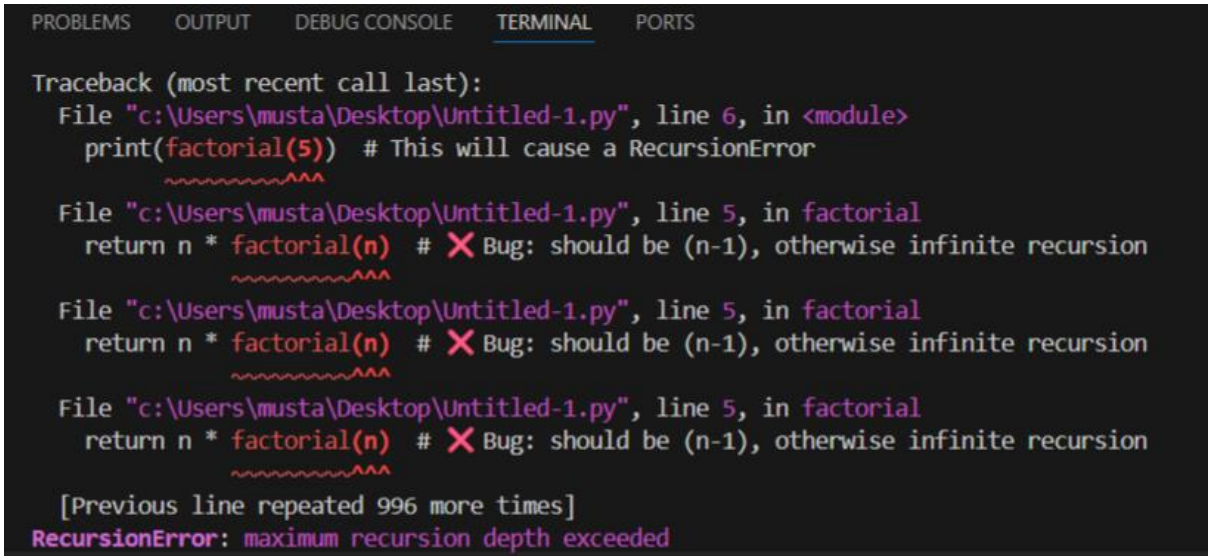
TASK#1

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

Bugged program given to the ai:

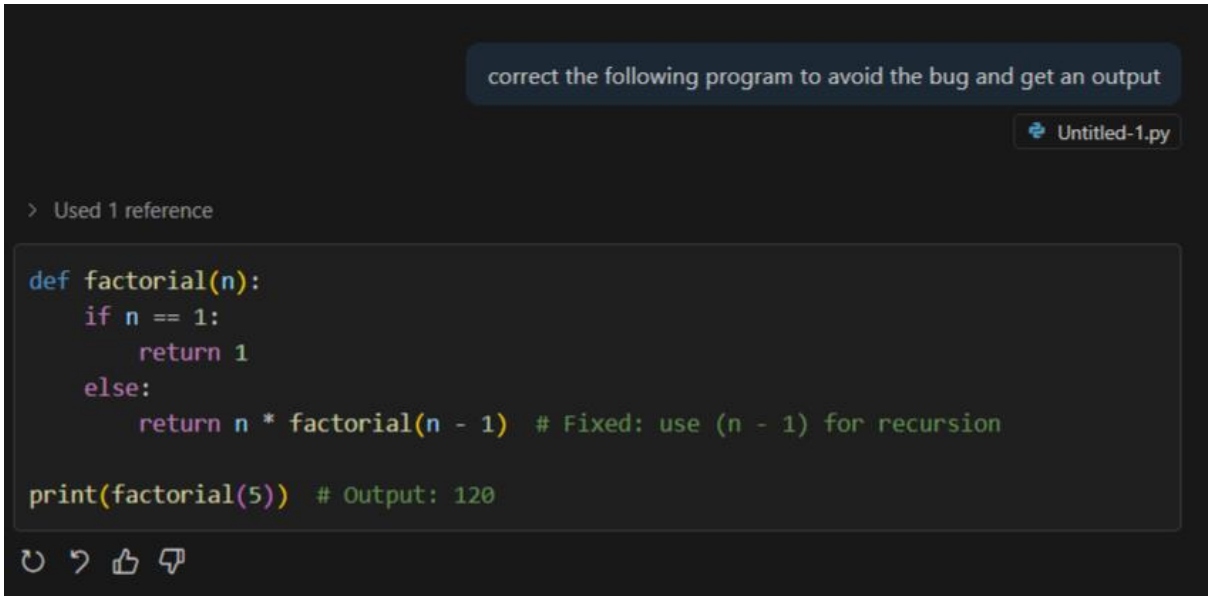


Output of bugged program:

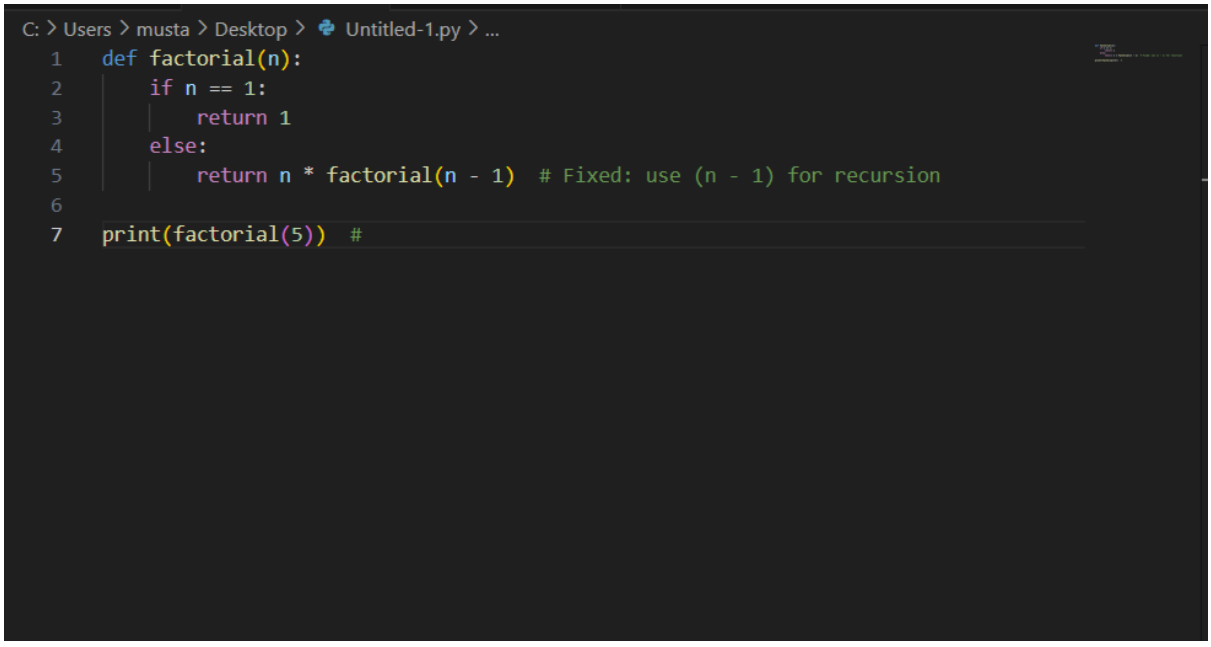


given prompt to the AI:

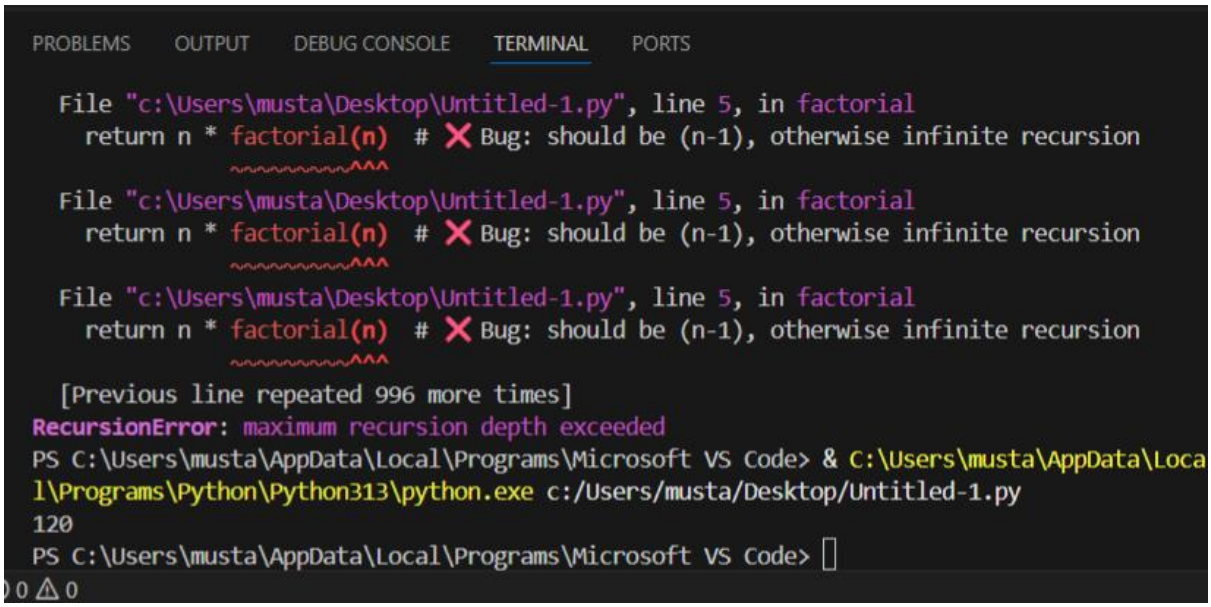
Correct the following program to avoid the bug and give an output



given program by the AI:



Final output after execution:

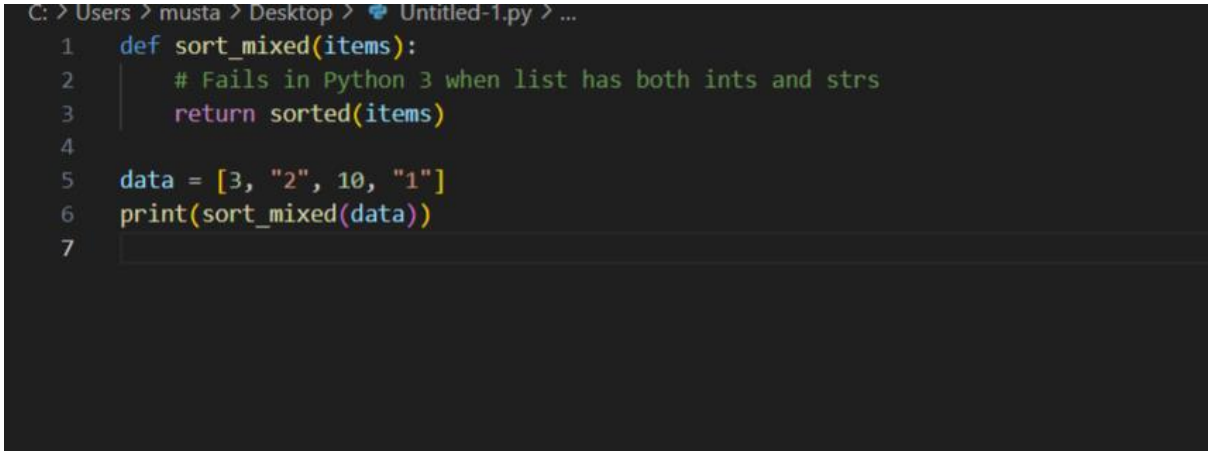


Observation:

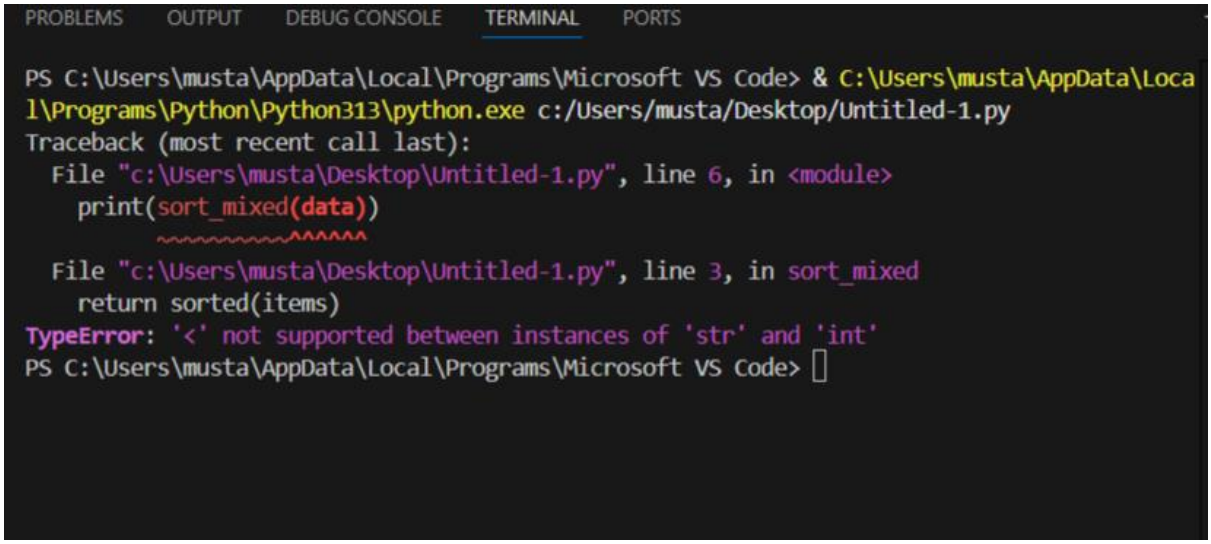
The buggy factorial function causes infinite recursion due to an incorrect or missing base case, which Copilot/Cursor AI can detect and fix.

Task #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

Bugged program given to the ai:

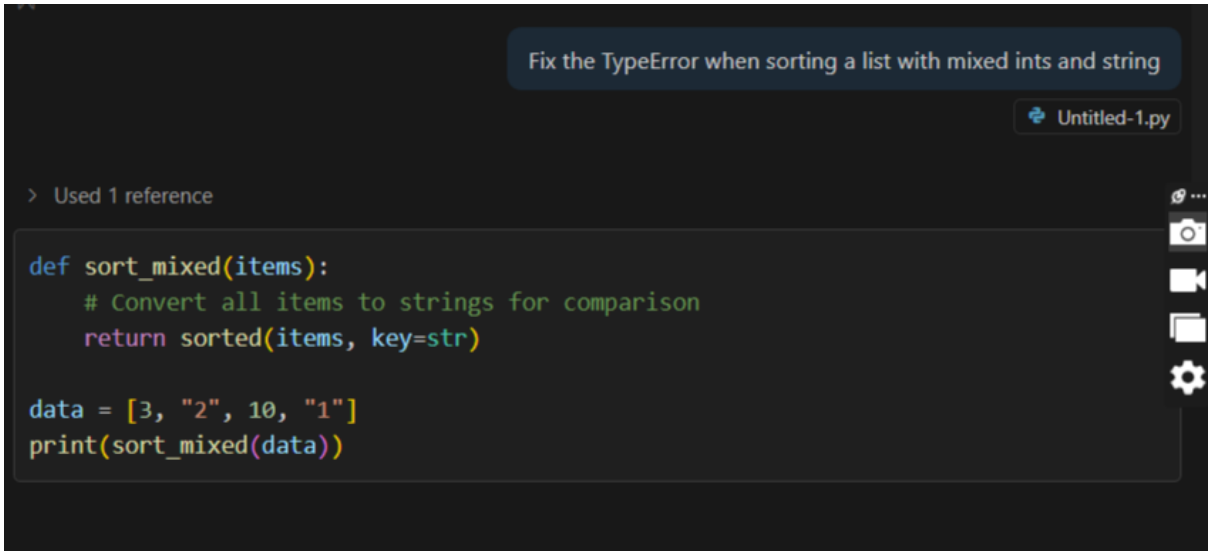


Output of the bugged program:



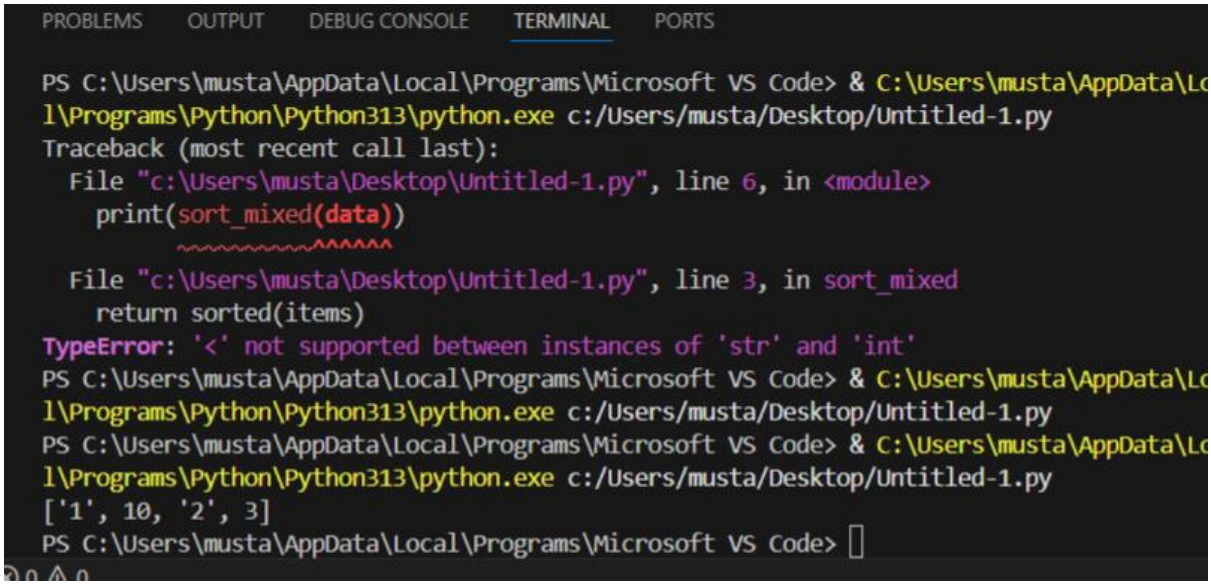
Given prompt to the ai:

fix the typeError when sorting a list with mixed ints and strings



Given program by the AI:

final output of the program:



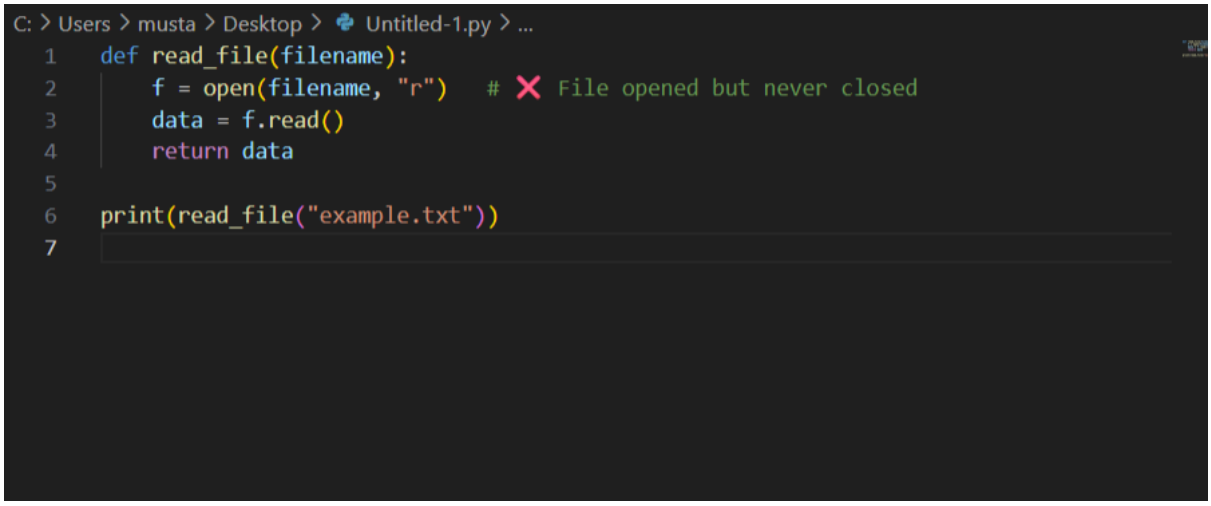
Observation:

Sorting raises TypeError because Python can't compare int and str; Copilot/Cursor can fix it by normalizing types (e.g., sorted(items, key=str) or converting all items to int) .

Task#3

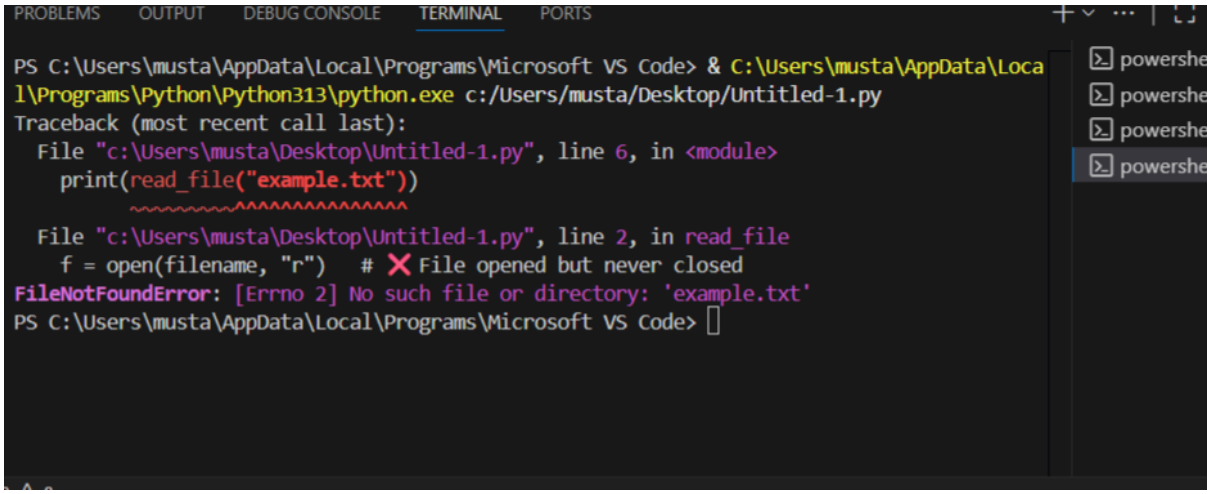
• Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block)

Given bugged program to the ai:



```
C: > Users > musta > Desktop > Untitled-1.py > ...
1   def read_file(filename):
2       f = open(filename, "r")    # ❌ File opened but never closed
3       data = f.read()
4       return data
5
6   print(read_file("example.txt"))
7
```

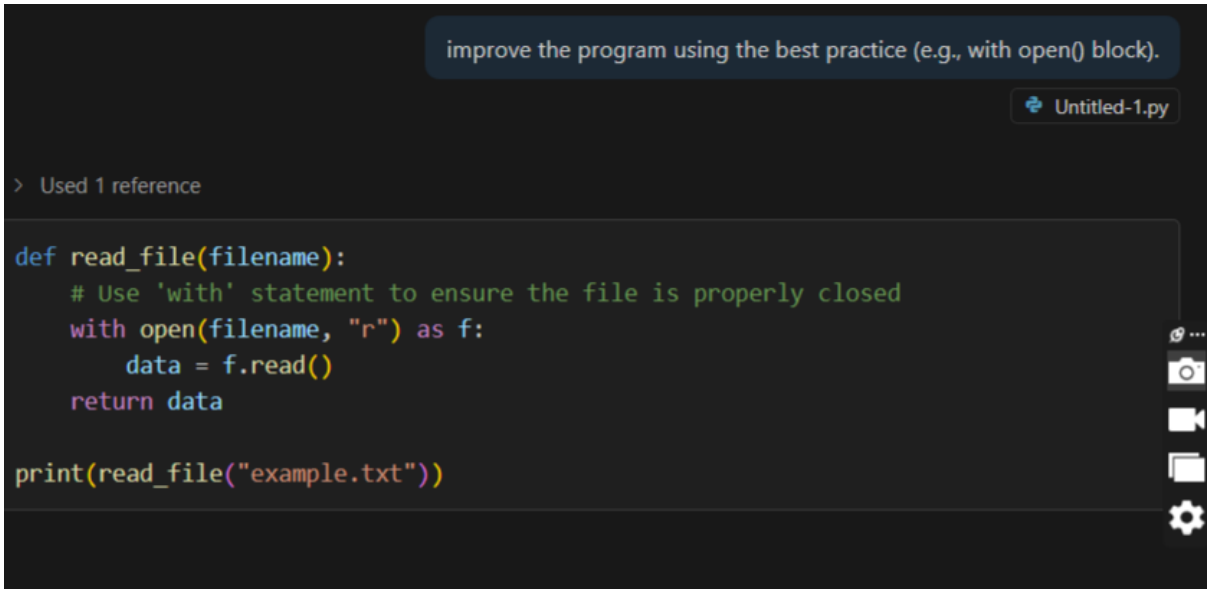
output of the bugged program:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\musta\AppData\Local\Programs\Microsoft VS Code> & C:\Users\musta\AppData\Local\Programs\Python\Python313\python.exe c:/Users/musta/Desktop/Untitled-1.py
Traceback (most recent call last):
  File "c:\Users\musta\Desktop\Untitled-1.py", line 6, in <module>
    print(read_file("example.txt"))
          ~~~~~~~~~^^^^^^^^^^^^^^^
  File "c:\Users\musta\Desktop\Untitled-1.py", line 2, in read_file
    f = open(filename, "r")    # ❌ File opened but never closed
FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'
PS C:\Users\musta\AppData\Local\Programs\Microsoft VS Code>

powershe
powershe
powershe
powershe
```

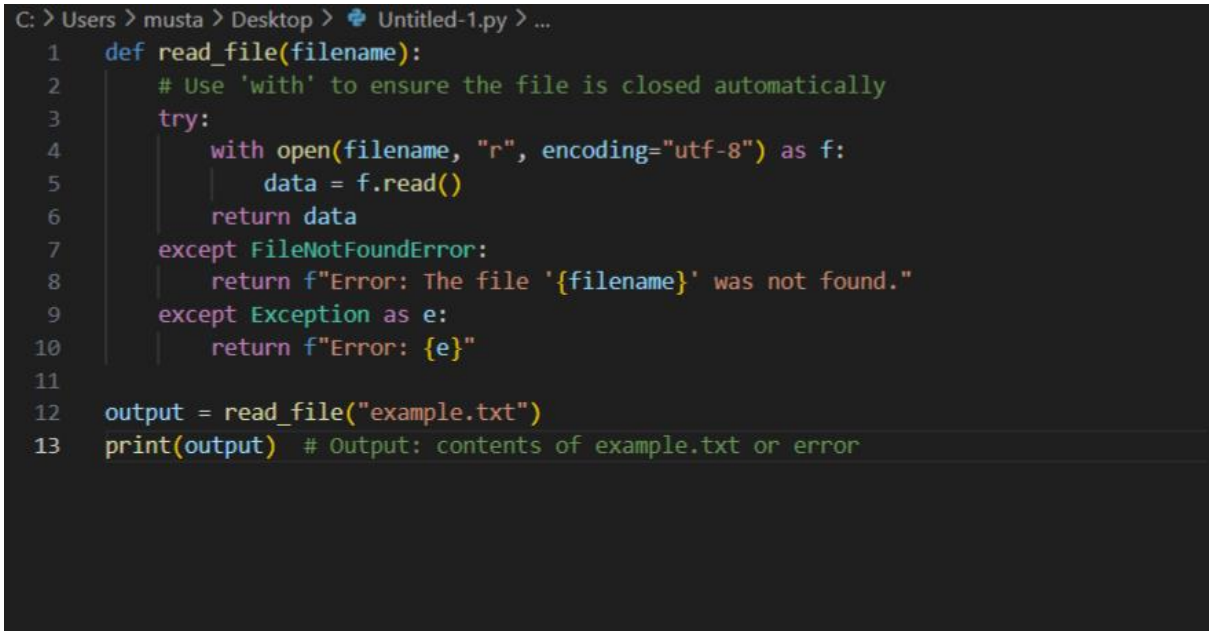
given prompt to the ai:

improve the program using the best practice (e.g., with open() block())

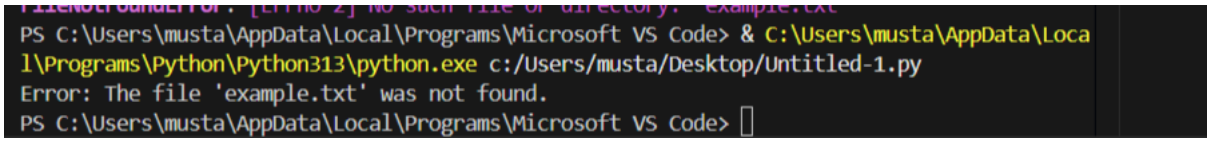


```
improve the program using the best practice (e.g., with open() block).

Untitled-1.py

> Used 1 reference

def read_file(filename):
    # Use 'with' statement to ensure the file is properly closed
    with open(filename, "r") as f:
        data = f.read()
    return data

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

Given program by the ai:



Final output of the program:



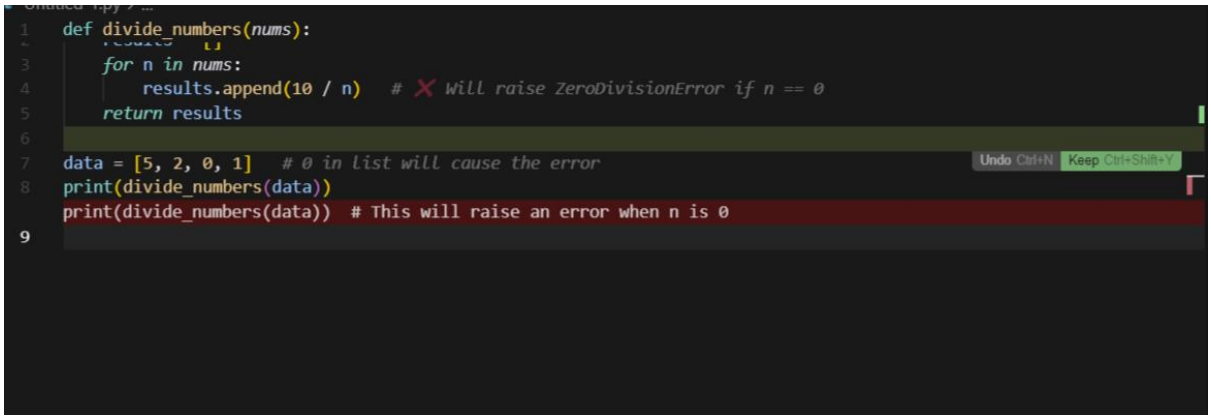
Observation:

The file is opened but never closed, which may cause resource leaks; Copilot/Cursor can fix it by using a with open() block to ensure automatic closure.

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

Given bugged program to the ai:

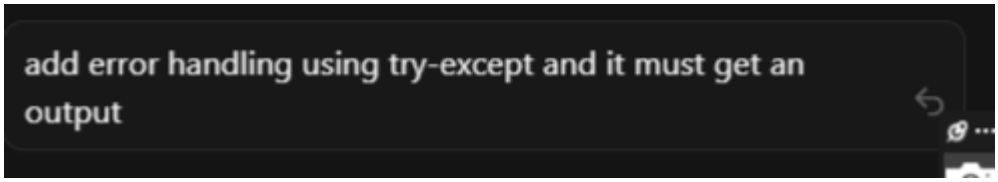


Output of the bugged program:

```
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/Untitled-1.py
Traceback (most recent call last):
  File "c:\Users\musta\.cursor\Untitled-1.py", line 8, in <module>
    print(divide_numbers(data))
          ^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\musta\.cursor\Untitled-1.py", line 4, in divide_numbers
    results.append(10 / n)   # ❌ Will raise ZeroDivisionError if n == 0
                   ~~~^~~
ZeroDivisionError: division by zero
PS C:\Users\musta\.cursor>

Ctrl+K to generate a command
```

Given prompt to the ai:

Add error handling using try-except and it must get an output



Given program by the ai:

```
Untitled-1.py > ...
1   def divide_numbers(nums):
2       results = []
3       for n in nums:
            results.append(10 / n)   # ❌ Will raise ZeroDivisionError if n == 0
4           try:
5               results.append(10 / n)
6           except ZeroDivisionError:
7               results.append(float('inf'))  # or use None/"undefined" as preferred
8           except Exception as e:
9               results.append(f"error: {e}")
10      return results                                   Undo Ctrl+N  Keep Ctrl+Shift+Y
11
12  data = [5, 2, 0, 1]   # 0 in list will cause the error
13  print(divide_numbers(data))
14  print(divide_numbers(data))  # This will raise an error when n is 0

                ^ 1 / 1 v    Undo all Ctrl+Shift+⌫   Keep all Ctrl+⏎   < 2 / 2 files >
```

Final output of the program:

```
ZeroDivisionError: division by zero
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/Untitled-1.py
[2.0, 5.0, inf, 10.0]
[2.0, 5.0, inf, 10.0]
PS C:\Users\musta\.cursor>
```

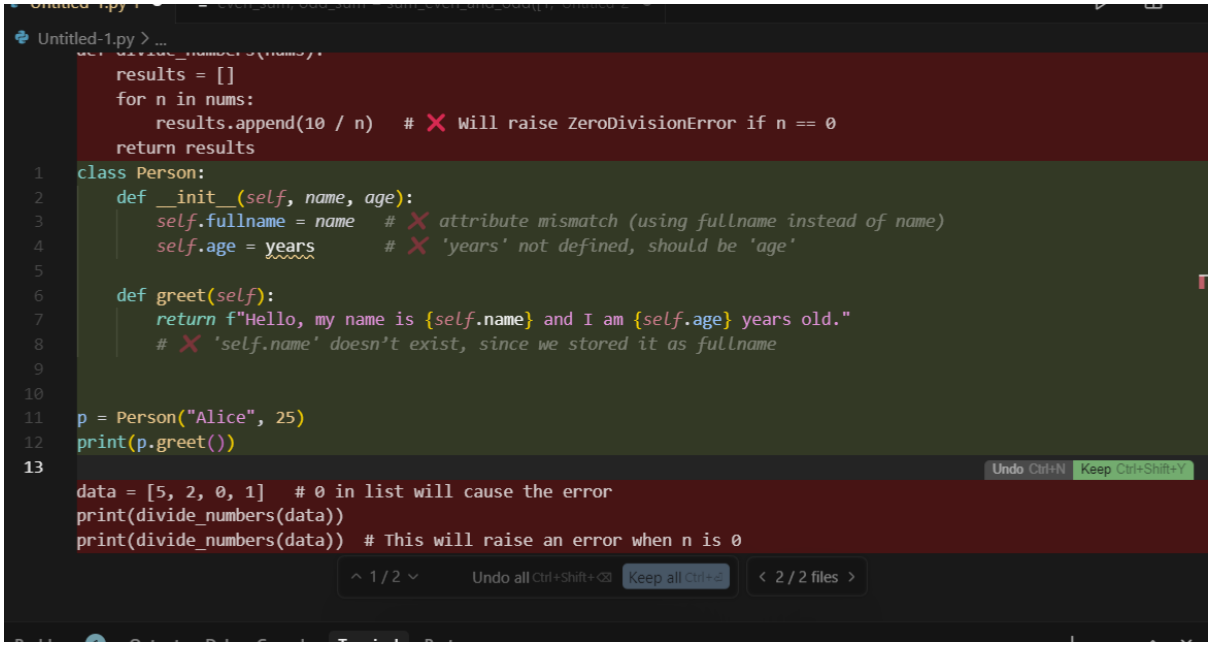
Observation:

The loop crashes with ZeroDivisionError when dividing by zero; Copilot/Cursor can fix it by wrapping the division in a try-except block and skipping or handling zero values to allow safe continuation.

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

Given bugged program to the ai:

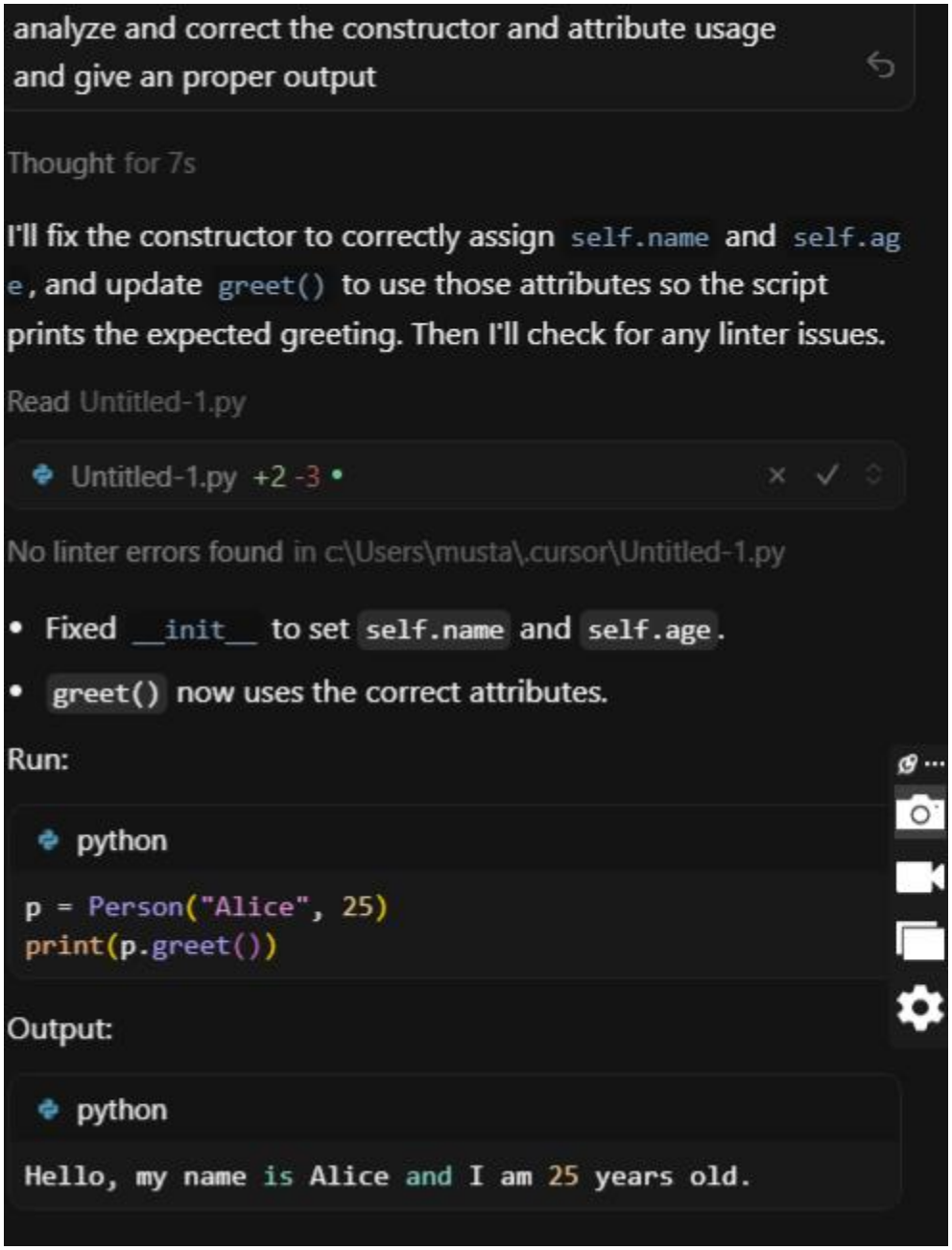


Output of the bugged program:

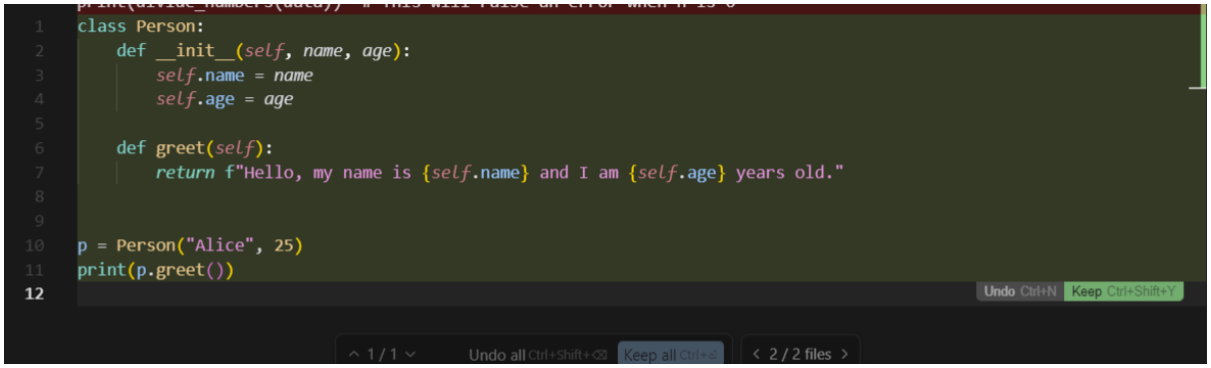
```
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/Untitled-1.py
Traceback (most recent call last):
  File "c:\Users\musta\.cursor\Untitled-1.py", line 11, in <module>
    p = Person("Alice", 25)
  File "c:\Users\musta\.cursor\Untitled-1.py", line 4, in __init__
    self.age = years       # ❌ 'years' not defined, should be 'age'
               ^^^^^
NameError: name 'years' is not defined. Did you mean: 'vars'?
PS C:\Users\musta\.cursor>
```

Given prompt to the ai:

Analyze and correct the constructor and attribute usage and give an proper output



Given program by the ai:



```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        return f"Hello, my name is {self.name} and I am {self.age} years old."


p = Person("Alice", 25)
print(p.greet())
```

Final output of the program:

```
NameError: name 'years' is not defined. Did you mean: 'vars'?
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/Untitled-1.py
Hello, my name is Alice and I am 25 years old.
PS C:\Users\musta\.cursor>
```

Observation:

The class constructor uses wrong parameter/attribute references, causing NameError or AttributeError; Copilot/Cursor can fix it by aligning __init__ parameters with proper self attributes and updating method references consistently.