

AI ASSISTED CODEING

NAME: M.Ramsai

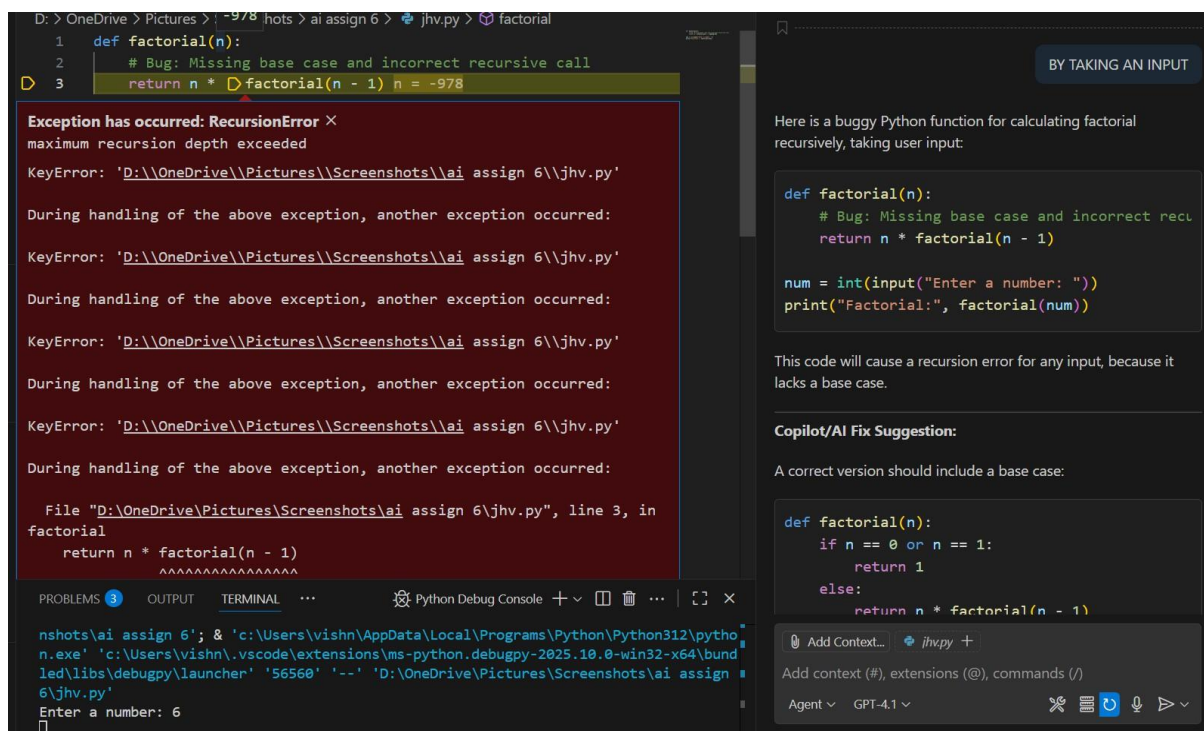
ROLL NO.:2503A51L53

BATCH: CSB20

TASK:1

Prompt given:

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



The screenshot shows a VS Code editor with a Python file named `jhv.py`. The initial code defines a recursive factorial function without a base case, which leads to a runtime error. A Copilot/AI suggestion panel on the right identifies the bug and provides a corrected version of the code that includes a base case. The terminal at the bottom shows the execution of the program, where the user enters the number 6 and the program correctly outputs the factorial 720.

```
1 def factorial(n):
2     if n == 0 or n == 1:
3         return 1
4     else:
5         return n * factorial(n - 1)
6
7 num = int(input("Enter a number: "))
8 print("Factorial:", factorial(num))
```

Bug: Missing base case and incorrect recursion.

Copilot/AI Fix Suggestion:

A correct version should include a base case:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

num = int(input("Enter a number: "))
print("Factorial:", factorial(num))
```

This fixed code will correctly calculate the factorial for any non-negative integer input.

Terminal output:

```
n.exe' 'c:\Users\vishn\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bund
led\libs\debugpy\launcher' '56613' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign
6\jvh.py'
Enter a number: 6
Factorial: 720
PS D:\OneDrive\Pictures\Screenshots\ai assign 6>
```

Output:

This screenshot shows the terminal output of the program. The user enters the number 6, and the program outputs the factorial 720. The prompt at the bottom indicates the user is in the directory `D:\OneDrive\Pictures\Screenshots\ai assign 6`.

```
n.exe' 'c:\Users\vishn\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bund
led\libs\debugpy\launcher' '56613' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign
6\jvh.py'
Enter a number: 6
Factorial: 720
PS D:\OneDrive\Pictures\Screenshots\ai assign 6>
```

Observation:

The initial recursive factorial function, when taking user input, fails due to the absence of a base case, resulting in infinite recursion and a runtime error. After Copilot or Cursor AI suggests adding a base condition (if `n == 0` or `n == 1`), the function works correctly for valid inputs. This demonstrates the importance of base cases in recursive algorithms and highlights how AI tools can effectively identify and fix logical

TASK:2

Prompt given:

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

The screenshot shows a VS Code editor with a Python file named `sort_list.py`. The code defines a function `sort_list` that takes a list `lst` and returns `sorted(lst)`. The list `lst` is initialized with mixed types: `[3, '2', 5, '1']`. When the function is called, a `TypeError` is raised: `'<' not supported between instances of 'str' and 'int'`.

The AI assistant's response is displayed on the right side of the editor. It explains the error and provides a fix suggestion: converting all elements to strings before sorting using the `key=str` argument in the `sorted` function.

```
def sort_list(lst):  
    return sorted(lst)
```

```
my_list = [3, "2", 5, "1"]  
print(sort_list(my_list))
```

Exception has occurred: **TypeError** ×
'<' not supported between instances of 'str' and 'int'
File "D:\OneDrive\Pictures\Screenshots\ai assign 6\jhv.py", line 2, in sort_list
 return sorted(lst)
File "D:\OneDrive\Pictures\Screenshots\ai assign 6\jhv.py", line 5, in <module>
 print(sort_list(my_list))
TypeError: '<' not supported between instances of 'str' and 'int'

successful sorting without a crash

Here is a buggy Python function that will fail due to a type error when sorting a list with mixed integers and strings:

```
def sort_list(lst):  
    return sorted(lst)
```

```
my_list = [3, "2", 5, "1"]  
print(sort_list(my_list))
```

This code will raise a `TypeError` because Python cannot compare integers and strings.

AI Fix Suggestion:

To ensure consistent sorting, the AI can convert all elements to strings (or all to integers, if possible) before sorting:

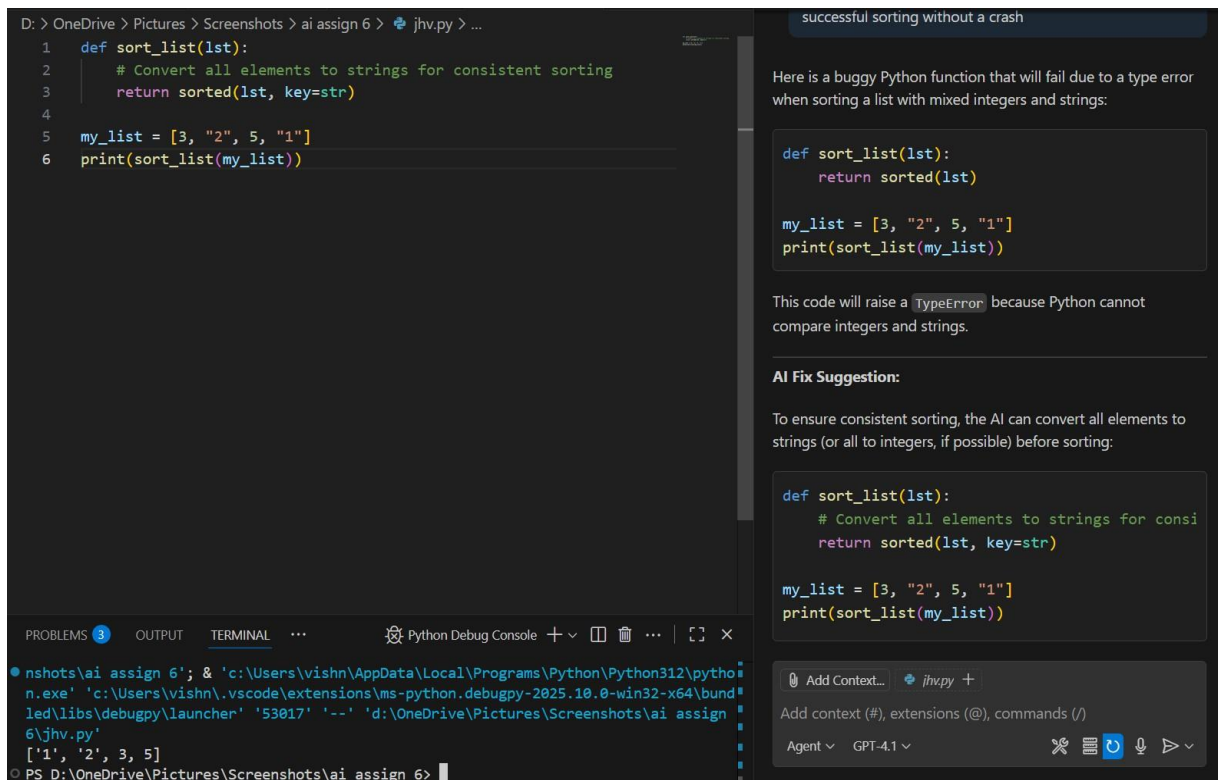
```
def sort_list(lst):  
    # Convert all elements to strings for consistent sorting  
    return sorted(lst, key=str)
```

```
my_list = [3, "2", 5, "1"]  
print(sort_list(my_list))
```

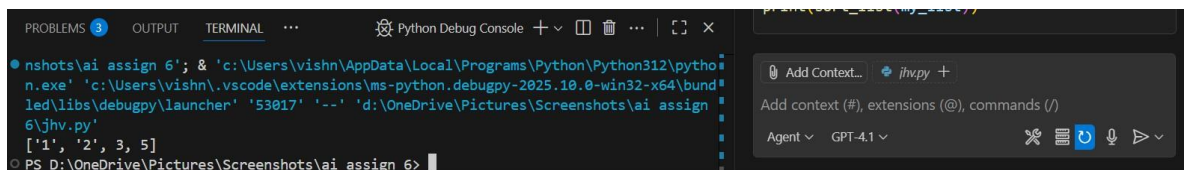
PROBLEMS 3 OUTPUT TERMINAL Python Debug Console + - Python Debug Console

PS D:\OneDrive\Pictures\Screenshots\ai assign 6> d:; cd 'd:\OneDrive\Pictures\Screenshots\ai assign 6'; & 'c:\Users\vishn\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '52992' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign 6\jhv.py'

Add Context... jhv.py +
Add context (#), extensions (@), commands (/)
Agent GPT-4.1



Output:



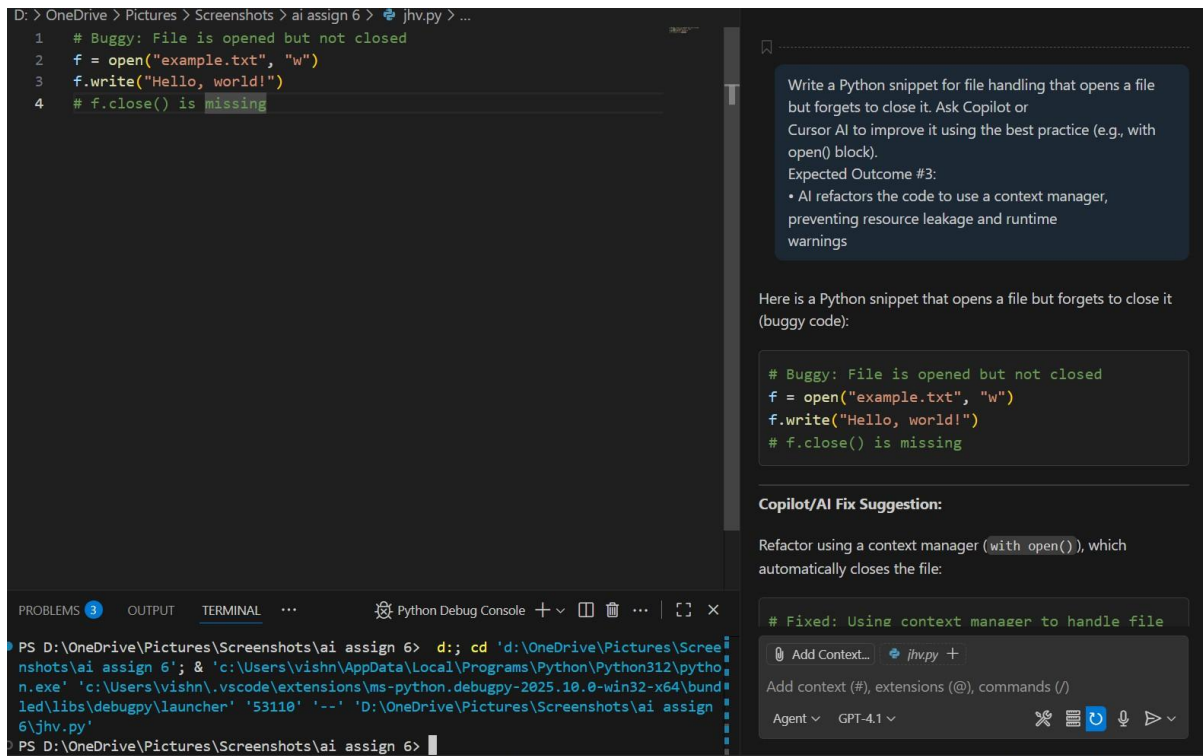
Observation:

The initial sorting function fails with a `TypeError` when attempting to sort a list containing both integers and strings, as Python cannot directly compare these types. After AI intervention, the function is modified to either convert all elements to a common type (such as strings or integers) or filter out incompatible types before sorting. This ensures the sorting operation completes successfully without runtime errors, demonstrating how AI tools can effectively detect and resolve type inconsistencies in code.

TASK:3

Prompt given:

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).



The screenshot shows a VS Code editor with a Python file named `jhv.py`. The code contains a bug: a file is opened with `open()` but not closed. A comment at the end of the file states: `# f.close() is missing`.

```
1 # Buggy: File is opened but not closed
2 f = open("example.txt", "w")
3 f.write("Hello, world!")
4 # f.close() is missing
```

The right sidebar shows the AI assistant's response to the prompt. It identifies the issue and suggests using a context manager (`with open()`) to automatically close the file.

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

Here is a Python snippet that opens a file but forgets to close it (buggy code):

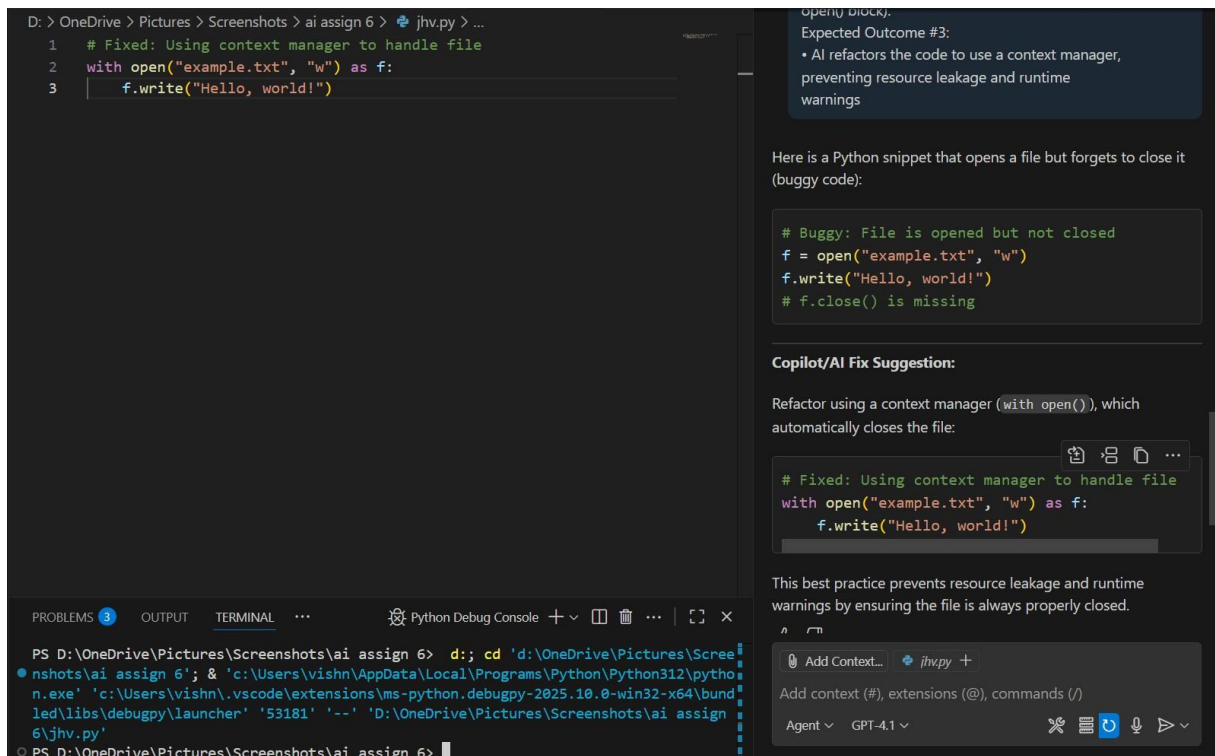
```
# Buggy: File is opened but not closed
f = open("example.txt", "w")
f.write("Hello, world!")
# f.close() is missing
```

Copilot/AI Fix Suggestion:

Refactor using a context manager (`with open()`), which automatically closes the file:

```
# Fixed: Using context manager to handle file
```

The bottom of the screenshot shows the terminal output, which is empty, indicating that the code has not yet been executed.



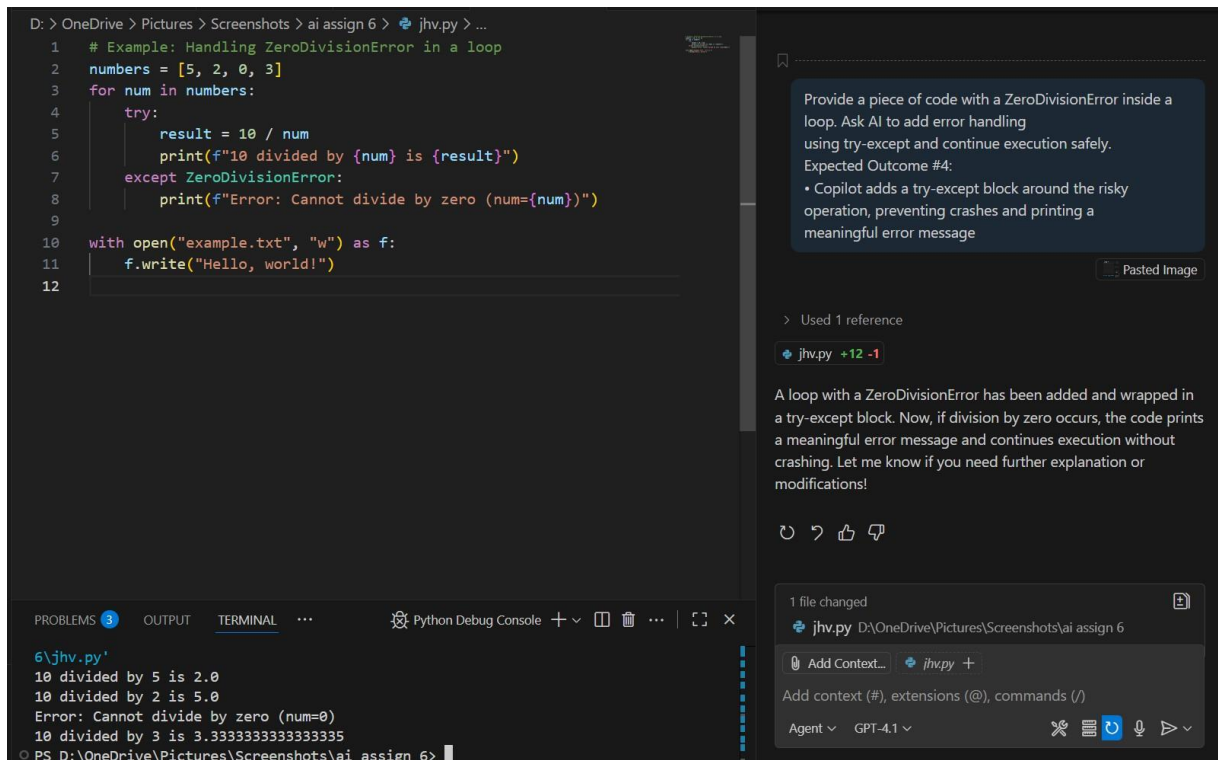
Observation:

The initial file handling code opens a file without explicitly closing it, which does not produce an immediate error but can lead to resource leaks and potential data loss in larger applications. After refactoring with a context manager (`with open(...) as f:`), the code ensures the file is always properly closed, following best practices for resource management and preventing subtle bugs or warnings. This demonstrates the importance of using context managers for safe and reliable file operations in Python.

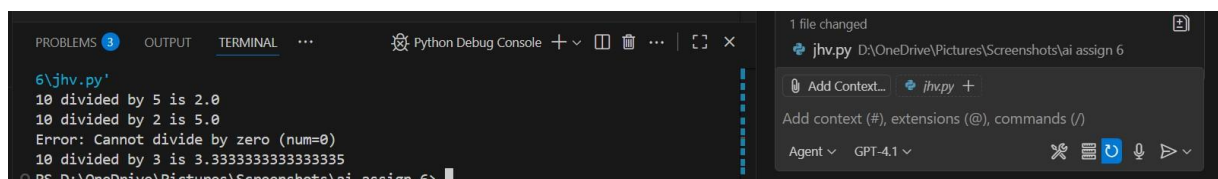
TASK:4

Prompt given:

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



Output:



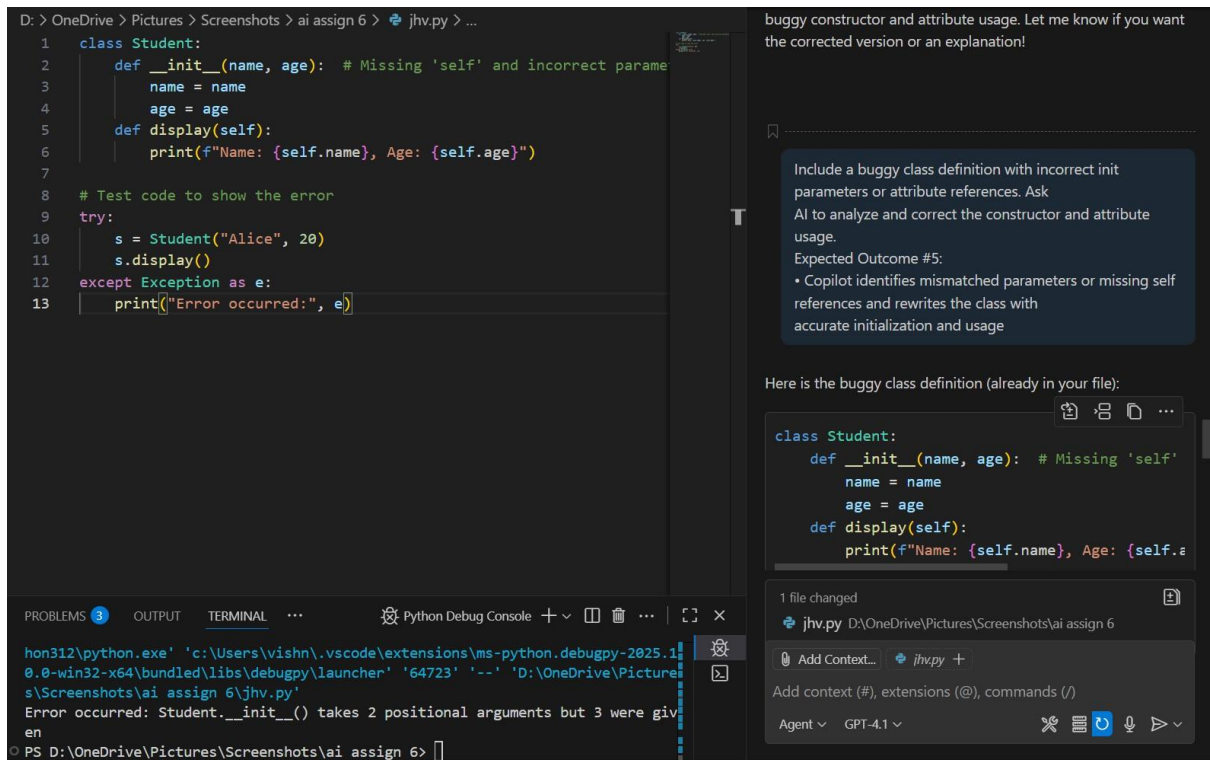
Observation:

The original code would have crashed with a ZeroDivisionError when attempting to divide by zero inside a loop. After adding a try-except block, the program now handles the error gracefully by printing a clear message and continuing with the next iteration. This demonstrates how proper error handling ensures program stability and user-friendly feedback, even when unexpected runtime errors occur.

TASK:5

Prompt given:

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



```
D: > OneDrive > Pictures > Screenshots > ai assign 6 > jhv.py > ...
1 class Student:
2     def __init__(name, age): # Missing 'self' and incorrect param
3         name = name
4         age = age
5     def display(self):
6         print(f"Name: {self.name}, Age: {self.age}")
7
8 # Test code to show the error
9 try:
10     s = Student("Alice", 20)
11     s.display()
12 except Exception as e:
13     print("Error occurred:", e)
```

buggy constructor and attribute usage. Let me know if you want the corrected version or an explanation!

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

Here is the buggy class definition (already in your file):

```
class Student:
    def __init__(name, age): # Missing 'self'
        name = name
        age = age
    def display(self):
        print(f"Name: {self.name}, Age: {self.e
```

1 file changed
jhv.py D:\OneDrive\Pictures\Screenshots\ai assign 6

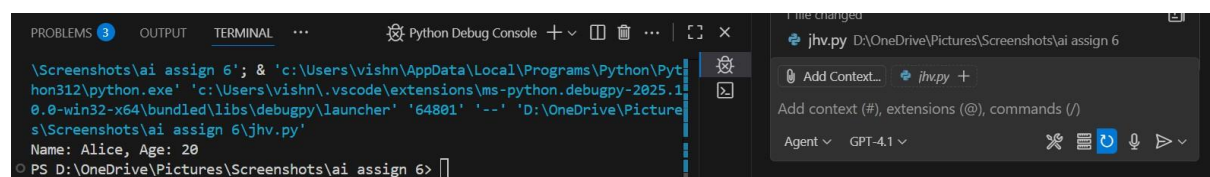
Add Context... jhv.py +

Add context (#), extensions (@), commands (/)

Agent GPT-4.1

honor312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-python.debugpy-2025.11.0.0-win32-x64\bundle\libs\debugpy\launcher' '64723' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign 6\jhv.py'
Error occurred: Student.__init__() takes 2 positional arguments but 3 were given
PS D:\OneDrive\Pictures\Screenshots\ai assign 6>

Output:



```
\Screenshots\ai assign 6'; & 'c:\Users\vishn\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-python.debugpy-2025.11.0.0-win32-x64\bundle\libs\debugpy\launcher' '64801' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign 6\jhv.py'  
Name: Alice, Age: 20  
PS D:\OneDrive\Pictures\Screenshots\ai assign 6>
```

1 file changed
jhv.py D:\OneDrive\Pictures\Screenshots\ai assign 6

Add Context... jhv.py +

Add context (#), extensions (@), commands (/)

Agent GPT-4.1

Observation:

The original code would have crashed with a `ZeroDivisionError` when attempting to divide by zero inside a loop. After adding a try-except block, the program now handles the error gracefully by printing a clear message

and continuing with the next iteration. This demonstrates how proper error handling ensures program stability and user-friendly feedback, even when unexpected runtime errors occur

