

AI ASSISTED CODEING

NAME: V . Vishnu Vardhan

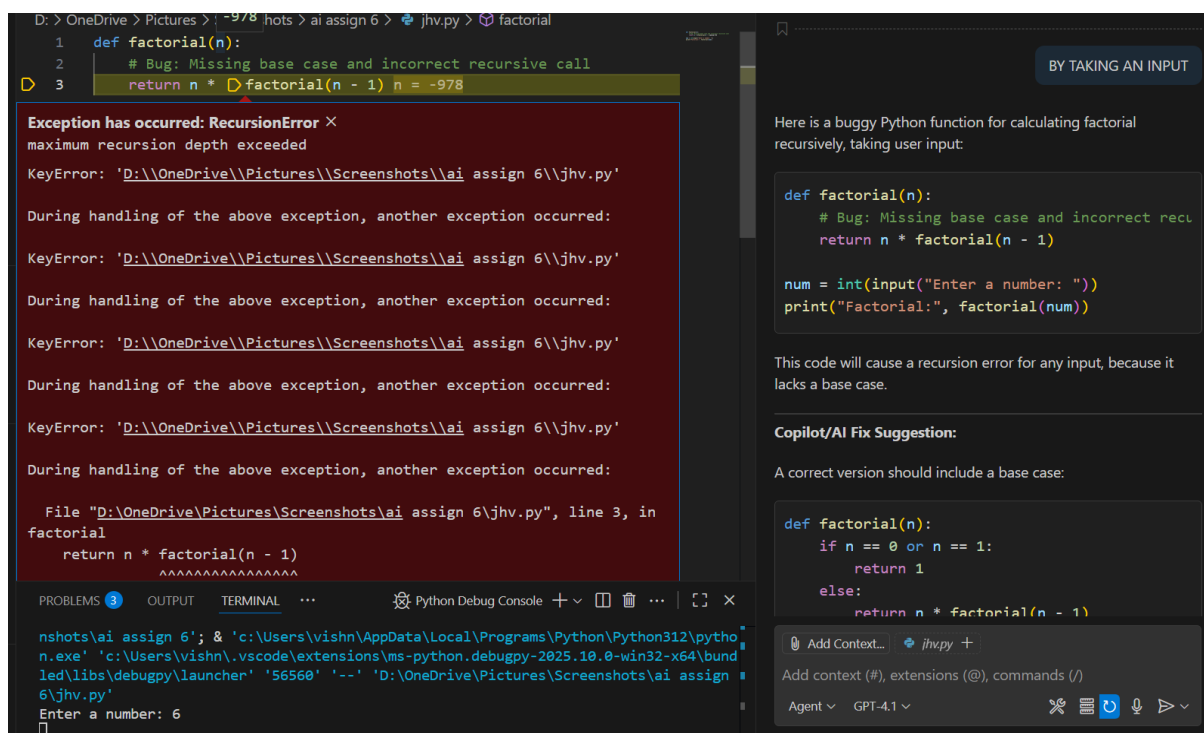
ROLL NO.:2503A51L26

BATCH: CSB19

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



TASK:2

Prompt given:

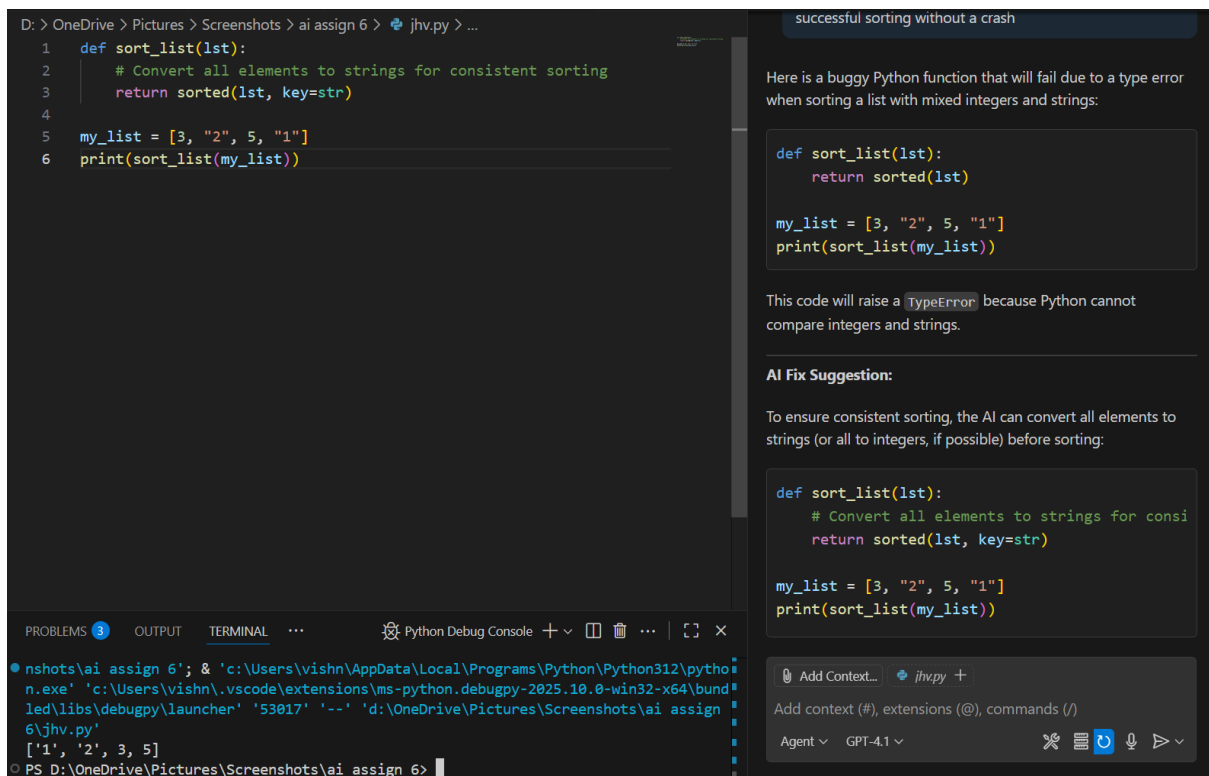
Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed integers and strings). Prompt AI to detect the issue and fix the code for consistent sorting

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'`.

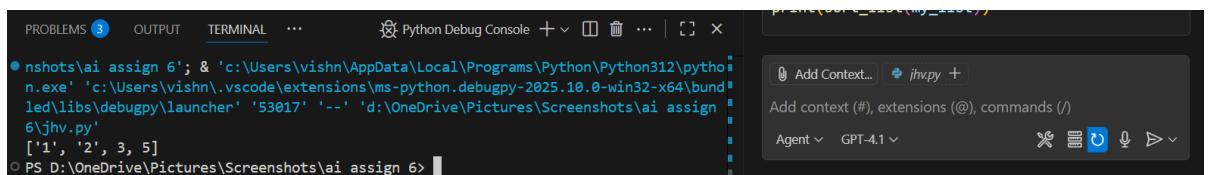
The AI assistant's response is displayed on the right side of the editor. It explains the error and provides a fix suggestion: converting all elements to strings before sorting. The fixed code is shown below:

```
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))
```

The AI assistant also provides a terminal command to run the fixed code: `python sort_list.py`.



Output:



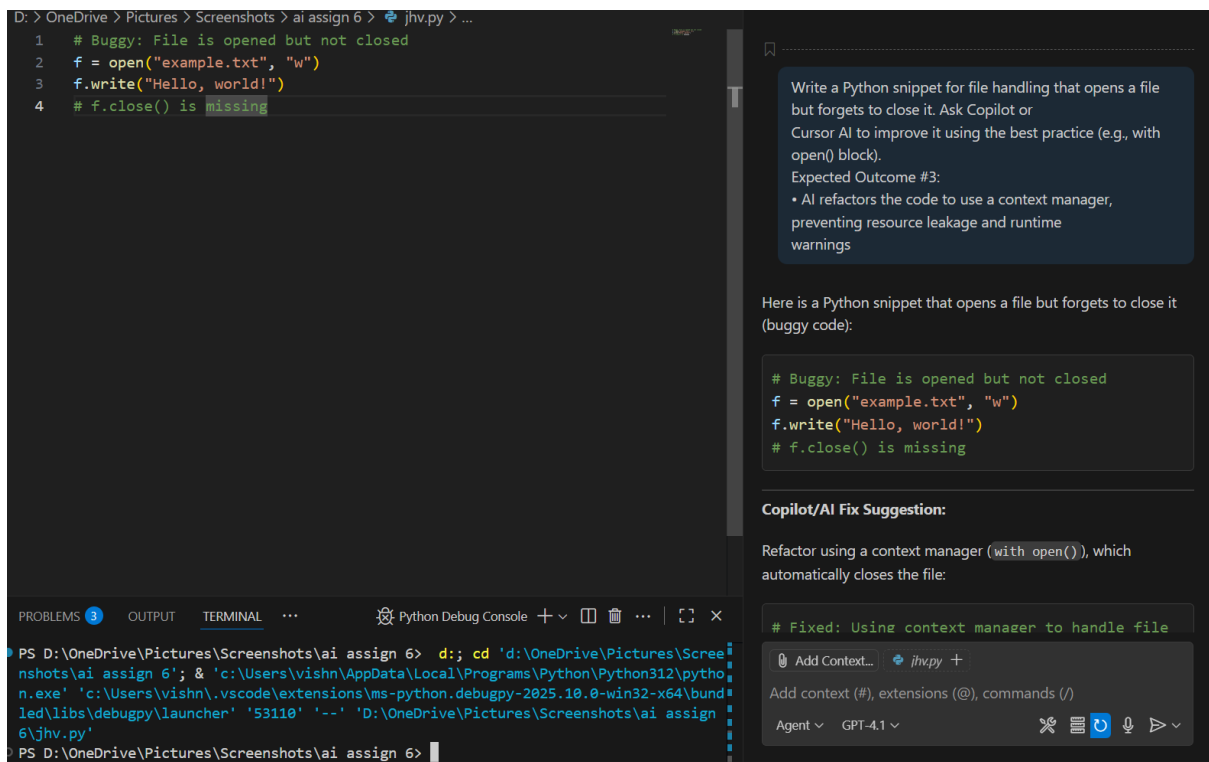
Observation:

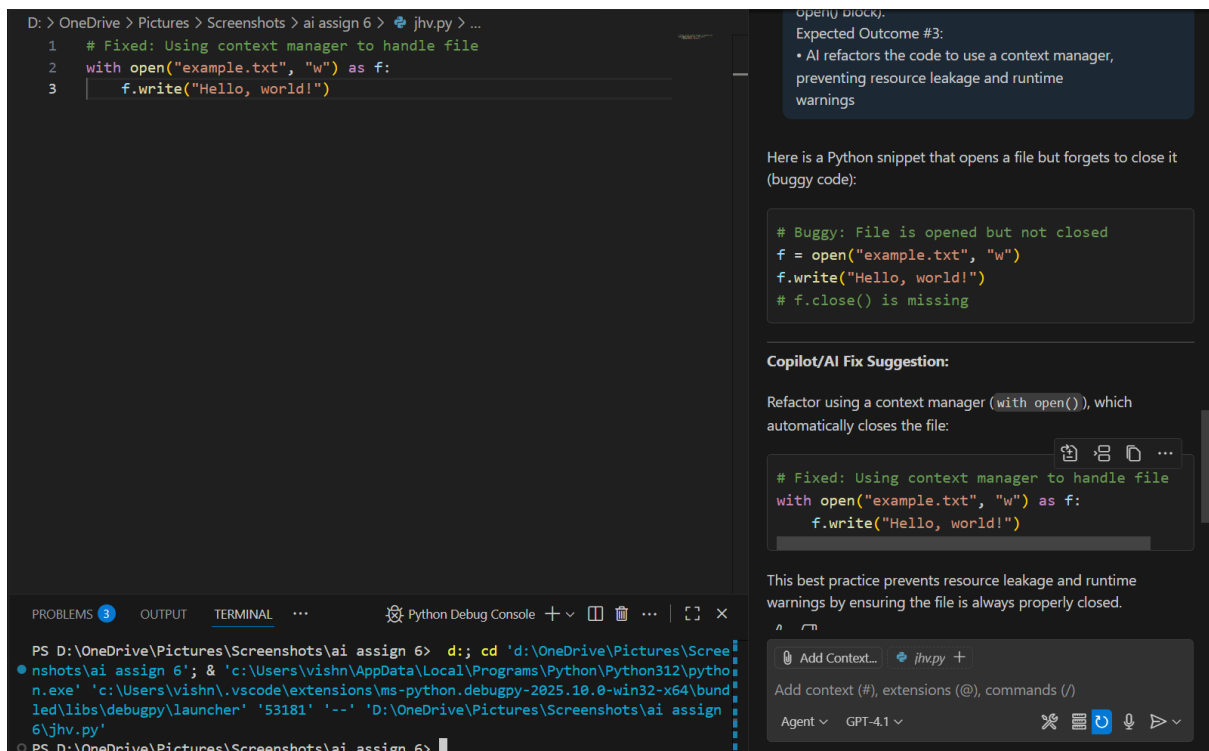
The initial sorting function fails with a `TypeError` when attempting to sort a list containing both integers and strings, as Python cannot directly compare these types. After AI intervention, the function is modified to either convert all elements to a common type (such as strings or integers) or filter 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).





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.

The screenshot shows a VS Code editor with a Python file named `jhv.py`. The code is as follows:

```
D: > OneDrive > Pictures > Screenshots > ai assign 6 > jhv.py > ...  
1 # Example: Handling ZeroDivisionError in a loop  
2 numbers = [5, 2, 0, 3]  
3 for num in numbers:  
4     try:  
5         result = 10 / num  
6         print(f"10 divided by {num} is {result}")  
7     except ZeroDivisionError:  
8         print(f"Error: Cannot divide by zero (num={num})")  
9  
10 with open("example.txt", "w") as f:  
11     f.write("Hello, world!")  
12
```

The terminal output at the bottom shows the execution results:

```
6\jvh.py'  
10 divided by 5 is 2.0  
10 divided by 2 is 5.0  
Error: Cannot divide by zero (num=0)  
10 divided by 3 is 3.3333333333333335  
PS D:\OneDrive\Pictures\Screenshots\ai assign 6>
```

Output:

A screenshot of a code editor interface. The main editor area shows a Python file named 'jvhv.py' with the following code:

```
10 divided by 5 is 2.0
10 divided by 2 is 5.0
Error: Cannot divide by zero (num=0)
10 divided by 3 is 3.3333333333333335
10 divided by 1 is 10.0
```

The terminal window at the bottom shows the execution output, which matches the code. On the right side, there is a sidebar with a file explorer showing '1 file changed' and a list of files including 'jvhv.py'. Below the file explorer, there is a section for 'Add context (#), extensions (@), commands (/)' and a dropdown menu for 'Agent' set to 'GPT-4.1'.

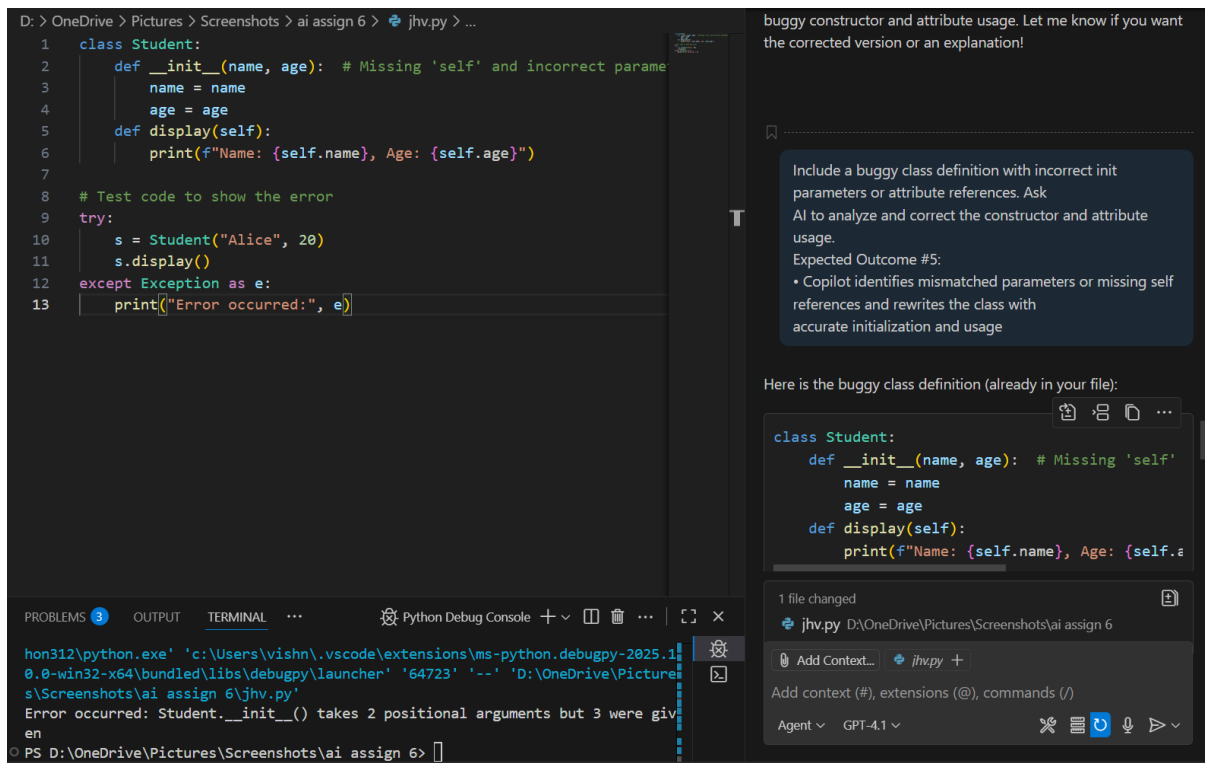
Observation:

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.

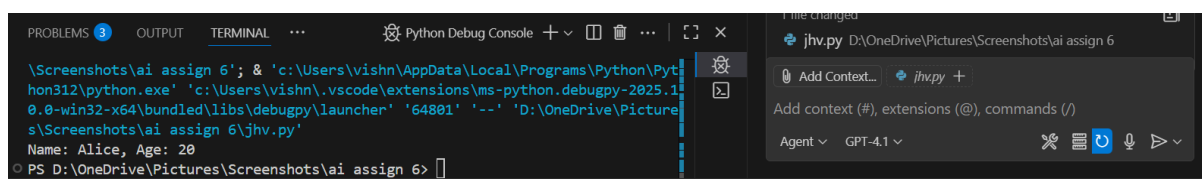
TASK:5

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.



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

