

## AI ASSISTED CODEING

NAME: B. RAJA GOPAL

ROLL NO.: 2503A51L01

BATCH: CSB19

mnbvcxzasdfghjklpoiuytrewq **TASK:**

**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 code defines a `factorial` function with a bug: it lacks a base case and has an incorrect recursive call. The function is called with `n = -978`, which causes a `RecursionError` (maximum recursion depth exceeded). The error message is displayed in the Python Debug Console. On the right side, the Copilot/AI Fix Suggestion panel shows the original buggy code and a corrected version that includes a base case (`if n == 0 or n == 1: return 1`). The terminal at the bottom shows the command to run the script and the user input `6`.

```
D: > OneDrive > Pictures > | -978 |hots > ai assign 6 > jhv.py > factorial
1 def factorial(n):
2     # Bug: Missing base case and incorrect recursive call
3     return n * factorial(n - 1) n = -978

Exception has occurred: RecursionError X
maximum recursion depth exceeded
KeyError: 'D:\\OneDrive\\Pictures\\Screenshots\\ai assign 6\\jhv.py'
During handling of the above exception, another exception occurred:
KeyError: 'D:\\OneDrive\\Pictures\\Screenshots\\ai assign 6\\jhv.py'
During handling of the above exception, another exception occurred:
KeyError: 'D:\\OneDrive\\Pictures\\Screenshots\\ai assign 6\\jhv.py'
During handling of the above exception, another exception occurred:
KeyError: 'D:\\OneDrive\\Pictures\\Screenshots\\ai assign 6\\jhv.py'
During handling of the above exception, another exception occurred:
KeyError: 'D:\\OneDrive\\Pictures\\Screenshots\\ai assign 6\\jhv.py'
During handling of the above exception, another exception occurred:
File "D:\\OneDrive\\Pictures\\Screenshots\\ai assign 6\\jhv.py", line 3, in factorial
    return n * factorial(n - 1)
    ^^^^^^^^^^^^^^^^^^^^^^^^^

PROBLEMS 3 OUTPUT TERMINAL ... Python Debug Console + - [ ] X
nshots\\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\\bundled\\libs\\debugpy\\launcher' '56560' '--' 'D:\\OneDrive\\Pictures\\Screenshots\\ai assign 6\\jhv.py'
Enter a number: 6
```

BY TAKING AN INPUT

Here is a buggy Python function for calculating factorial recursively, taking user input:

```
def factorial(n):
    # Bug: Missing base case and incorrect recursive call
    return n * factorial(n - 1)

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

This code will cause a recursion error for any input, because it lacks a base case.

**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)
```

Add Context... jhv.py +

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

Agent GPT-4.1

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 is highlighted as a bug. The code prompts the user to enter a number and prints the factorial. The terminal shows the program running and calculating the factorial of 6 as 720. A Copilot/AI Fix Suggestion panel on the right provides a corrected version of the code that includes a base case for `n == 0 or n == 1`.

```
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 logic.

**Copilot/AI Fix Suggestion:**

```
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:

The screenshot shows the terminal output of the program. It prompts the user to enter a number, and the user enters 6. The program calculates the factorial of 6 and outputs 720.

```
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:

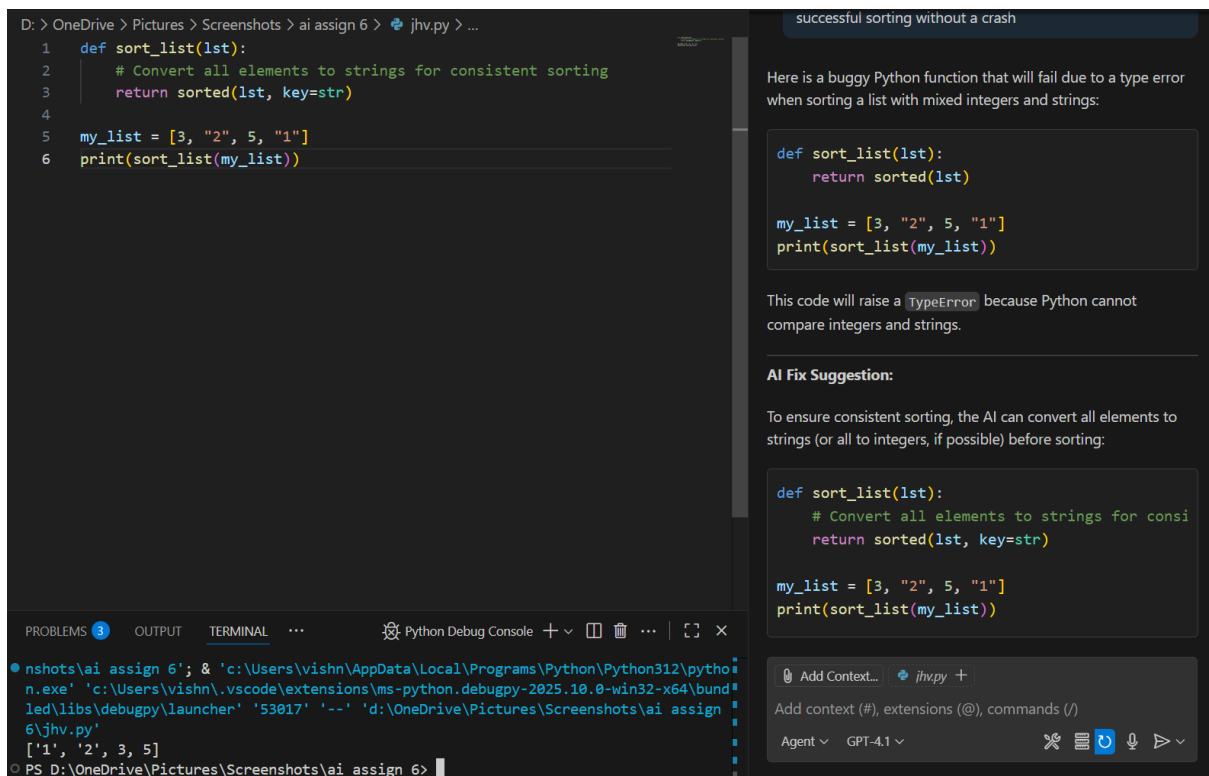
Provide a list sorting function thaailed due to a type error (e.g., sorting list with mixed integers and strings). Prompt AI to detec the issue andx fix the code for consistent soting

The image shows a VS Code editor window with a Python file named `sort_list.py`. The code defines a function `sort_list(lst)` that returns `sorted(lst)`. Below the function, there is a list `my_list = [3, "2", 5, "1"]` and a call to `print(sort_list(my_list))`. A red error message is displayed, indicating a `TypeError` because Python cannot compare integers and strings. The error message states: "Exception has occurred: TypeError", "'<' not supported between instances of 'str' and 'int'", and "File 'D:\OneDrive\Pictures\Screenshots\ai assign 6\jhv.py', line 2, in sort\_list".

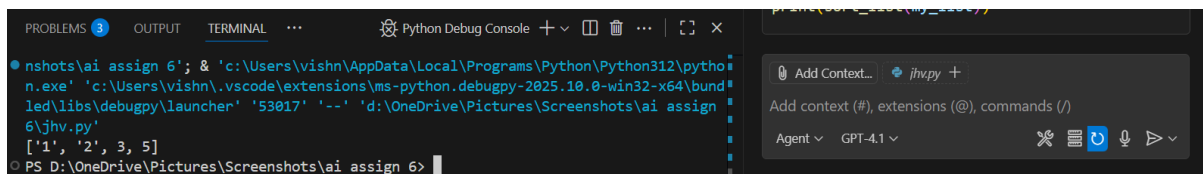
Below the error message, the code is shown again, highlighting the list `my_list` and the function call. The list contains mixed types: integers (3, 5) and strings ("2", "1").

On the right side of the editor, there is a panel titled "successful sorting without a crash". It contains a text block explaining the issue: "Here is a buggy Python function that will fail due to a type error when sorting a list with mixed integers and strings:". Below this, the original buggy code is shown. Then, an "AI Fix Suggestion:" is provided, which suggests converting all elements to strings before sorting: `return sorted(lst, key=str)`. The fixed code is shown below the suggestion, using `sorted(lst, key=str)` to sort the list.

At the bottom of the editor, there is a terminal window showing the command prompt output. The command `python sort_list.py` is executed, and the output is displayed. The terminal also shows the file path and the command used to run the script.



## Output:



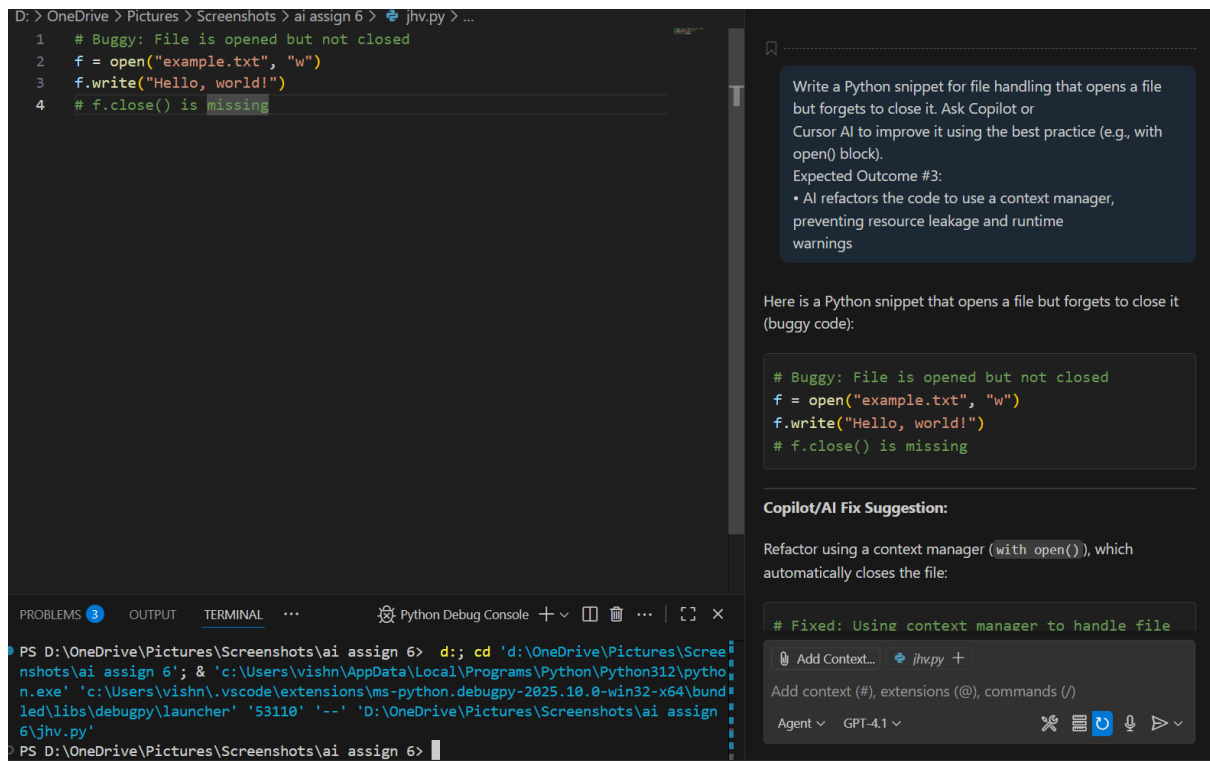
## Observation:

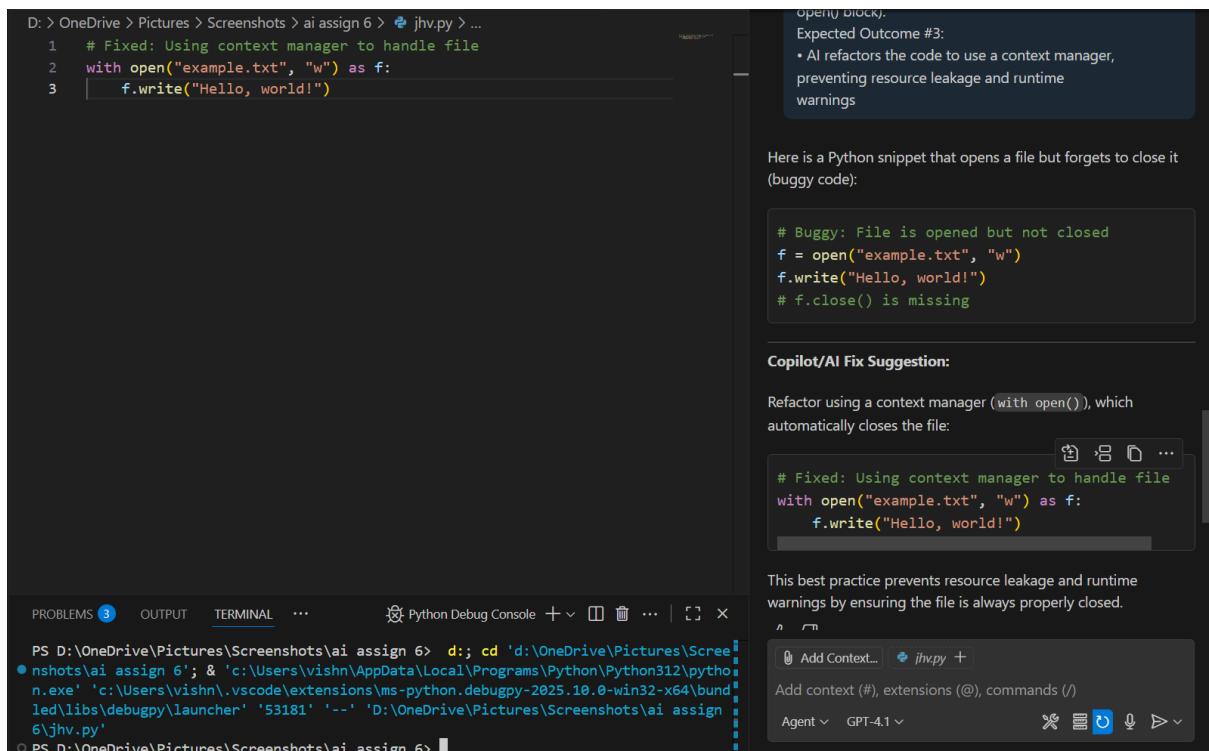
The initial sorting function fails with a `TypeError` when attempting to sort a list containing both integers and strings, as Python cannot directly compare these types. After AI intervention, the function is modified to either convert all elements to a common type (such as strings or integers) or filter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

```
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 shows the execution of the code:

```
6\jnv.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>
```

The right sidebar shows a chat window with the prompt: "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". The response states: "A loop with a ZeroDivisionError has been added and wrapped in a try-except block. Now, if division by zero occurs, the code prints a meaningful error message and continues execution without crashing. Let me know if you need further explanation or modifications!".

### Output:

The screenshot shows the terminal output of the code execution:

```
6\jnv.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>
```

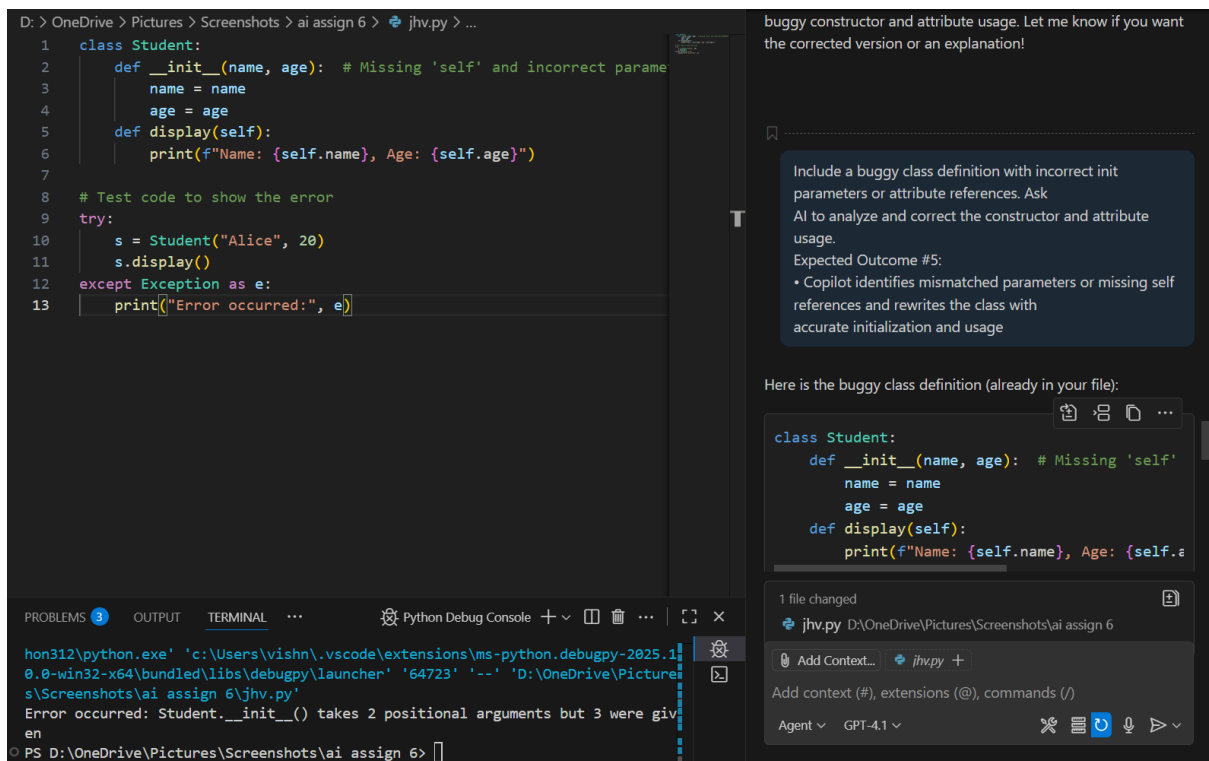
### Observation:

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. AskAI 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(f"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.s
```

1 file changed

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

Add Context... jhv.py +

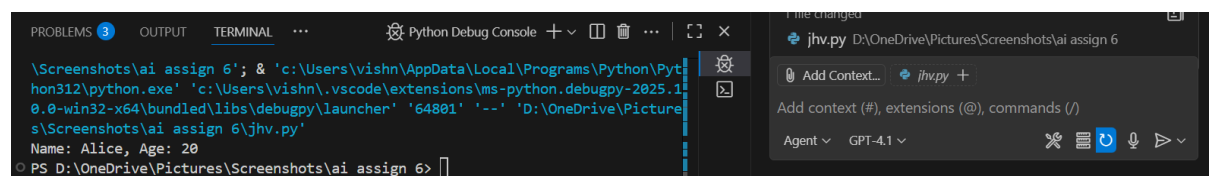
Add context (#), extensions (@), commands (/)

Agent GPT-4.1

Python Debug Console

```
hon312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-python.debugpy-2025.1
0.0-win32-x64\bundled\libs\debugpy\launcher' '64723' '--' 'D:\OneDrive\Picture
s\Screenshots\ai assign 6\jhv.py'
Error occurred: Student.__init__() takes 2 positional arguments but 3 were giv
en
PS D:\OneDrive\Pictures\Screenshots\ai assign 6>
```

## Output:



```
\Screenshots\ai assign 6'; & 'c:\Users\vishn\AppData\Local\Programs\Python\Pyt
hon312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-python.debugpy-2025.1
0.0-win32-x64\bundled\libs\debugpy\launcher' '64801' '--' 'D:\OneDrive\Picture
s\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

