

# Assignment 7.4

Name: N.Akhil

Ht.no: 2303A51065

Bt.no : 16

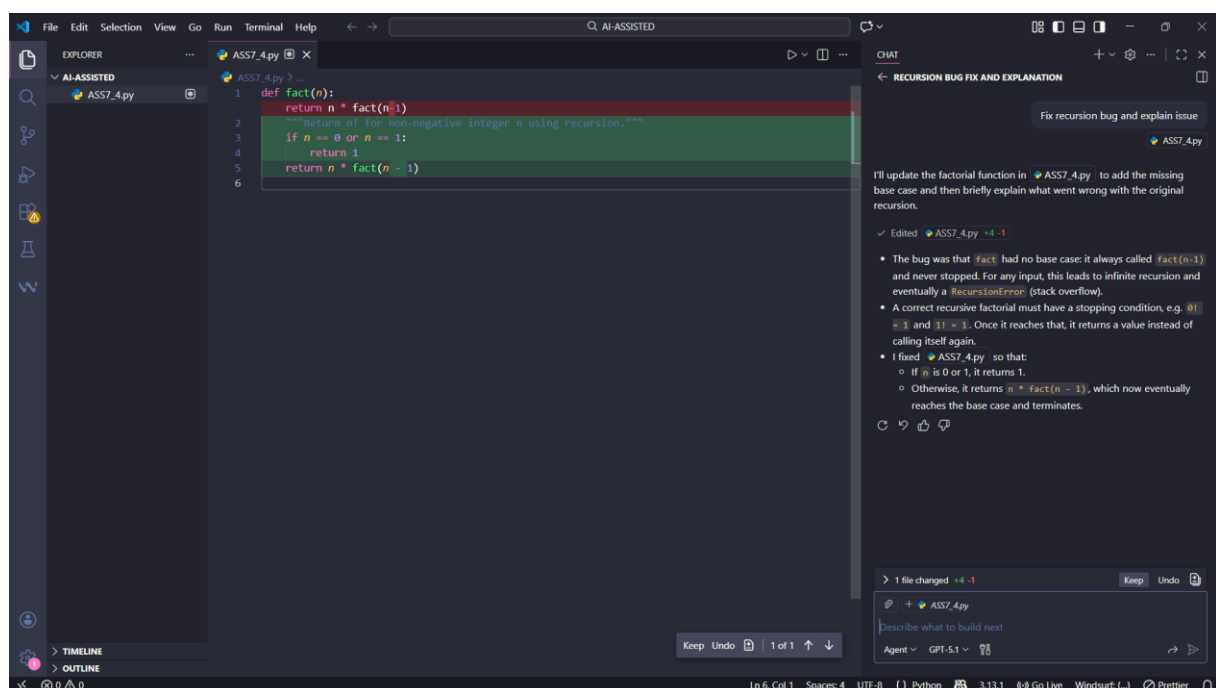
## Task 1: Debugging a Recursive Calculation Module

### 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

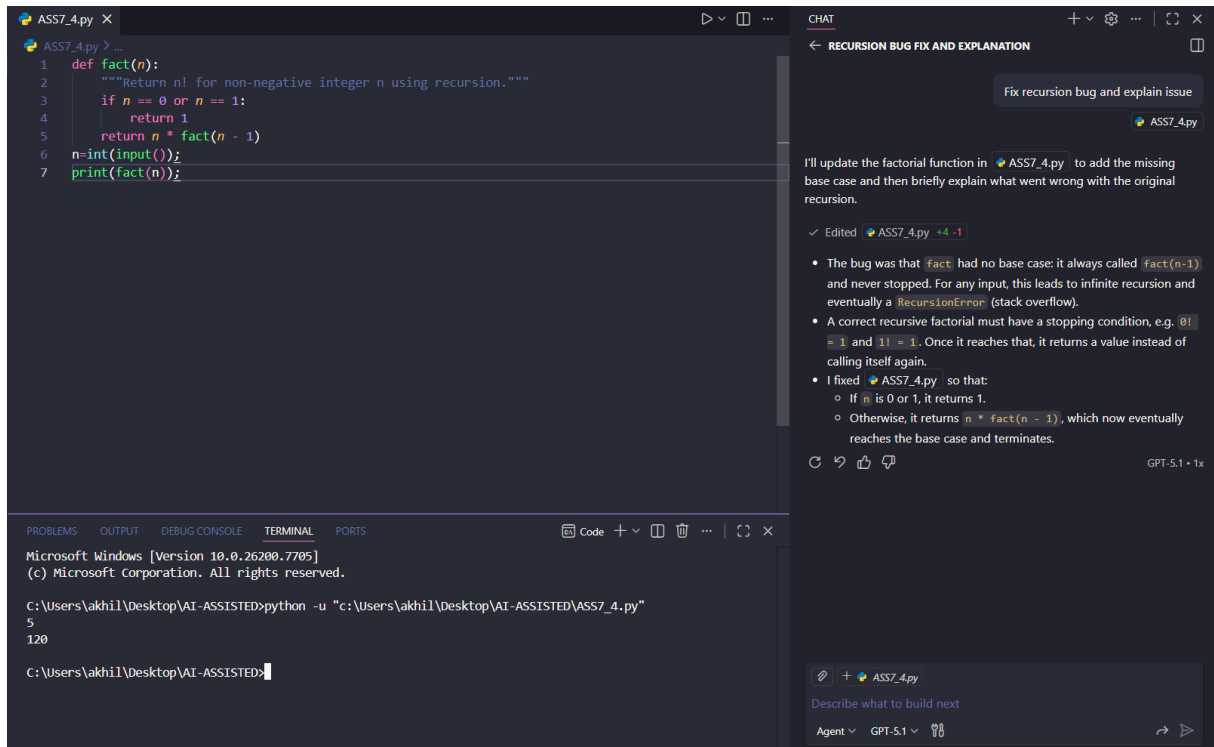
### Task Description

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).



- The bug was that `fact` had no base case: it always called `fact(n-1)` and never stopped. For any input, this leads to infinite recursion and eventually a `RecursionError` (stack overflow).

- A correct recursive factorial must have a stopping condition, e.g.  $0! = 1$  and  $1! = 1$ . Once it reaches that, it returns a value instead of calling itself again.



## Task 2: Fixing Data Type Errors in a Sorting Utility

### Scenario

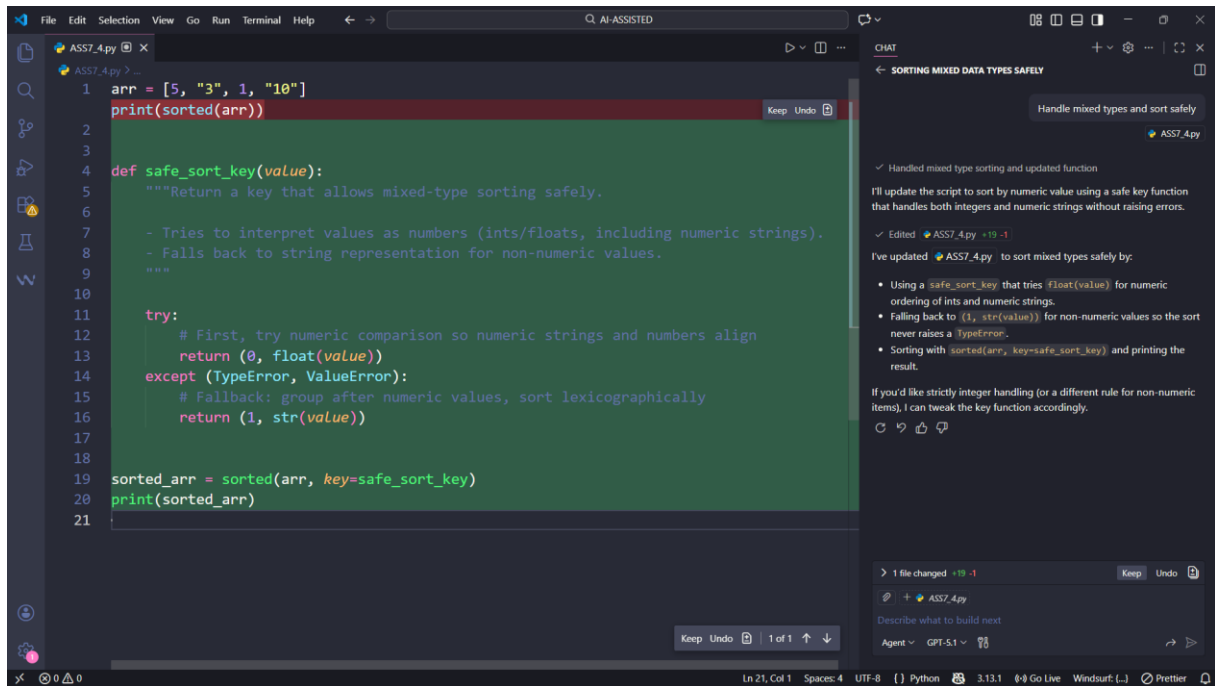
You are developing a data processing script that sorts user input values.

The program crashes when users enter mixed data types.

### Task Description

You are provided with a list-sorting function that fails due to a

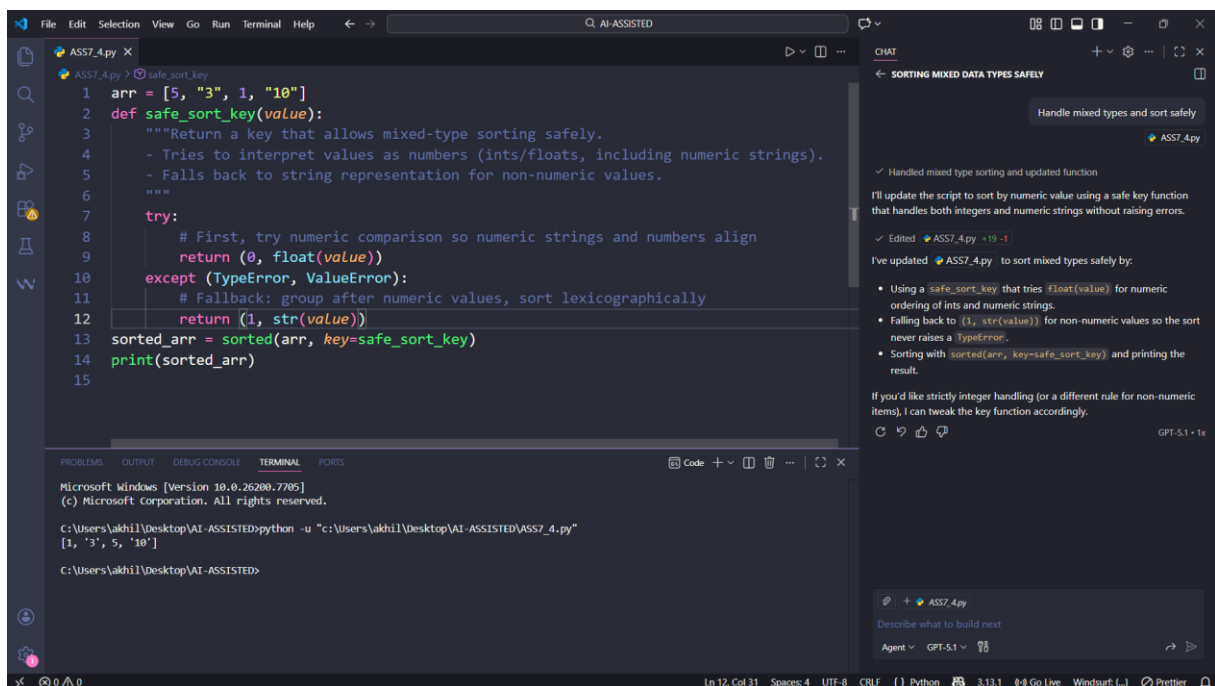
`TypeError` caused by mixed data types (e.g., integers and strings).



Using a `safe_sort_key` that tries `float(value)` for numeric ordering of ints and numeric strings.

Falling back to `(1, str(value))` for non-numeric values so the sort never raises a `TypeError`.

Sorting with `sorted(arr, key=safe_sort_key)` and printing the result.



## Task 3: Improving File Handling Reliability

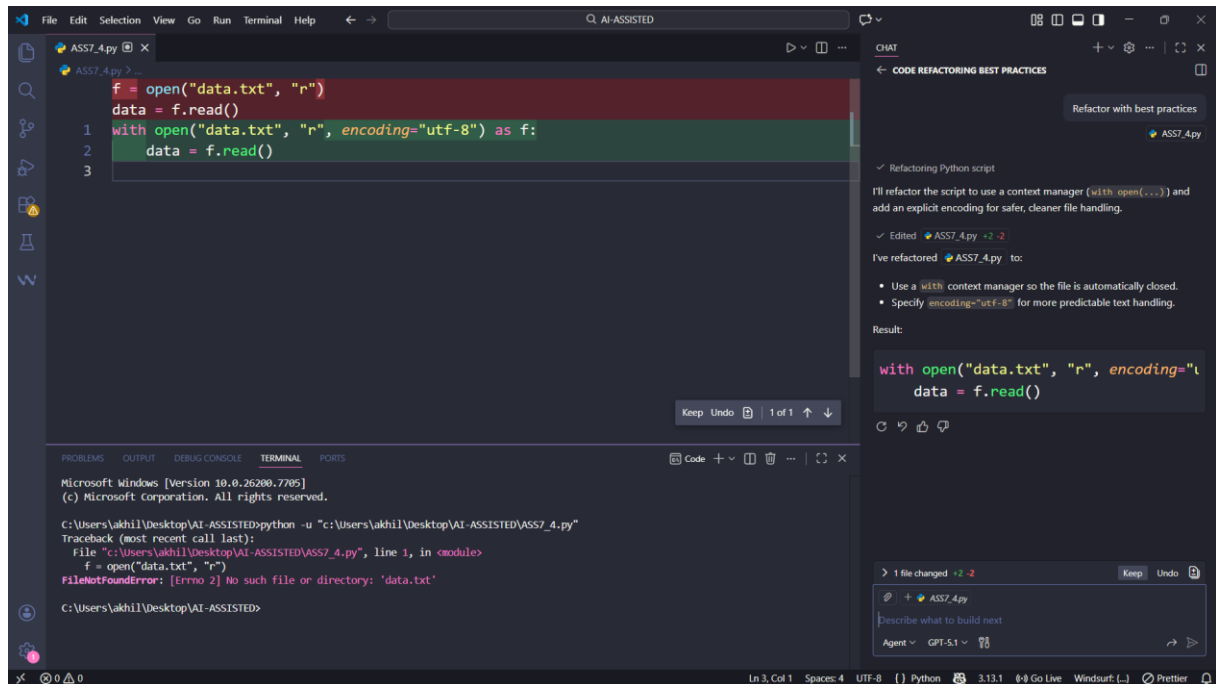
### Scenario

A backend script reads data from files regularly. Over time, the system

shows performance issues due to improper resource management.

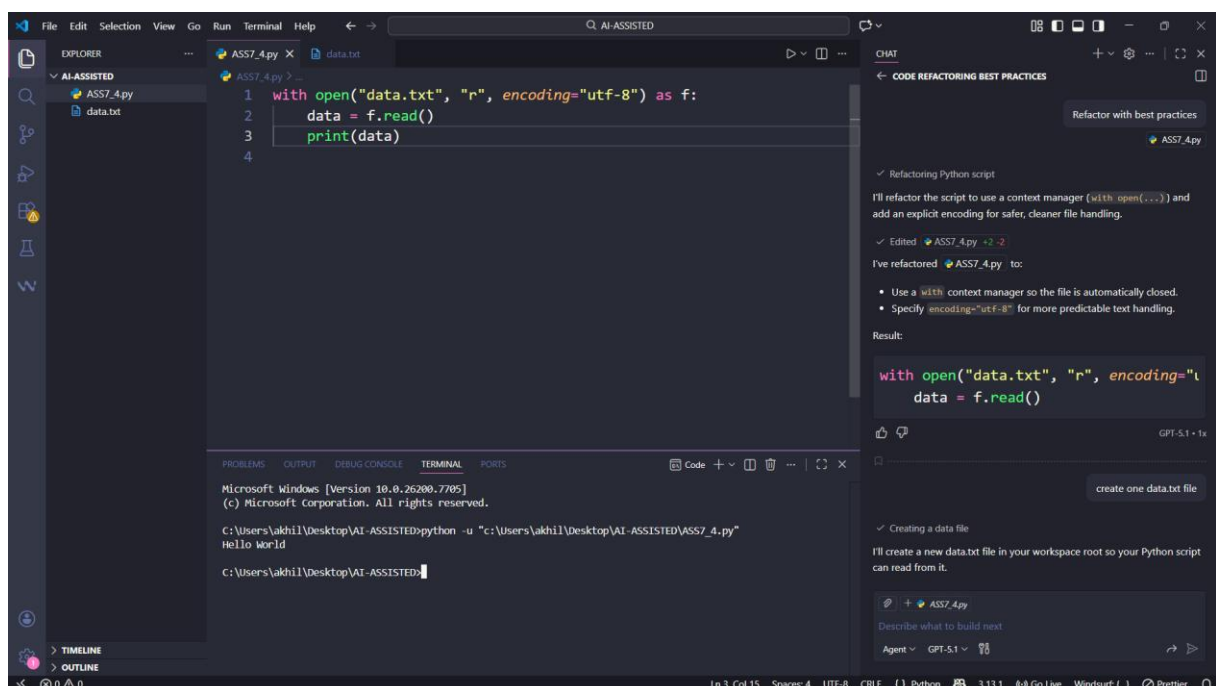
## Task Description

You are given a Python file-handling snippet that opens a file but does not explicitly close it.



Use a `with` context manager so the file is automatically closed.

Specify `encoding="utf-8"` for more predictable text handling



## Task 4: Handling Runtime Errors Gracefully in Loops

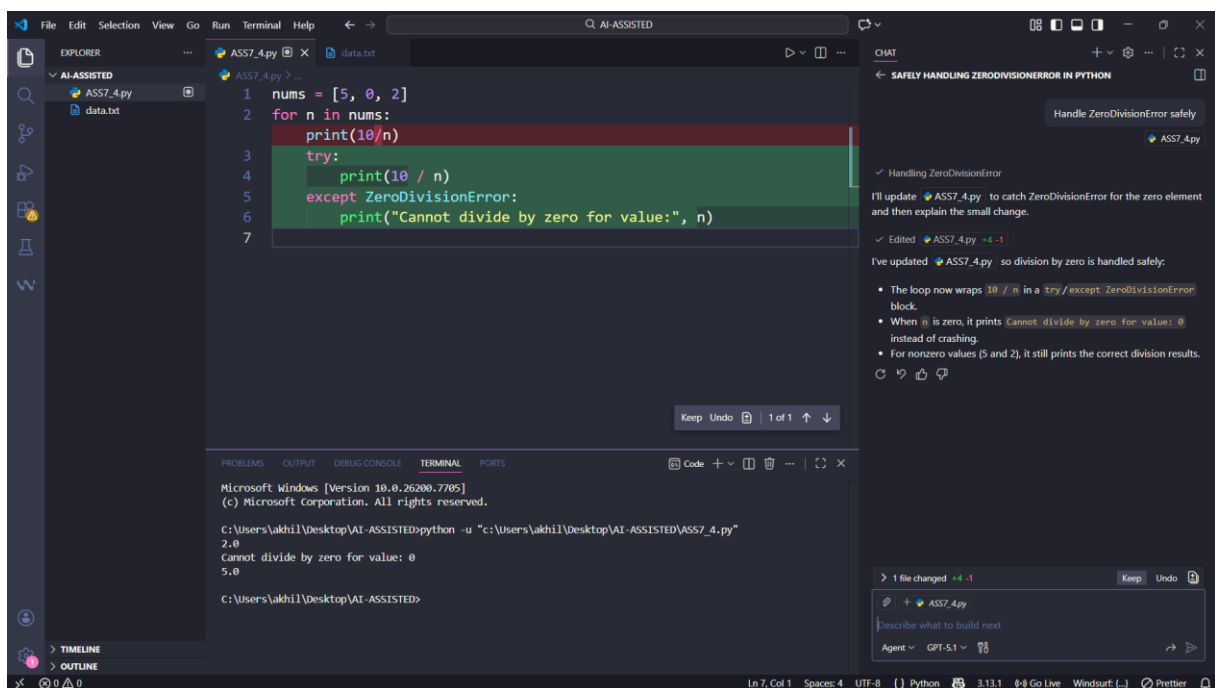
### Scenario

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.



The loop now wraps `10 / n` in a `try/except ZeroDivisionError` block.

When `n` is zero, it prints `Cannot divide by zero for value: 0` instead of crashing.

For nonzero values (5 and 2), it still prints the correct division results

## Task 5: Debugging Class Initialization Errors

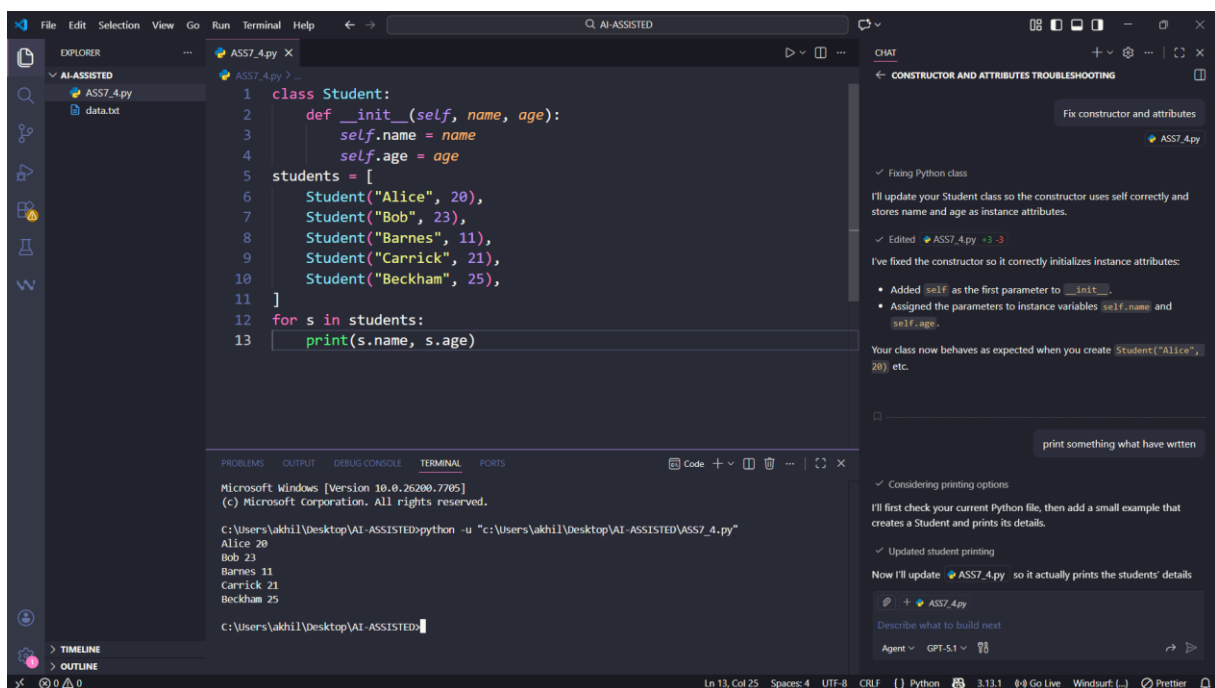
### Scenario

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`)



```
1 class Student:
2     def __init__(self, name, age):
3         self.name = name
4         self.age = age
5
6     students = [
7         Student("Alice", 20),
8         Student("Bob", 23),
9         Student("Barnes", 11),
10        Student("Carrick", 21),
11        Student("Beckham", 25),
12    ]
13    for s in students:
14        print(s.name, s.age)
```

Microsoft Windows [Version 10.0.26200.7705]  
(c) Microsoft Corporation. All rights reserved.

C:\Users\akhi\Desktop\AI-ASSISTED>python -u "c:\Users\akhi\Desktop\AI-ASSISTED\ASS7\_4.py"

Alice 20  
Bob 23  
Barnes 11  
Carrick 21  
Beckham 25

C:\Users\akhi\Desktop\AI-ASSISTED>

CHAT

CONSTRUCTOR AND ATTRIBUTES TROUBLESHOOTING

Fix constructor and attributes

✓ Fixing Python class

I'll update your Student class so the constructor uses self correctly and stores name and age as instance attributes.

✓ Edited ASS7\_4.py +3 -3

I've fixed the constructor so it correctly initializes instance attributes:

- Added `self` as the first parameter to `__init__`.
- Assigned the parameters to instance variables `self.name` and `self.age`.

Your class now behaves as expected when you create `Student("Alice", 20)` etc.

print something what have written

✓ Considering printing options

I'll first check your current Python file, then add a small example that creates a Student and prints its details.

✓ Updated student printing

Now I'll update ASS7\_4.py so it actually prints the students' details

Describe what to build next

Agent GPT-5.1

Added `self` as the first parameter to `__init__`.

Assigned the parameters to instance variables `self.name` and `self.age`.