

# AI ASSIGNMENT-7.4

NAME:B.Hasini

ROLL NO:2503A51L13

## Task Description #1:

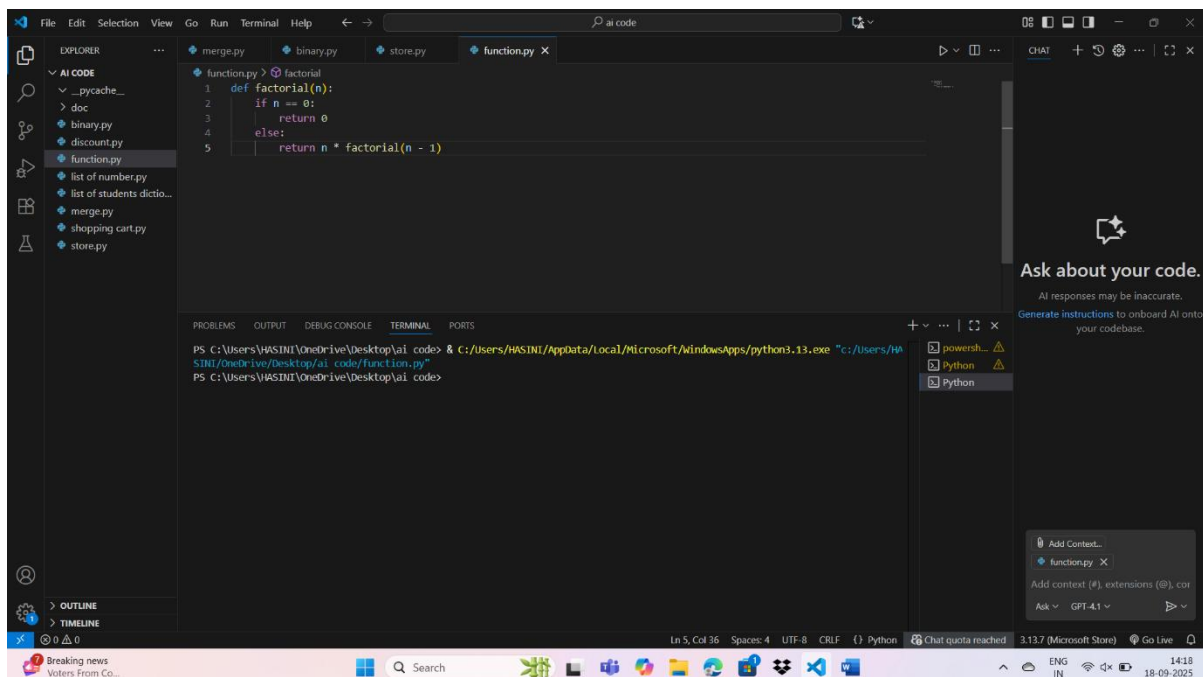
- Introduce a buggy Python function that calculates the factorial of a number using recursion. Use Copilot or Cursor AI to detect and fix the logical or syntax errors.

## Expected Outcome #1:

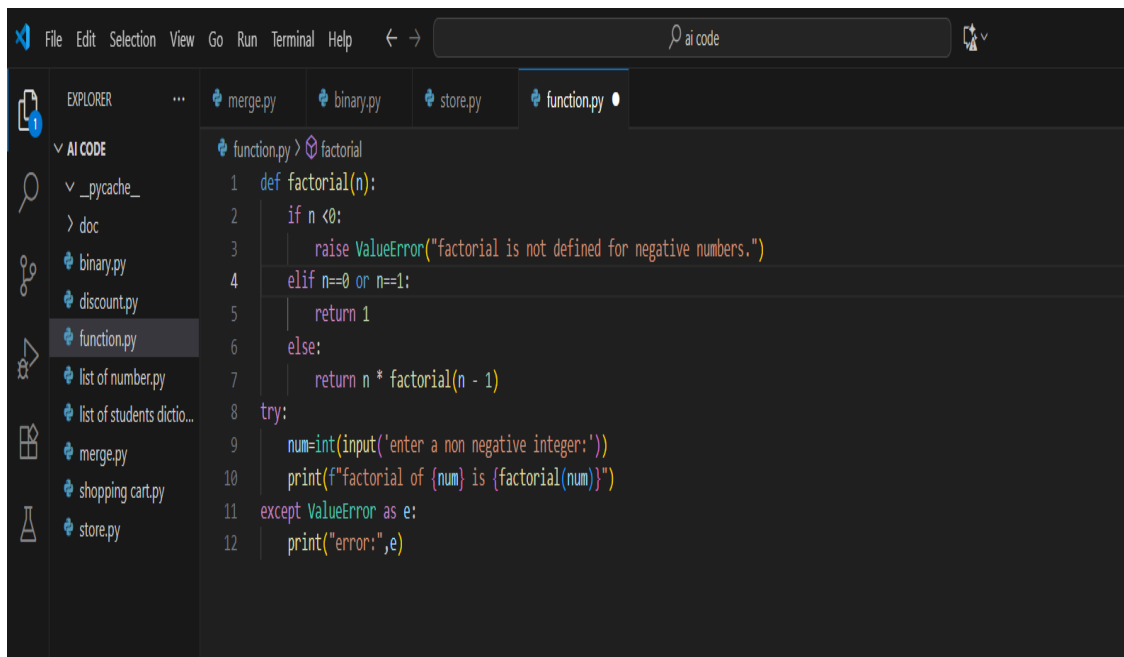
- Copilot or Cursor AI correctly identifies missing base condition or incorrect recursive call and suggests a functional factorial implementation.

## OUTPUT:

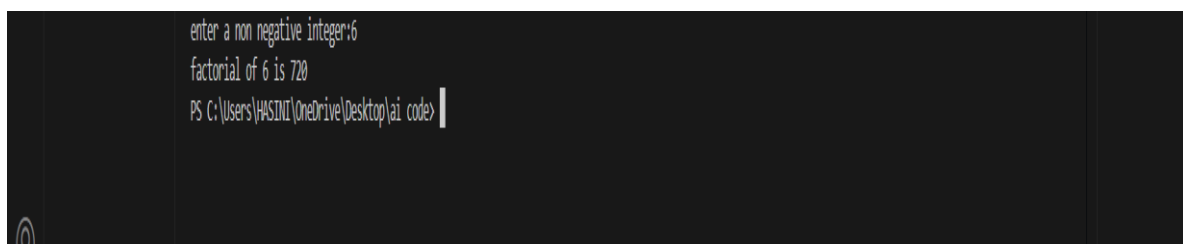
## BUGGY CODE:



## CORRECTED CODE:



```
1 def factorial(n):
2     if n < 0:
3         raise ValueError("factorial is not defined for negative numbers.")
4     elif n==0 or n==1:
5         return 1
6     else:
7         return n * factorial(n - 1)
8 try:
9     num=int(input('enter a non negative integer:'))
10    print(f"factorial of {num} is {factorial(num)}")
11 except ValueError as e:
12    print("error:",e)
```



```
enter a non negative integer:6
factorial of 6 is 720
PS C:\Users\H4SDINZ\OneDrive\Desktop\ai_code>
```

**OBSERVATION:** Negative inputs cause infinite recursion and eventually a RecursionError. AI generated a corrected code with error exception which does not allow negative values.

- Returning 0 for  $n == 0$  contradicts the mathematical definition of  $\text{factorial}(0) = 1$
- **Correct Base Case:** Returns 1 for both 0 and 1, aligning with factorial rules

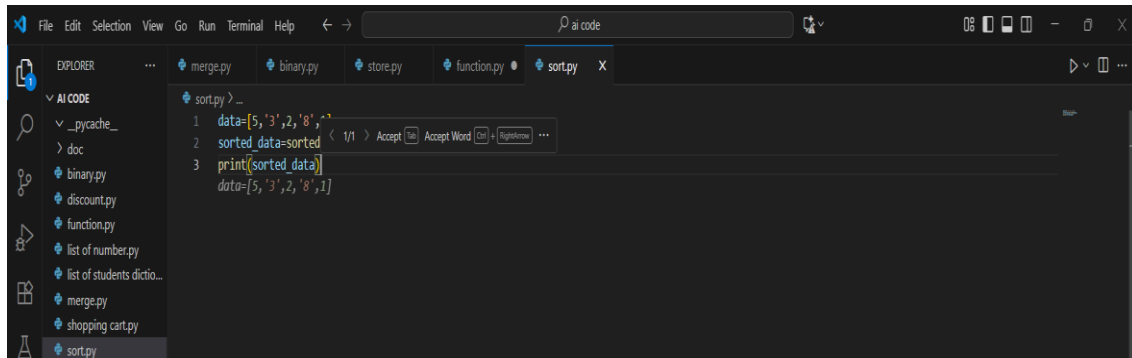
## Task Description #2:

• 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.

## Expected Outcome #2:

• AI detects the type inconsistency and either filters or converts list elements, ensuring successful sorting without a crash.

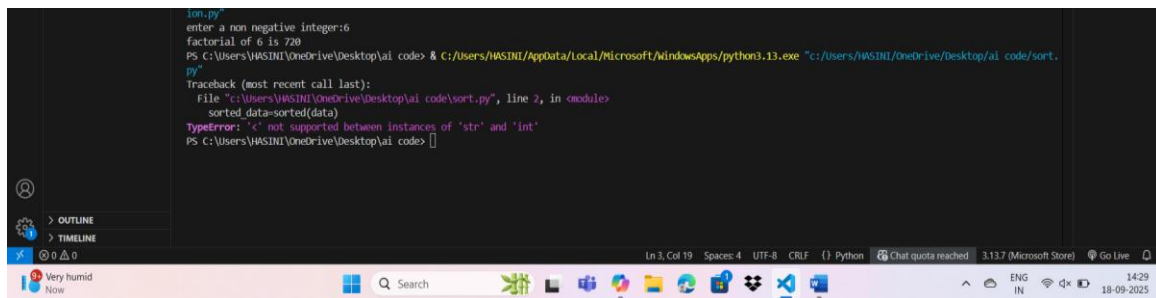
## BUGGY CODE:



The screenshot shows the Visual Studio Code editor with a file named `sort.py` open. The code in the editor is as follows:

```
1 data=[5,'3',2,'8','< 1/1 > Accept Word (m) + Signflow ...>']
2 sorted_data=sorted
3 print(sorted_data)
   data=[5,'3',2,'8',1]
```

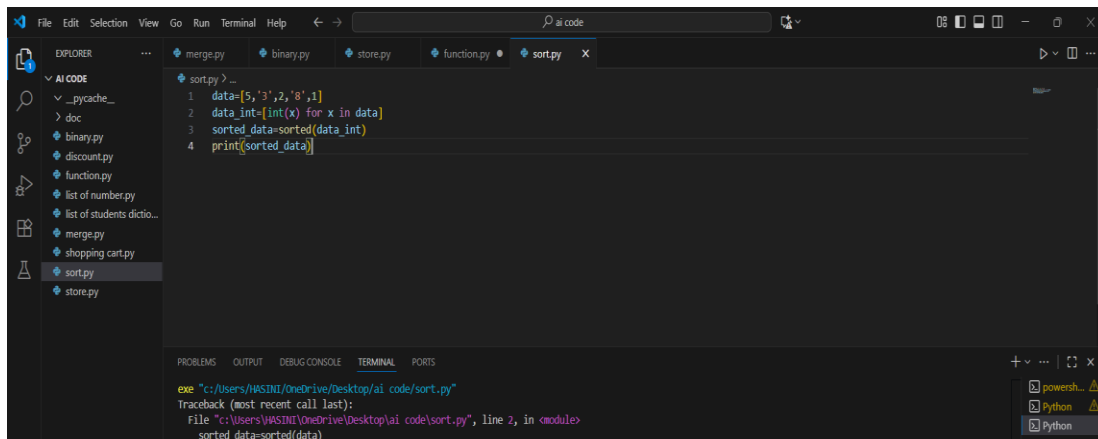
The Explorer sidebar on the left shows a project structure with a folder named `AI CODE` containing several Python files, including `sort.py`.



The screenshot shows the terminal window in VS Code. It displays the command to run the script and the resulting error:

```
PS C:\Users\HASINI\OneDrive\Desktop\ai code> & C:/Users/HASINI/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/HASINI/OneDrive/Desktop/ai code/sort.py"
Traceback (most recent call last):
  File "c:/Users/HASINI/OneDrive/Desktop/ai code/sort.py", line 2, in <module>
    sorted_data=sorted(data)
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\HASINI\OneDrive\Desktop\ai code> []
```

## CORRECTED CODE:



The screenshot shows the Visual Studio Code editor with the `sort.py` file open. The code has been corrected to handle mixed datatypes:

```
1 data=[5,'3',2,'8',1]
2 data_int=[int(x) for x in data]
3 sorted_data=sorted(data_int)
4 print(sorted_data)
```

The terminal window at the bottom shows the command to run the script and the output:

```
PS C:\Users\HASINI\OneDrive\Desktop\ai code> & C:/Users/HASINI/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/HASINI/OneDrive/Desktop/ai code/sort.py"
[1, 2, 3, 5, 8]
PS C:\Users\HASINI\OneDrive\Desktop\ai code> []
```

**OBSERVATION:** Mixed datatypes are given as input. AI has converted them into one datatype and sorted the data.

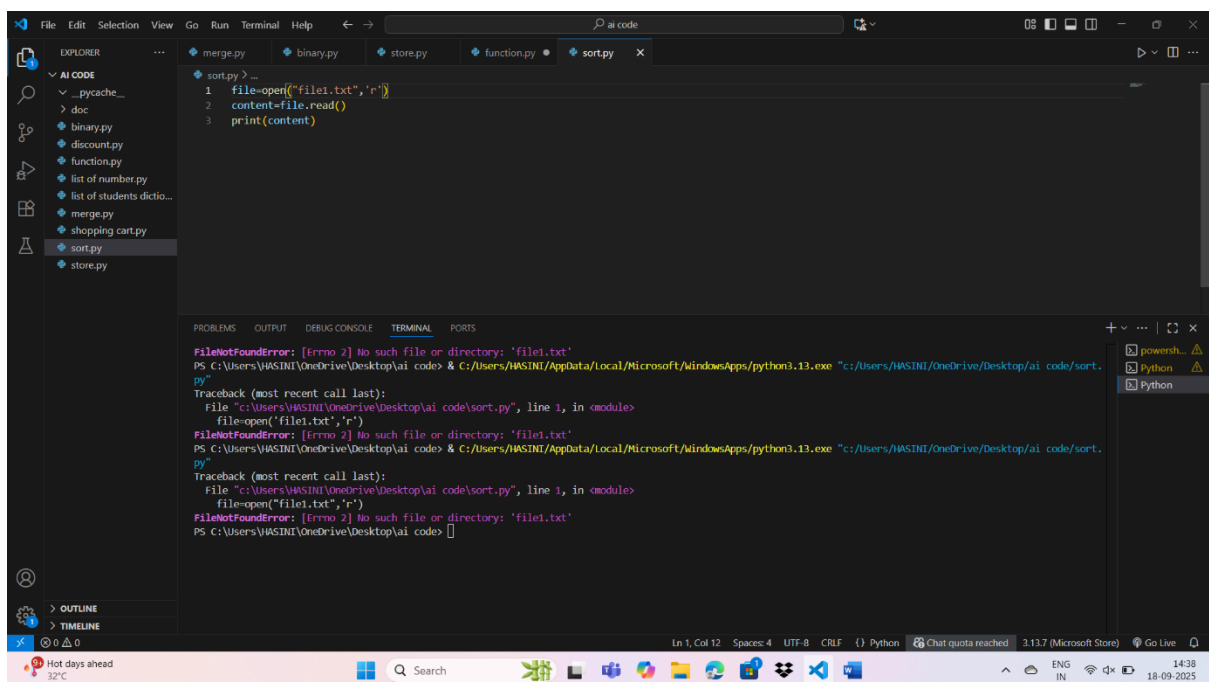
## Task Description #3:

- 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).

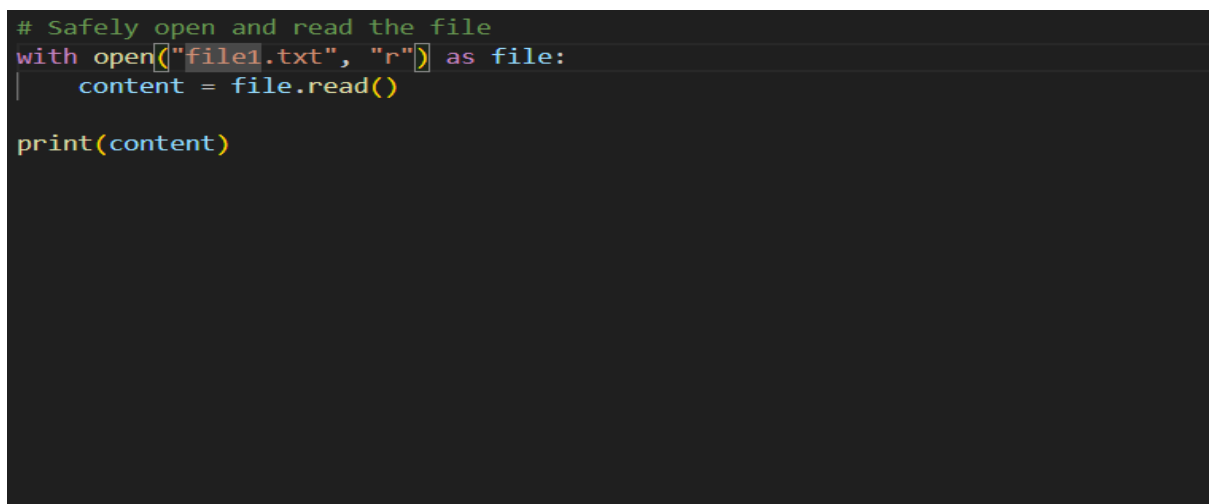
### Expected Outcome #3:

- AI refactors the code to use a context manager, preventing resource leakage and runtime warnings.

### BUGGY CODE:



### CORRECTED CODE:



```
PS C:\today> c::; cd 'c:\today'; & 'c:\Program Files\Python313\python.exe' 'c:\Users\ADHARSH\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '53058' '--' 'c:\today\seven-3.py'  
print("hello world")  
print("this is me")hello widget
```

## **OBSERVATION:**

The `file.close()` is missing so, file remains open after reading, which can:

Leak system resources, Lock the file (especially on Windows),

Trigger runtime warnings or errors in larger applications

**No Exception Handling:** If the file doesn't exist or can't be read, the code will crash without a fallback.

### **Using a Context Manager (with open):**

Automatically closes the file when the block exits—even if an error occurs

Prevents resource leakage and improves reliability.

## **Task Description #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.

## **Expected Outcome #4:**

- Copilot adds a `try-except` block around the risky operation, preventing crashes and printing a meaningful error message.

## **BUGGY CODE:**

The screenshot shows the VS Code interface with a file explorer on the left containing several Python files. The main editor displays a file named `sort.py` with the following code:

```
1 numbers=[10,5,0,2]
2 for num in numbers:
3     result=100/num
4     print(f"100 divided by {num} is {result}")
5
```

The terminal at the bottom shows the execution of the script, which results in a `ZeroDivisionError: division by zero` at line 3.

## CORRECTED CODE:

```
numbers = [10, 5, 0, 2]

for num in numbers:
    try:
        result = 100 / num
        print(f"100 divided by {num} is {result}")
    except ZeroDivisionError:
        print(f"Cannot divide by zero when num = {num}. Skipping...")
```

```
PS C:\today> c:: cd 'c:\today'; & 'c:\Program Files\Python313\python.exe' 'c:\Users\ADHARSH\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '54730' '--' 'c:\today\seven-4.py'
100 divided by 10 is 10.0
100 divided by 5 is 20.0
Cannot divide by zero when num = 0. Skipping...
100 divided by 2 is 50.0
```

## OBSERVATION:

There is no Error Handling.

**Poor User Feedback:** No indication of what went wrong or which value caused the issue.

**Try Except block added:** isolates risky operation and catches the specific error.

**Improved Feedback:** Prints a clear message when an error occurs, aiding debugging and user understanding.

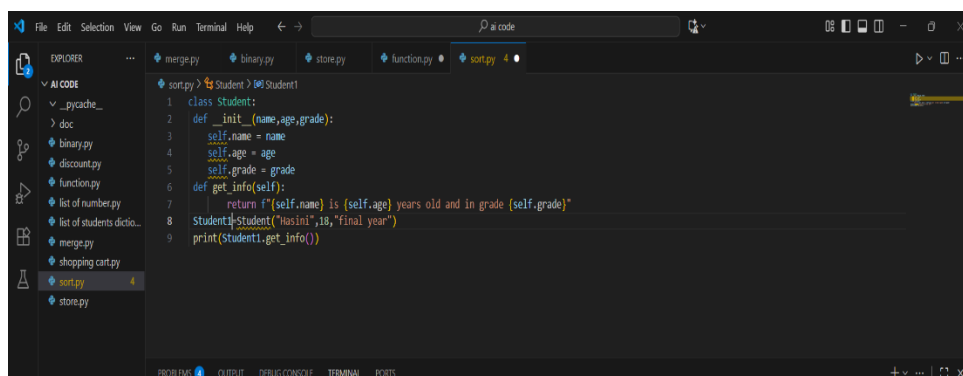
### Task Description #5:

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

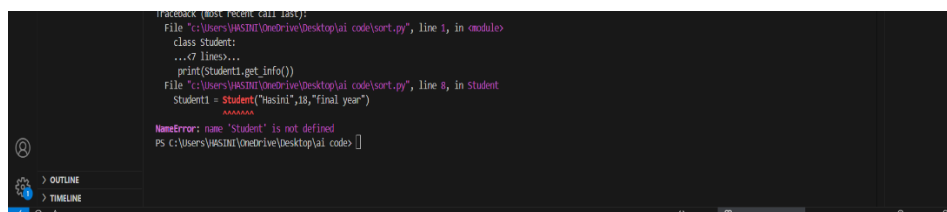
### Expected Outcome #5:

- Copilot identifies mismatched parameters or missing self references and rewrites the class with accurate initialization and usage.

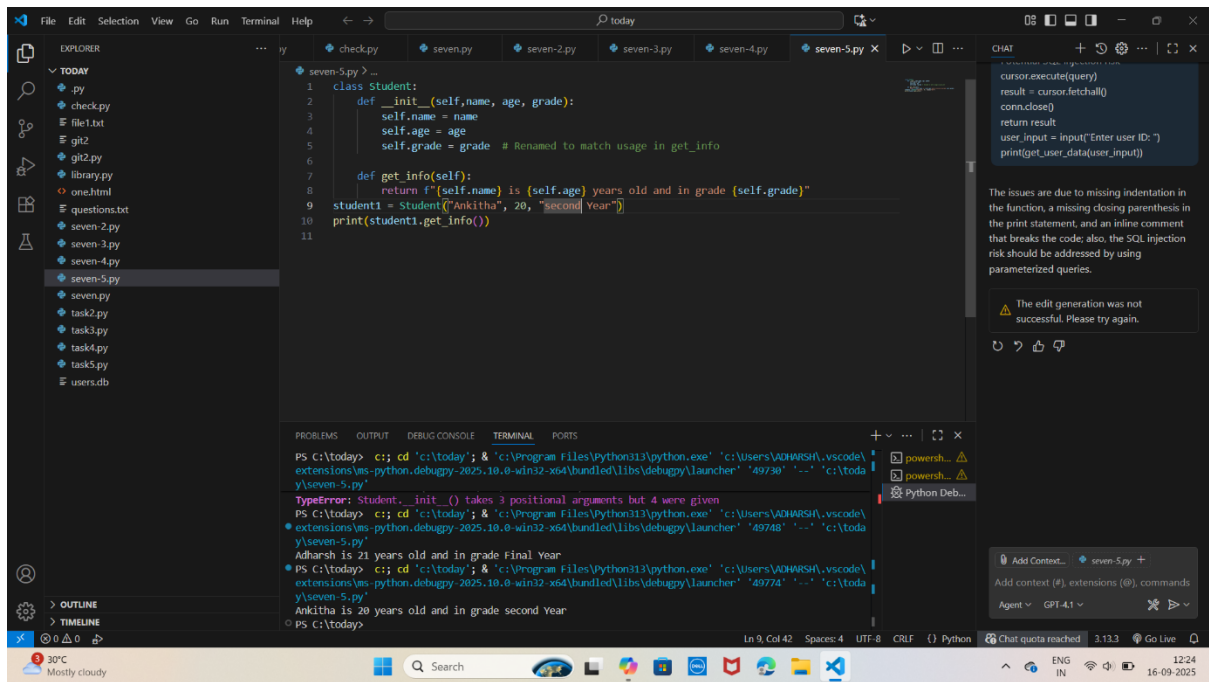
### BUGGY CODE:



```
1 class Student:
2     def __init__(name,age,grade):
3         self.name = name
4         self.age = age
5         self.grade = grade
6     def get_info(self):
7         return f'{self.name} is {self.age} years old and in grade {self.grade}'
8     Student1=Student("hasini",18,"final year")
9     print(Student1.get_info())
```



```
Traceback (most recent call last):
  File "c:\Users\VASINI\OneDrive\Desktop\ai code\sort.py", line 1, in module
    class Student:
    ...7 lines...
    print(Student1.get_info())
  File "c:\Users\VASINI\OneDrive\Desktop\ai code\sort.py", line 8, in Student
    Student1 = Student("hasini",18,"final year")
              ^^^^^^^
NameError: name 'Student' is not defined
PS C:\Users\VASINI\OneDrive\Desktop\ai code>
```



## OBSERVATION:

- Missing self in `__init__` parameters: Python requires self as the first argument in instance methods to refer to the object itself.
- self is the reference to the current instance—required in all instance methods.



