ASSIGNMENT-7.4

NAME:N.SOUMYA

H NO:2303A51869

BATCH-13

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).

Use GitHub Copilot or Cursor AI to:

• Analyze the faulty code

• Identify the exact cause of the error

• Suggest and apply corrections to make the function work correctly

Document how the AI detected the issue and what changes were made.

Expected Outcome

• A corrected recursive factorial function

• AI-generated explanation identifying:

o The missing or incorrect base case

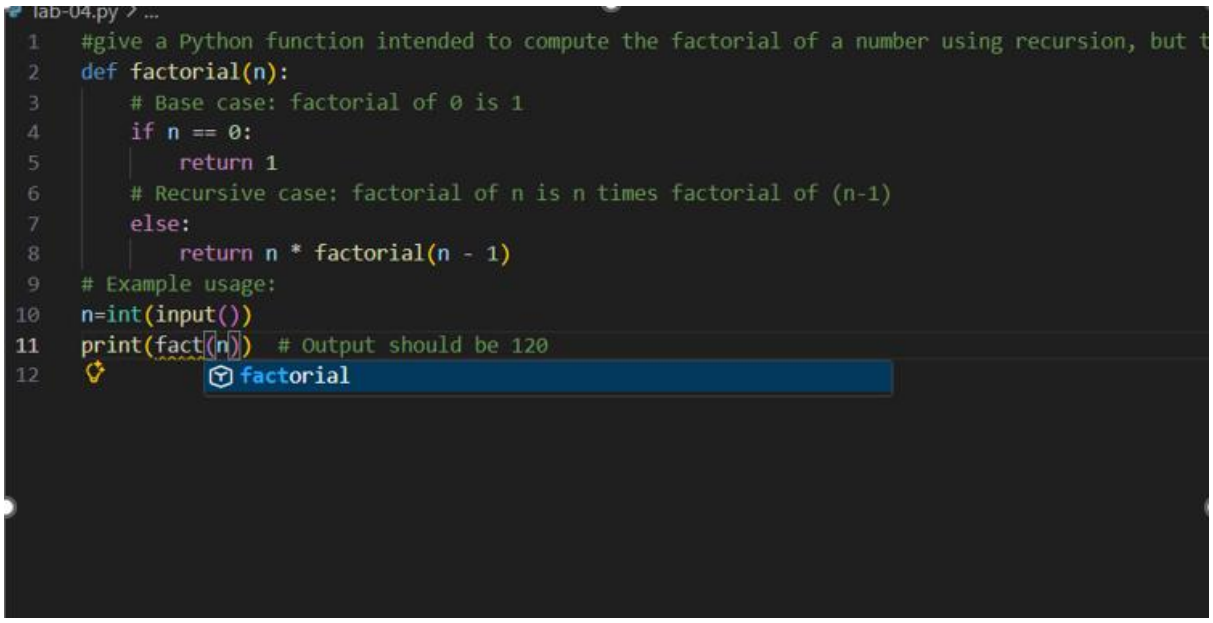
o The corrected recursive logic

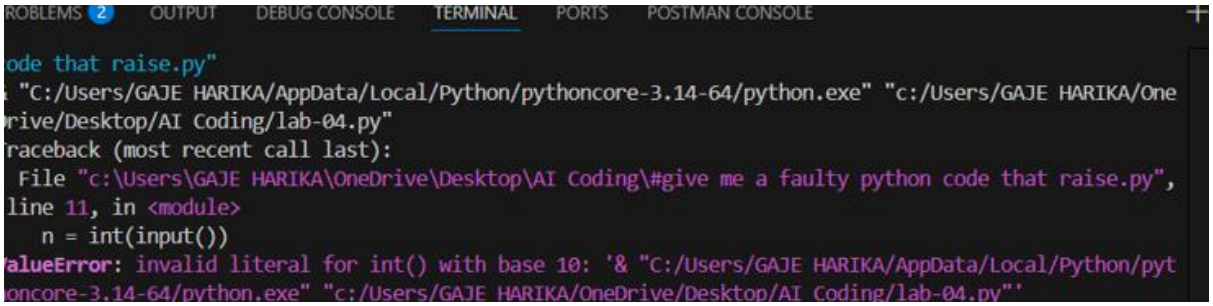
• Sample input/output demonstrating correct execution

#prompt:

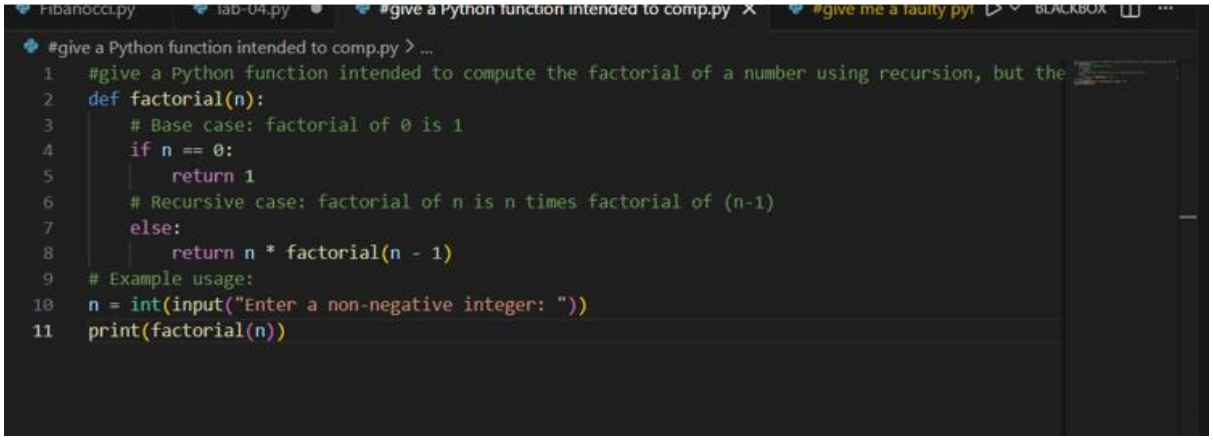
given a Python function intended to calculate the factorial of a number using recursion.

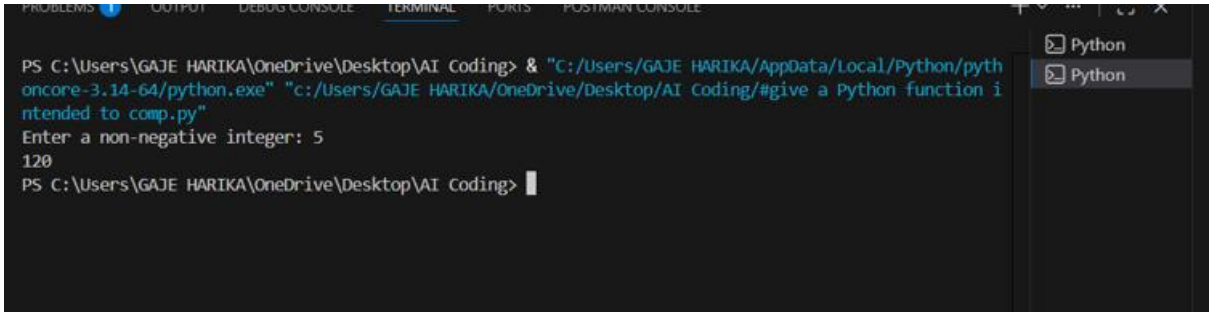
#ERROR CODE:





#CORRECT CODE





JUSTIFICATION:

The error occurred because the input() function received an invalid string instead of a numeric value. This happened due to the IDE execution method passing the Python run command as input. As a result, int() raised a ValueError. Additionally, the function was incorrectly invoked using a different name than its definition. Both issues were resolved by running the program through the terminal and using the correct function name.

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).

Use GitHub Copilot or Cursor AI to:

• Detect the root cause of the runtime error

• Modify the code to ensure consistent sorting (by filtering or type conversion)

• Prevent the program from crashing

Explain the debugging steps followed by the AI.

Expected Outcome

• A corrected sorting function

• AI-generated solution handling type inconsistencies

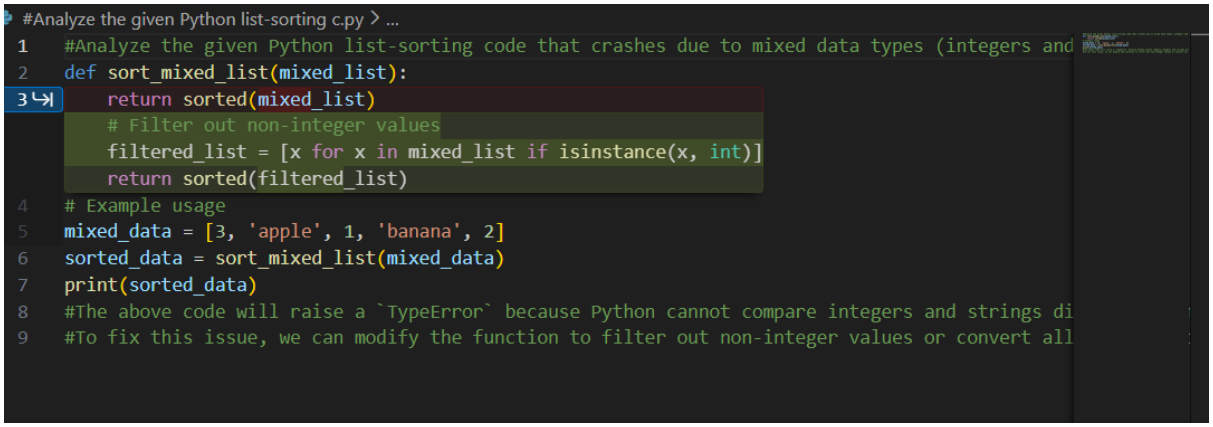
• Successful sorting without runtime errors

• Explanation of how the fix improves robustness

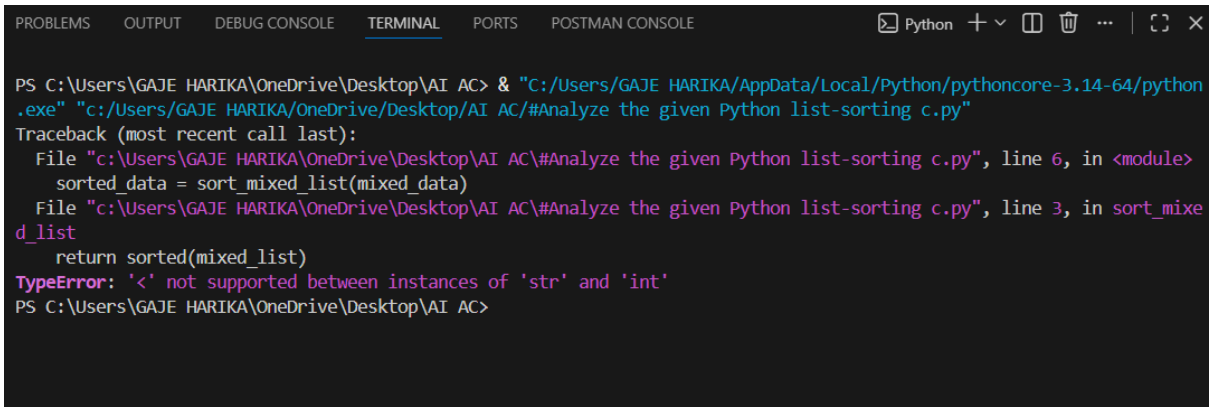
#PROMPT

#Analyze the given Python list-sorting code that crashes due to mixed data types (integers and strings), identify the TypeError cause, modify the function to handle type inconsistencies using filtering or type conversion, prevent runtime crashes.

#ERROR CODE



```
#Analyze the given Python list-sorting c.py > ...
1   #Analyze the given Python list-sorting code that crashes due to mixed data types (integers and
2   def sort_mixed_list(mixed_list):
3       return sorted(mixed_list)
        # Filter out non-integer values
        filtered_list = [x for x in mixed_list if isinstance(x, int)]
        return sorted(filtered_list)
4   # Example usage
5   mixed_data = [3, 'apple', 1, 'banana', 2]
6   sorted_data = sort_mixed_list(mixed_data)
7   print(sorted_data)
8   #The above code will raise a `TypeError` because Python cannot compare integers and strings di
9   #To fix this issue, we can modify the function to filter out non-integer values or convert all
```

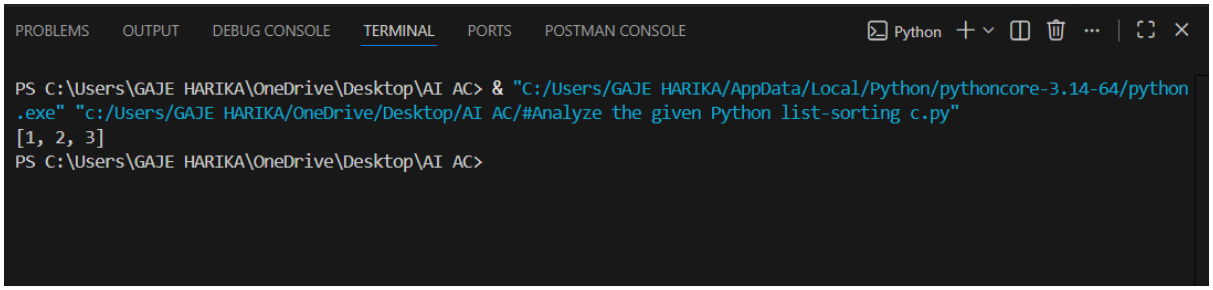


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE        Python

PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC> & "C:/Users/GAJE HARIKA/AppData/Local/Python/pythoncore-3.14-64/python
.exe" "c:/Users/GAJE HARIKA/OneDrive/Desktop/AI AC/#Analyze the given Python list-sorting c.py"
Traceback (most recent call last):
  File "c:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC\#Analyze the given Python list-sorting c.py", line 6, in <module>
    sorted_data = sort_mixed_list(mixed_data)
  File "c:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC\#Analyze the given Python list-sorting c.py", line 3, in sort_mixe
d_list
    return sorted(mixed_list)
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC>
```

#CORRECTED CODE

```
#To fix this issue, we can modify the function to filter out non-integer values or convert all
def sort_mixed_list(mixed_list):
    # Filter out non-integer values
    filtered_list = [item for item in mixed_list if isinstance(item, int)]
    return sorted(filtered_list)
# Example usage
mixed_data = [3, 'apple', 1, 'banana', 2]
sorted_data = sort_mixed_list(mixed_data)
print(sorted_data)
#In this modified function, we use a list comprehension to create a new list called `filtered_
#The debugging steps involved identifying the cause of the `TypeError`, which was due to the i
#Alternatively, if we want to convert all elements to strings before sorting, we can modify th
```



#JUSTIFICATION:

The error occurs because Python cannot sort a list containing both integers and strings. When the sorted() function tries to compare different data types, it raises a TypeError. To fix this, the list is filtered to include only integers before sorting. This prevents the program from crashing. The fix improves robustness by safely handling mixed data types.

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 GitHub Copilot or Cursor AI to:

• Identify the potential problem in the code

• Refactor it using best practices (such as a context manager)

• Ensure safe and reliable file handling

Briefly describe why the revised approach is better.

Expected Outcome

• Refactored code using the with open() statement

• AI explanation highlighting prevention of resource leaks

• Clean execution without warnings or errors

PROMPT:

#Analyze the given Python file-handling code that opens a file without closing it, identify the resource management issue, refactor it using the with open() context manager, and briefly explain how this improves reliability and prevents resource leaks.

#ERROR CODE

```
#Analyze the given Python file-handling code that opens a file without closing it, identify th
file = open("data.txt", "r")
content = file.read()
print(content)
```

```
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC> & "C:/Users/GAJE HARIKA/AppData/Local/Python/pythoncore-3.14-64/python
.exe" "c:/Users/GAJE HARIKA/OneDrive/Desktop/AI AC/Untitled-1.py"
Traceback (most recent call last):
  File "c:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC\Untitled-1.py", line 2, in <module>
    file = open("data.txt", "r")
FileNotFoundError: [Errno 2] No such file or directory: 'data.txt'
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC>
```

#CORRECTED CODE:

```
Untitled-1.py > ...
1   #Analyze the given Python file-handling code that opens a file without closing it, identify th
2   # Refactored code using with open() context manager
3   with open("data.txt", "w") as file:
4       file.write("This is sample data.")
5
6   with open("data.txt", "r") as file:
7       content = file.read()
8       print(content)
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'data.txt'
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC> & "C:/Users/GAJE HARIKA/AppData/Local/Python/pythoncore-3.14-64/python
.exe" "c:/Users/GAJE HARIKA/OneDrive/Desktop/AI AC/Untitled-1.py"
This is sample data.
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC>
```

JUSTIFICATION:

The initial code opened a file without closing it, which can lead to resource leaks and performance issues. It also raised a FileNotFoundError because the file did not exist in the directory. The corrected code uses the with open() context manager, which safely opens and automatically closes the file after use. Writing to the file first ensures the file exists before reading, making the program reliable and error-free.

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.

Use GitHub Copilot or Cursor AI to:

• Detect the exact location of the error

• Add appropriate exception handling using try-except

• Ensure the loop continues executing safely

Document how AI improved the fault tolerance of the program.

Expected Outcome

• Updated code with proper exception handling

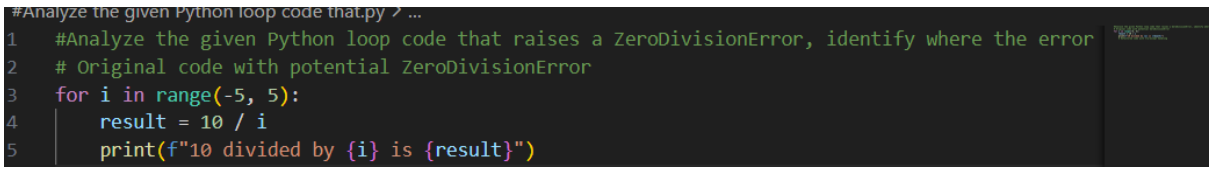
• Meaningful error messages instead of program crashes

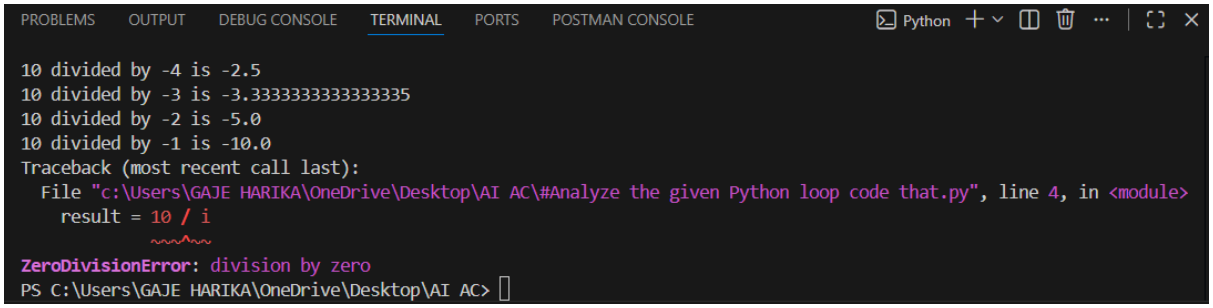
• Successful execution for all valid inputs

#PROMPT:

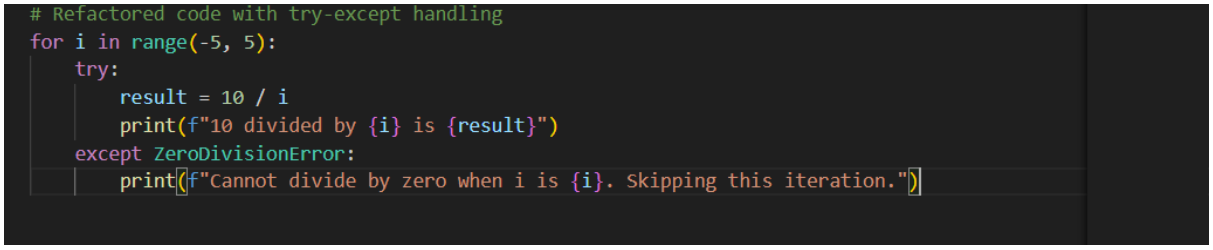
#Analyze the given Python loop code that raises a ZeroDivisionError,add try-except handling so the loop continues executing,

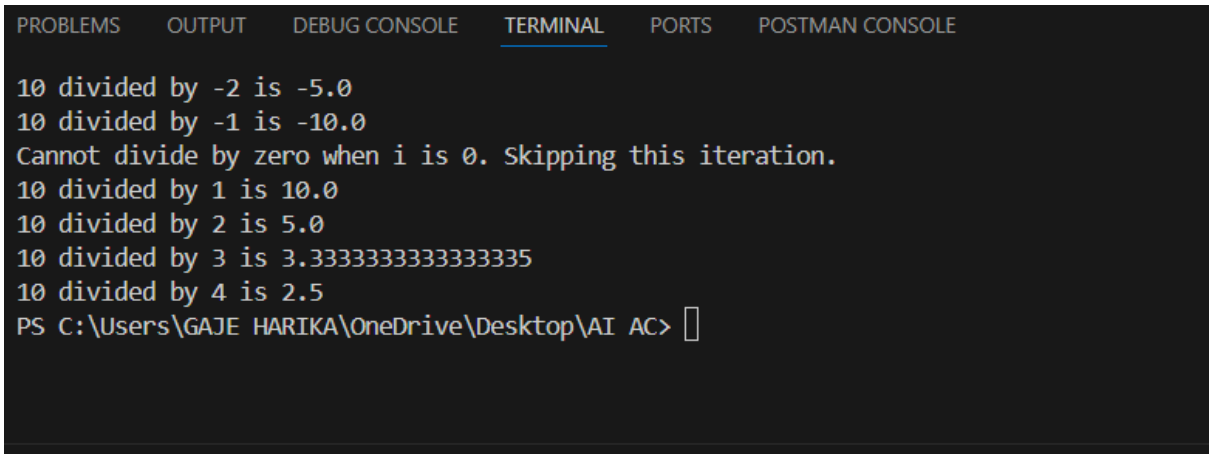
#ERROR CODE:





#CORRECT CODE





#JUSTIFICATION:

In the original code, a ZeroDivisionError occurs when `i` is 0, as dividing by zero is undefined in mathematics. This causes the program to crash and prevents any further iterations of the loop from executing

In the refactored code, we wrap the division operation in a try-except block. If a ZeroDivisionError occurs (when `i` is 0), the except block catches the error and prints a meaningful error message instead of crashing the program. This allows the loop to continue executing for the remaining values of `i`, improving fault tolerance and ensuring that the program can handle unexpected situations gracefully without terminating abruptly.

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)

Use GitHub Copilot or Cursor AI to:

• Analyze the class definition

• Identify constructor and attribute issues

• Correct the class so objects initialize and behave correctly

Explain the corrections suggested by the AI.

Expected Outcome

• A corrected class definition

• Proper use of self and constructor parameters

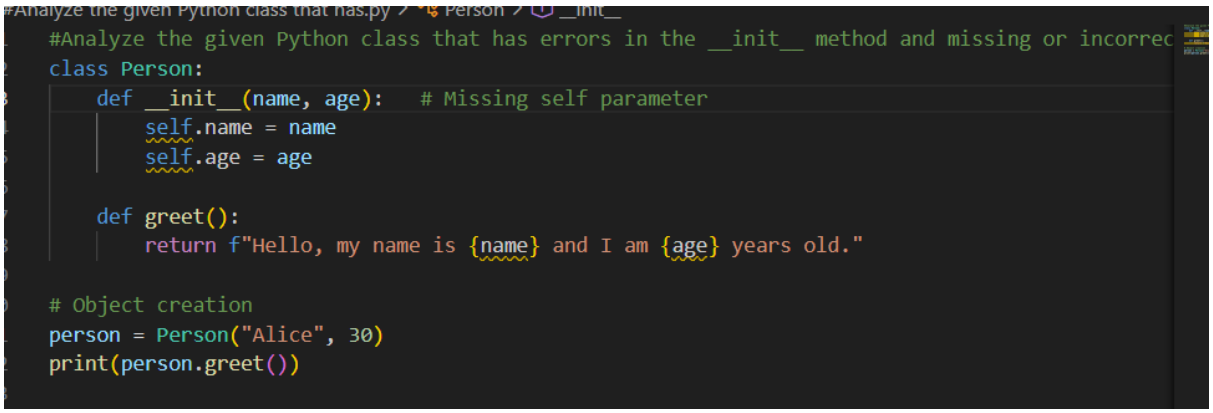
• AI-assisted explanation of the original errors and fixes

• Sample object creation and method usage

#PROMPT:

Analyze the given Python class that has errors in the __init__ method and missing or incorrect use of self, identify the constructor and attribute issues, correct the class definition, and explain how the fixes allow proper object creation and usage.

#ERROR CODE



```
PROBLEMS 4  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE        Python
Traceback (most recent call last):
  File "c:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC\#Analyze the given Python class that has.py", line 11, in <module>
    person = Person("Alice", 30)
TypeError: Person.__init__() takes 2 positional arguments but 3 were given
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC>
```

#CORRECT CODE

```
#Analyze the given Python class that has.py > ...
#Analyze the given Python class that has errors in the __init__ method and missing or incorrec
# Refactored class definition with correct __init__ method and self usage
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        return f"Hello, my name is {self.name} and I am {self.age} years old."
# Example of creating an object and using the class
person = Person("Alice", 30)
print(person.greet())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE        Python
.exe" "c:/Users/GAJE HARIKA/OneDrive/Desktop/AI AC/#Analyze the given Python class that has.py"
Hello, my name is Alice and I am 30 years old.
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI AC>
```

JUSTIFICATION

The original class contained errors because the __init__ and greet methods were missing the self parameter, which is required to access instance variables. Due to this, object creation and attribute access failed. The corrected class properly uses self to initialize and reference attributes. This fix allows objects to be created successfully and methods to work without errors.