

AI Assistant Coding

Name: K.Lokesh

HTNO: 2303A51862

Batch-13

Date: 05-02-2026

Assignment - 7.4

Problem statements with solution:

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.

```
def factorial(n):
    result = 1
    while True:
        result *= i
    return result

print(factorial(5)) # Expected output: 120
```

```
clg@Arushs-MacBook-Air sru 3rd 2nd sem % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task1.py"
Traceback (most recent call last):
  File "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task1.py", line 8, in <module>
    print(factorial(5)) # Expected output: 120
  File "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task1.py", line 4, in factorial
    result *= i
NameError: name 'i' is not defined
```

CHAT

+ ⌂ ⚙ ⌂ ⌂ X

← IDENTIFYING THE BUG IN CODE



what's the bug

✓ Read Python file

I found the bugs in your code:

1. **Infinite loop:** The `while True:` condition never terminates—there's no `break` statement and `[i]` is never modified.
2. **Undefined variable** `[i]`: The variable `[i]` is never defined, so the loop references a non-existent variable.
3. **Missing loop counter:** You need a loop to iterate from 1 to `[n]` and multiply them together.

Here's the fix:

✓ Edited `* task1.py +1 -1`

Changed `while True:` to `for i in range(1, n + 1):` to properly iterate from 1 to `n`. This multiplies all numbers together to calculate the factorial correctly.

C ↵ ⌂ ⌂

```
1 def factorial(n):
2     result = 1
3     while True:
4         for i in range(1, n + 1):
5             result *= i
6     return result
7
8 print(factorial(5)) # Expected output: 120
```

Keep Undo ⌂

correct execution:

```
Ai coding > assignment 7.4 > task1.py > factorial
1 def factorial(n):
2     result = 1
3     for i in range(1, n + 1):
4         result *= i
5     return result
6
7
8 print(factorial(5)) # Expected output: 120

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
● clg@Arushs-MacBook-Air ~ % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/7.4/task1.py"
120
```

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.

```

def sort_user_values(values):
    return sorted(values)

user_input = [10, "20", 5, "3"]
sorted_values = sort_user_values(user_input)
print(sorted_values)

```

```

clg@Arushs-MacBook-Air sru 3rd 2nd sem % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task2.py"
Traceback (most recent call last):
  File "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task2.py", line 7, in <module>
    sorted_values = sort_user_values(user_input)
  File "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task2.py", line 2, in sort_user_values
    return sorted(values)
TypeError: '<' not supported between instances of 'str' and 'int'

```

```

def sort_user_values(values):
    return sorted(values)

user_input = [10, "20", 5, "3"]
user_input = [10, 20, 5, 3]

sorted_values = sort_user_values(user_input)
print(sorted_values)

```

Keep Undo ⌛ | 1 of 1 ↑ ↓

correct execution:

```

Ai coding > assignment 7.4 > task2.py > [ɔ] sorted_values
1 def sort_user_values(values):
2     return sorted(values)
3
4
5 user_input = [10, 20, 5, 3]
6
7 sorted_values = sort_user_values(user_input)
8 print(sorted_values)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

clg@Arushs-MacBook-Air sru 3rd 2nd sem % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task2.py"
[3, 5, 10, 20]
clg@Arushs-MacBook-Air sru 3rd 2nd sem %

```

The fix improves robustness in several ways:

1. Eliminates type errors: Mixed types cause Python to crash when trying to compare incompatible objects. By ensuring all values are integers, the sorted() function works reliably.
2. Type consistency: The code now handles data predictably. Every element can be compared with every other element using the same logic.

3. Deterministic behavior: With consistent types, sorting produces expected results every time. The output is deterministic rather than error-prone.
4. Better maintainability: Code dealing with homogeneous data types is easier to understand and debug.

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.

```
Ai coding > assignment 7.4 > task3.py > ...
1 def read_data():
2     file = open("data.txt", "r") # File opened but never closed
3     data = file.read()
4     return data
5
6
7 # Simulating repeated file access
8 for _ in range(10000):
9     read_data()
10

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

clg@Arushs-MacBook-Air ~ % /usr/bin/python3 "/Users/clg/Myh/sru/assignment 7.4/task3.py"
Traceback (most recent call last):
  File "/Users/clg/Myh/sru/assignment 7.4/task3.py", line 9, in <module>
    read_data()
  File "/Users/clg/Myh/sru/assignment 7.4/task3.py", line 2, in read_data
    file = open("data.txt", "r") # File opened but never closed
FileNotFoundError: [Errno 2] No such file or directory: 'data.txt'
```

Ai coding > assignment 7.4 > task3.py > ...

```
1 def read_data():
2     file = open("data.txt", "r") # File opened but never closed
3     data = file.read()
4     with open("data.txt", "r") as file:
5         data = file.read()
6     return data
7
8 # Simulating repeated file access
9 for _ in range(10000):
10    read_data()
```

Keep Undo ⌛ | 1 of 1 ↑ ↓

This screenshot shows a code editor with a syntax error highlighted in red. The error is in line 2, where the file object is used before it is closed. The code is intended to use a context manager to handle the file properly.

correct execution:

Ai coding > assignment 7.4 > task3.py > read_data

```
1 def read_data():
2     with open("/Ai coding/assignment 7.4/data.txt", "r") as file:
3         data = file.read()
4     return data
5
6
7 # Simulating repeated file access
8 for _ in range(10000):
9    read_data()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

clg@Arushs-MacBook-Air ~ % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task3.py"

This screenshot shows the same code as the previous one, but it has been corrected. The syntax error is fixed, and the code now runs successfully in a terminal window below the code editor. The terminal output shows the command being run.

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.

```

1 def process_values(values):
2     results = []
3     for v in values:
4         result = 100 / v # ✖️ crashes when v == 0
5         results.append(result)
6     return results
7
8
9 data = [10, 5, 0, 20, 4]
10
11 # Program stops at v == 0
12 output = process_values(data)
13 print(output)
14

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

clg@Arusns-MacBook-Air sru 3rd 2nd sem % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task3.py"
● clg@Arusns-MacBook-Air sru 3rd 2nd sem % clear
● clg@Arusns-MacBook-Air sru 3rd 2nd sem % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task3.py"
○ clg@Arusns-MacBook-Air sru 3rd 2nd sem % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task4.py"
Traceback (most recent call last):
  File "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task4.py", line 12, in <module>
    output = process_values(data)
  File "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task4.py", line 4, in process_values
    result = 100 / v # ✖️ crashes when v == 0
ZeroDivisionError: division by zero

```

```

Ai coding > assignment 7.4 > task4.py > ...
1 def process_values(values):
2     results = []
3     for v in values:
4         result = 100 / v # ✖️ crashes when v == 0
5         results.append(result)
6     try:
7         result = 100 / v
8         results.append(result)
9     except ZeroDivisionError:
10        results.append(None) # or 0, or skip it
11     except (TypeError, ValueError):
12        results.append(None) # Handle invalid types
13
14     return results
15
16 data = [10, 5, 0, 20, 4]
17
18 # Program stops at v == 0
19 output = process_values(data)
20 print(output)
21

```

Keep Undo ⌛ | 1 of 1 ↑ ↓

correct execution:

```

Ai coding > assignment 7.4 > task4.py > ...
1 def process_values(values):
2     results = []
3     for v in values:
4         try:
5             result = 100 / v
6             results.append(result)
7         except ZeroDivisionError:
8             results.append(None) # or 0, or skip it
9         except (TypeError, ValueError):
10            results.append(None) # Handle invalid types
11
12     return results
13
14 data = [10, 5, 0, 20, 4]
15
16 # Program stops at v == 0
17 output = process_values(data)
18 print(output)
19

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

clg@Arusns-MacBook-Air sru 3rd 2nd sem % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task4.py"
[10.0, 20.0, None, 5.0, 25.0]

```

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.

The screenshot shows a code editor with a Python file named `task5.py`. The code defines a `User` class with an `__init__` method and a `greet` method. A comment indicates an attempt to create and use the object. The terminal below shows a traceback for a `TypeError` occurring at line 11, where the `__init__` method is called with three arguments instead of two.

```
1  class User:
2      def __init__(name, age):
3          self.name = name
4          age = age
5
6      def greet():
7          return "Hello, my name is " + self.name
8
9
10 # Attempt to create and use the object
11 u = User("Alice", 30)
12 print(u.name)
13 print(u.greet())
14
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
clg@rushs-MacBook-Air ~ % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task5.py"
Traceback (most recent call last):
  File "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task5.py", line 11, in <module>
    u = User("Alice", 30)
TypeError: __init__() takes 2 positional arguments but 3 were given
```

```
1 class User:
2     def __init__(name, age):
3         self.name = name
4         age = age
5         self.age = age
6
7     def greet():
8         def greet(self):
9             return "Hello, my name is " + self.name
10
11 # Attempt to create and use the object
12 u = User("Alice", 30)
13 print(u.name)
14 print(u.greet())
15
```

Keep Undo ⌛ | 3 of 3 ↑ ↓

Problem:

1. First parameter must be `self` to reference the instance
2. Creates a local variable, doesn't store it on the object
3. Method has no `self` parameter, can't access instance variables

Fix:

1. `def __init__(self, name, age):`
2. `self.age = age` to store on instance
3. `def greet(self):` to enable `self.name` access

`self` represents the current object instance. All instance methods must have `self` as the first parameter.

`self.name` and `self.age` store data on the object, while local variables (`age = age`) are discarded after the function ends

correct execution:

```
Ai coding > assignment 7.4 > task5.py > u
1 class User:
2     def __init__(self, name, age):
3         self.name = name
4         self.age = age
5
6     def greet(self):
7         return "Hello, my name is " + self.name
8
9
10 # Attempt to create and use the object
11 u = User("Alice", 30)
12 print(u.name)
13 print(u.greet())
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
clg@Aruhs-MacBook-Air ~ % /usr/bin/python3 "/Users/clg/Myh/sru/sru 3rd 2nd sem/Ai coding/assignment 7.4/task5.py"
Alice
Hello, my name is Alice
```