

Assignment-7.4

Task01: 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:
 - The missing or incorrect base case
 - The corrected recursive logic

Sample input/output demonstrating correct execution

```

Task1.py
-----
1 Fixed recursive factorial implementation with proper base case,
2 input validation, and explanatory comments.
3
4 Original Faulty implementation:
5     def factorial(n):
6         return n * factorial(n-1)
7
8 Problems fixed:
9 - Missing base case (caused infinite recursion / RecursionError)
10 - No input validation for negative or non-integer inputs
11
12 This file provides a corrected `factorial` function and a small
13 sample runner that prints outputs for demonstration.
14 """
15
16 def factorial(n):
17     """Return n! using recursion.
18
19     Corrections and notes:
20     - Base Case: when n <= 1 return 1 (covers 0! and 1!).
21     - Input validation: only non-negative integers are accepted.
22     - Recursive Case uses n * factorial(n-1) which is correct once
23     the base case is present.
24
25     Args:
26         n (int): non-negative integer whose factorial to compute.
27
28     Returns:
29         int: factorial of n.
30
31     Raises:
32         TypeError: if n is not an integer.
33         ValueError: if n is negative.
34
35     # Ensure the input is an integer
36     if not isinstance(n, int):
37         raise TypeError("factorial() only accepts integers")
38
39     # Factorial is not defined for negative numbers
40     if n < 0:
41         raise ValueError("factorial() not defined for negative values")
42
43     # Base case: 0! = 1 and 1! = 1
44     if n <= 1:
45         return 1

```

CHAT

DEBUGGING A FAULTY PYTHON FACTORIAL FUNCTION

- Missing/incorrect base condition
- Incorrect recursive logic
- AI Debugging Explanation
 - How the issue was detected
 - Why it caused crash/incorrect output
- Corrections Applied
- Sample Input/Output

Organized tasks and implemented guidelines

I'll create a short TODO plan for the multi-step fix.

Created 5 todos

I'll open Task1.py to inspect the current code before editing.

Updated code, documentation, and TODO list

Now I'll patch Task1.py with the corrected recursive factorial, comments, and a small test runner.

Revising tool call

Now I'll apply the patch containing the corrected function and test runner.

Edited Task1.py +58 -1

Formatted corrected code and planned updates

I'll run Task1.py to verify the sample outputs now.

Run push command?

python "c:\Users\tloke\OneDrive\AI\Task1.py"

Allow Skip

Fix function (2/5)

Task1.py

Describe what to build next

Agent Auto

OUTPUT:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Open file in editor (ctrl + click)
PS C:\Users\tloke\OneDrive\AI> & C:/Users/tloke/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/tloke/OneDrive/AI/Task1.py
5! = 120
0! = 1
1! = 1
7! = 5040
PS C:\Users\tloke\OneDrive\AI>

```

TASK-02:

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 Type Error 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

Task1.py

```
task1.py
```

task2.py

task2.py > ...

1 *****

2 Robust sorting helper that handles mixed data types safely.

3

4 Original faulty implementation:

5 def sort_values(data):

6 return sorted(data)

7

8 Problem: When 'data' contains items with types that are not directly comparable (e.g., int and str), Python raises a 'TypeError' while trying to compare elements during sorting.

9

10 This file provides a safer 'sort_values' implementation that defines a deterministic key for sorting mixed types so the function will never raise a TypeError for common Python built-ins.

11 *****

12

13 from typing import Iterable, List, Any, Tuple

14

15

16

17 def _sort_key(x: Any) -> Tuple[int, Any]:

18 """Return a tuple key used to compare mixed-type values safely.

19

20 Key structure (type_priority, comparable_value) where:

21 - type_priority: integer grouping by broad type (numbers first, then strings, then other types). Lower priority sorts earlier.

22 - comparable_value: a value of a consistent, comparable type used for ordering within the group.

23

24 This avoids direct comparisons between incompatible types like int and str which raise TypeError in Python 3.

25 *****

26

27 # Numbers (int, float) - treat all as floats for ordering

28 if isinstance(x, (int, float)):

29 return (0, float(x))

30

31

32 # Strings: attempt to detect numeric strings and treat them as numbers

33 if isinstance(x, str):

34 # Try numeric conversion (so '10' sorts before '2' numerically)

35 try:

36 return (0, float(x))

37 except ValueError:

38 # Non-numeric strings sort after numbers; use lowercase for

39 # case-insensitive lexicographic ordering

40 return (1, x.lower())

41

42

43

44

45

CHAT

DEBUGGING A FAULTY PYTHON FACTORIAL FUNCTION

3. Add comments in the code explaining the fix.

6. Document the debugging steps followed by the AI.

7. Explain how the fix improves robustness and reliability.

8. Provide sample input/output showing successful sorting without runtime errors.

Faulty code:

```
def sort_values(data):  
    return sorted(data)
```

Expected Output Format:

1. Corrected Python Code

2. Error Explanation

- Root cause of TypeError
- Why mixed types cause failure

3. AI Debugging Process

- How the AI detected the issue
- How the solution was chosen

4. Fix Applied

- Type filtering OR type conversion logic

5. Robustness Improvement Explanation

6. Sample Input/Output Demonstration

Planned Todo List tasks and updated Task2.py

I'll add a short TODO, inspect Task2.py, then patch it to handle mixed-type lists safely.

Managed and tracked todo items

Progress: Task2.py is empty. Next I'll write a robust sort_values implementation and add demo tests.

Edited Task2.py - 91 - 1

Implement safe sort (2/5)

1 file changed +91 -1

Keep Undo

Describe what to build next

Agent Auto

OUTPUT:

```
Demo 1: input = [3, '1', 2, '10', 'apple', 'Banana', 0.5]
          sorted = [0.5, '1', 2, 3, '10', 'apple', 'Banana']

Demo 2: input = ['a', 'B', 'c', 2, 1, '10']
          sorted = [1, 2, '10', 'a', 'B', 'c']

Demo 3: input = [None, 5, '3', {'k': 'v'}, (2,)]
          sorted = ['3', 5, (2,), None, {'k': 'v'}]

Demo 4: input = ['100', '20', 3, 4.5, 'abc']
          sorted = [3, 4.5, '20', '100', 'abc']

PS C:\Users\tloke\OneDrive\AI>
```

Task-03: 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

The screenshot shows the GitHub Copilot interface during the refactoring of a faulty Python factorial function. The left side displays the original code with a warning message: "Problem: the file was opened but never closed. This can lead to resource leakage (open file descriptors), delayed writes on some platforms, and exhausted file handles in long-running programs." The right side provides an AI-generated explanation and steps to fix the issue:

- AI Generated Why this approach can cause performance and reliability issues.
- Refactor the code using best practices (use a context manager with `with open()`).
- Ensure safe, clean, and reliable file handling.
- Add comments explaining the fix.
- Briefly describe why the revised approach is better.
- Ensure clean execution without warnings or errors.
- Document how the AI detected the issue and what changes were made.

The faulty code is shown as:

```
file = open("data.txt", "r")
data = file.read()
print(data)
```

The expected output format is listed as:

1. Refactored Python Code (using with open)
2. Problem Identification
3. AI Debugging Explanation
4. Fix Applied
5. Why the Revised Approach is Better
6. Clean Execution Explanation

The AI also suggests a plan:

- ✓ Planned and structured file handling process
- I'll update Task3.py to use a context manager, create a demo data.txt, and run it to verify output.

Tool status:

- ✓ Managed and tracked todo items for planning
- ✓ Edited Task3.py +58 -1
- Run push command?
- python "c:\Users\tloke\OneDrive\AI\Task3.py"

Refactor progress:

- > Refactor to context manager (2/5)
- Describe what to build next
- Agent Auto

Bottom status bar:

Ln 58, Col 1 Spaces: 4 UTF-8 CRLF {} python 3.14.2 ⚡ Go Live

```
data.txt
1 Sample data file for Task3 demo.
2 Line 2: Hello, world!
3
```

OUTPUT:

```
PS C:\Users\tloke\OneDrive\AI> & C:/Users/tloke/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/tloke/OneDrive/AI/Task3.py
Sample data file for Task3 demo.
Line 2: Hello, world!
Open file in editor (ctrl + click)
PS C:\Users\tloke\OneDrive\AI> & C:/Users/tloke/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/tloke/OneDrive/AI/Task3.py
Sample data file for Task3 demo.
Line 2: Hello, world!
```

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

The screenshot shows the GitHub Copilot AI interface. On the left, the Python code for Task4.py is displayed. The code defines a function `safe_divide` that takes an iterable of floats and returns a list of strings. It handles division by zero and non-numeric inputs. The AI has provided a detailed explanation of the code's behavior and suggested improvements. On the right, a sidebar titled "DEBUGGING A FAULTY PYTHON FACTORIAL FUNCTION" lists 8 steps for debugging. Below it, "Faulty code" and "Expected Output Format" are shown, along with a "Planning robust code creation" section and a "Managed and tracked todo items" section. A todo item for "Add exception handling" is highlighted.

```

Task4.py > ...
1 """
2 Safe division loop with ZeroDivisionError handling.
3
4 Original faulty snippet:
5 values = [10, 5, 0, 2, 0, 1]
6 for v in values:
7     result = 100 / v
8     print("Result:", result)
9
10 Problem: when `v` is 0 the expression `100 / v` raises `ZeroDivisionError`
11 and the program crashes. The code below adds try/except to handle this
12 case gracefully and continue processing remaining values.
13 """
14
15 from typing import Iterable, List
16
17
18 def safe_divide(values: Iterable[float]) -> List[str]:
19     """Divide 100 by each value in 'values', handling division-by-zero.
20
21     For each 'v' in 'values', attempt '100 / v'. If 'v' is zero, catch
22     'ZeroDivisionError', log a meaningful message, and continue. The
23     function returns a list of result messages (strings) so callers can
24     inspect outputs programmatically.
25
26     Args:
27         values: Iterable of numeric values.
28
29     Returns:
30         List of string messages describing each result or error.
31     """
32     results: List[str] = []
33     for idx, v in enumerate(values):
34         try:
35             # Potential ZeroDivisionError occurs here when v == 0
36             res = 100 / v
37         except ZeroDivisionError:
38             # Handle the error and continue processing remaining items
39             msg = f"Index {idx}: cannot divide by zero (value={v})"
40             print(msg)
41             results.append(msg)
42             continue
43         except TypeError as exc:
44             # Handle non-numeric inputs gracefully
45             msg = f"Index {idx}: invalid operand {v!r} ({exc})"
46             print(msg)
47             results.append(msg)
48
49     return results
50
51
52 if __name__ == "__main__":
53     print(safe_divide([10, 5, 0, 2, 0, 1]))

```

OUTPUT:

```

PS C:\Users\tloke\OneDrive\AI> & C:/Users/tloke/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/tloke/OneDrive/AI/Task4.py
Result: 10.0
Result: 20.0
Index 2: cannot divide by zero (value=0)
Result: 50.0
Index 4: cannot divide by zero (value=0)
Result: 100.0
PS C:\Users\tloke\OneDrive\AI>

```

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

The screenshot shows a code editor with a Python file named `Task5.py`. The code defines a `Student` class with `__init__` and `display` methods. The code is annotated with comments explaining the fixes applied, such as adding `self` to `__init__` and using `self.name` and `self.age` in `display`.

Faulty code:

```
class Student:  
    def __init__(name, age):  
        name = name  
        age = age  
  
    def display():  
        print("Name:", name)  
        print("Age:", age)
```

Problems fixed:

- Missing 'self' parameter in `__init__` and methods.
- Assignments inside `__init__` did not set instance attributes.
- Method `display` did not accept 'self' and referenced undefined names.

This module provides a corrected "Student" class with comments and a small demo showing object creation and method usage.

Expected Output Format:

1. Corrected Class Definition
2. Error Identification
3. AI Debugging Explanation
4. Corrections Applied
5. Reliability Improvement Explanation
6. Sample Object Creation & Method Usage
7. Sample Output

OUTPUT:

```
PS C:\Users\tloke\OneDrive\AI> & C:/Users/tloke/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/tloke/OneDrive/AI/Task5.py  
Created students:  
Student(name='Alice', age=20)  
Student(name='Bob', age=22)  
  
Display details:  
Name: Alice  
Age: 20  
Name: Bob  
Age: 22  
  
Birthday update:  
Happy birthday, Alice! You are now 21.  
Name: Alice  
Age: 21  
PS C:\Users\tloke\OneDrive\AI>
```