

**SR UNIVERSITY**

**AI ASSIST CODING**

**Lab 7: Error Debugging with AI – Systematic Approaches to Finding and Fixing Bugs**

NAME: SHAIK FAHEEM

HTNO :2503A51L39

**Lab Objectives:**

- To identify and correct syntax, logic, and runtime errors in Python programs using AI tools.
- To understand common programming bugs and AI-assisted debugging suggestions.
- To evaluate how AI explains, detects, and fixes different types of coding errors.
- To build confidence in using AI to perform structured debugging practices.

**Lab Outcomes (LOs):**

After completing this lab, students will be able to:

- Use AI tools to detect and correct syntax, logic, and runtime errors.
- Interpret AI-suggested bug fixes and explanations.
- Apply systematic debugging strategies supported by AI-generated insights.
- Refactor buggy code using responsible and reliable programming patterns.

**TASK#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

**Bugged program given to the ai:**

```
ai6.py > ...
1 # buggy_factorial.py
2
3 def factorial(n):
4     # intended: recursive factorial
5     if n == 0:
6         return 0          # BUG 1: factorial(0) should be 1
7     else:
8         return n * factorial(n) # BUG 2: calls factorial(n) - infinite recursion
9
10 print("5! =", factorial(5))
11
```

**Output of bugged program:**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam> & "C:/Users/Shaik Faheem/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Shaik Faheem/Desktop/intezam/ai6.py"
Traceback (most recent call last):
  File "c:/Users/Shaik Faheem/Desktop/intezam/ai6.py", line 10, in <module>
    print("5! =", factorial(5))
    ~~~~~
    ~~~~~

  File "c:/Users/Shaik Faheem/Desktop/intezam/ai6.py", line 8, in factorial
    return n * factorial(n) # BUG 2: calls factorial(n) - infinite recursion
    ~~~~~

  File "c:/Users/Shaik Faheem/Desktop/intezam/ai6.py", line 8, in factorial
    return n * factorial(n) # BUG 2: calls factorial(n) - infinite recursion
    ~~~~~

  File "c:/Users/Shaik Faheem/Desktop/intezam/ai6.py", line 8, in factorial
```

given prompt to the AI:

Correct the following program to avoid the bug and give an output

correct the following program to avoid the bug and get an output

given program by the AI:

```
3 def factorial(n):
7     raise TypeError("factorial() only accepts integers")
8     if n < 0:
9         raise ValueError("factorial() not defined for negative numbers")
10
11     # base cases
12     if n == 0 or n == 1:
13         return 1
14
15     # recursive step (decrease n)
16     return n * factorial(n - 1)
17
18
19 if __name__ == "__main__":
20     print("0! =", factorial(0)) # 1
21     print("1! =", factorial(1)) # 1
22     print("5! =", factorial(5)) # 120
23
```

Final output after execution:

```
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam> & "C:/Users/Shaik Faheem/AppData/Local/Programs/Python/Python312/python.exe" "c:/
OneDrive/Desktop/intezam/ai6.py"
0! = 1
1! = 1
5! = 120
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
```

**Observation:**

The buggy factorial function causes infinite recursion due to an incorrect or missing base case, which Copilot/Cursor AI can detect and fix.

**Task #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

### Bugged program given to the ai:

Welcome

dukaan.html

game.html

boot.html

students.json

ai6.py

ai6.py > ...

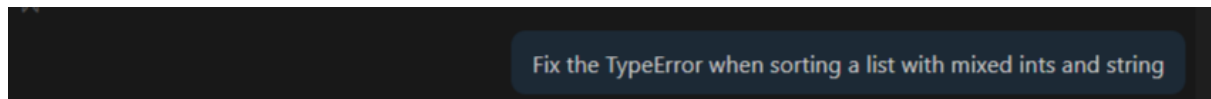
```
1 # buggy_sort.py
2 def sort_list(lst):
3     """Attempt to sort the list (may contain ints and strings)."""
4     # This will raise TypeError in Python 3 if lst contains both ints and strings
5     return sorted(lst)
6
7 if __name__ == "__main__":
8     data = [3, "1", 2, "10", "a", 5]
9     print("Before:", data)
10    print("Sorted:", sort_list(data)) # -> TypeError: '<' not supported between instances of 'str' and 'int'
11
```

### Output of the bugged program:

```
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam> & "C:/Users/Shaik Faheem/AppData/Local/Programs/Python/Python312/python.exe" OneDrive/Desktop/intezam/ai6.py
Before: [3, '1', 2, '10', 'a', 5]
Traceback (most recent call last):
  File "c:\Users\Shaik Faheem\OneDrive\Desktop\intezam\ai6.py", line 10, in <module>
    print("Sorted:", sort_list(data)) # -> TypeError: '<' not supported between instances of 'str' and 'int'
    ^^^^^^^^^^^^^^^^^
  File "c:\Users\Shaik Faheem\OneDrive\Desktop\intezam\ai6.py", line 5, in sort_list
    return sorted(lst)
    ^^^^^^^^^
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
```

**Given prompt to the ai:**

## fix the `TypeError` when sorting a list with mixed ints and strings



**Given program by the AI:**

```

1 # fixed_sort.py
2 from typing import Iterable, List, Tuple, Any
3
4 def sort_mixed_list(seq: Iterable) -> List[Any]:
5     """
6     Sort an iterable containing mixed ints/floats and strings:
7     - Numeric values (int/float and numeric-looking strings) come first, sorted by numeric value.
8     - Non-numeric strings follow, sorted alphabetically (case-insensitive).
9     Original element types are preserved in the returned list.
10    """
11
12    def key_fn(item: Any) -> Tuple[int, float or None, str]:
13        # Category 0 -> numeric (numbers and numeric strings)
14        # Category 1 -> non-numeric (compare by lowercase string)
15        if isinstance(item, (int, float)):
16            return (0, float(item), "")
17        s = str(item).strip()
18        try:
19            # try to interpret as float (works for ints, floats, "10", "2.5", "-3")
20            num = float(s)
21            return (0, num, "")
22        except ValueError:
23            return (1, None, s.lower())

```

## final output of the program:

```
OneDrive/Desktop/intezam/ai6.py
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam> & "C:/Users/Shaik Faheem/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/OneDrive/Desktop/intezam/ai6.py"
Input: [3, '1', 2, '10', 'a', 5]
Sorted: ['1', 2, 3, 5, '10', 'a']

Input: ['apple', '2', 10, '10.5', 'Banana', -1, '-2.5']
Sorted: ['-2.5', -1, '2', 10, '10.5', 'apple', 'Banana']

Input: ['3.14', 2, '02', 'abc', 2.0]
Sorted: [2, '02', 2.0, '3.14', 'abc']
```

## Observation:

Sorting raises TypeError because Python can't compare int and str; Copilot/Cursor can fix it by normalizing types (e.g., sorted(items, key=str) or converting all items to int) .

## Task#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)

## Given bugged program to the ai:

```
ai6.py > ...
1  # buggy_file_handling.py
2
3  def read_file():
4      f = open("example.txt", "r")  # File opened but never closed!
5      data = f.read()
6      return data
7
8  print(read_file())
9
```

## output of the bugged program:

```
Traceback (most recent call last):
  File "c:\Users\Shaik Faheem\OneDrive\Desktop\intezam\ai6.py", line 8, in <module>
    print(read_file())
    ~~~~~^
  File "c:\Users\Shaik Faheem\OneDrive\Desktop\intezam\ai6.py", line 4, in read_file
    f = open("example.txt", "r")  # File opened but never closed!
    ~~~~~^
FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
```

given prompt to the ai:

improve the program using the best practice (e.g., with open() block())

improve the program using the best practice (e.g., with open() block).

Given program by the ai:

```
ai6.py > ...
1 # fixed_file_handling.py
2
3 def read_file():
4     # with-statement automatically closes the file
5     with open("example.txt", "r") as f:
6         data = f.read()
7     return data
8
9 print(read_file())
10
```

Final output of the program:

```
with open('example.txt', 'r') as f:
    data = f.read()
    print(data)
FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
```

Observation:

The file is opened but never closed, which may cause resource leaks; Copilot/Cursor can fix it by using a with open() .

#### Task#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

Given bugged program to the ai:

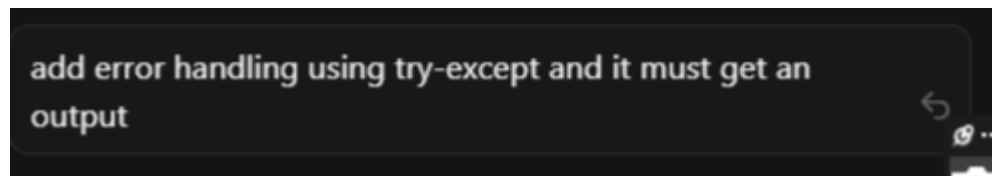
```
ai6.py > ...
1 # buggy_division.py
2
3 numbers = [10, 5, 0, 2, 0, 8]
4
5 for n in numbers:
6     result = 100 / n # ⚠ ZeroDivisionError when n = 0
7     print(f"100 / {n} = {result}")
8
```

### Output of the bugged program:

```
Traceback (most recent call last):
  File "c:\Users\Shaik Faheem\OneDrive\Desktop\intezam\ai6.py", line 6, in <module>
    result = 100 / n # ΔZeroDivisionError when n = 0
    ~~~~~
ZeroDivisionError: division by zero
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
0 Δ 0
```

### Given prompt to the ai:

Add error handling using try-except and it must get an output



### Given program by the ai:

```
ai6.py > ...
1 # fixed_division.py
2
3 numbers = [10, 5, 0, 2, 0, 8]
4
5 for n in numbers:
6     try:
7         result = 100 / n
8         print(f"100 / {n} = {result}")
9     except ZeroDivisionError:
10        print(f"Skipping division by zero for n = {n}")
11        continue
12
```

### Final output of the program:

```
100 / 10 = 10.0
100 / 5 = 20.0
Skipping division by zero for n = 0
100 / 2 = 50.0
Skipping division by zero for n = 0
100 / 8 = 12.5
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
Δ 0
```

### Observation:

The loop crashes with ZeroDivisionError when dividing by zero; Copilot/Cursor can fix it by wrapping the division in a try-except block and skipping or handling zero values to allow safe continuation.

## Task#5

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

Given bugged program to the ai:

```
ai6.py > ...
3 class Employee:
4     def __init__(name, id, salary=0):
5
6         # 1) Missing 'self' as first parameter (should be def __init__(self, ...))
7         # 2) using 'id' shadows builtin 'id' (confusing)
8         # 3) assigns undefined variables below
9         self.name = name
10        self.id = emp_id          # BUG: emp_id is undefined
11        self.salary = salary_monthly # BUG: salary_monthly is undefined
12
13    def yearly_salary(self):
14        # This will fail because self.salary was never set correctly above
15        return self.salary * 12
16
17
18 # Example use that will crash
19 e = Employee("Alice", 101, 3000)
20 print(e.yearly_salary())
21
```

Output of the bugged program:

```
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam> & "C:/Users/Shaik Faheem/AppData/Local/Programs/Python/Python312/python.exe" -i OneDrive/Desktop/intezam/ai6.py
Traceback (most recent call last):
  File "c:\Users\Shaik Faheem\OneDrive\Desktop\intezam\ai6.py", line 19, in <module>
    e = Employee("Alice", 101, 3000)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: Employee.__init__() takes from 2 to 3 positional arguments but 4 were given
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
```

Given prompt to the ai:

Analyze and correct the constructor and attribute usage and give an proper output

analyze and correct the constructor and attribute usage  
and give an proper output

Given program by the ai:

```
ai6.py / ...
3 class Employee:
28     def give_raise(self, amount: float) -> None:
29         raise ValueError("raise amount cannot be negative")
30         self.monthly_salary += amount
31
32     def __repr__(self) -> str:
33         return (f"Employee(name={self.name!r}, emp_id={self.emp_id}, "
34               f"monthly_salary={self.monthly_salary})")
35
36
37 # Example usage (works correctly)
38 if __name__ == "__main__":
39     e = Employee("Alice", 101, 3000)
40     print(e)          # Employee(name='Alice', emp_id=101, monthly_salary=3000.0)
41     print(f"Yearly: {e.yearly_salary()}") # Yearly: 36000.0
42     e.give_raise(200)
43     print(f"After raise: {e.monthly_salary}") # After raise: 3200.0
44
45
```



### Final output of the program:

```
Employee(name='Alice', emp_id=101, monthly_salary=3000.0)
Yearly: 36000.0
After raise: 3200.0
PS C:\Users\Shaik Faheem\OneDrive\Desktop\intezam>
```

Ln 41, Col 1

### Observation:

The class constructor uses wrong parameter/attribute references, causing `NameError` or `AttributeError`; Copilot/Cursor can fix it by aligning `__init__` parameters with proper self attributes and updating method references consistently.