LAB Assignment - 7

G. Yeshashwini

2303A52023

Batch - 38

Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs

Lab Objectives

• To identify and correct syntax, logic, and runtime errors in Python programs using AI tools

Week4 -

Wednesday

• 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 for 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 using AI-generated insights

• Refactor buggy code using reliable programming patterns

Task 1: Fixing Syntax Errors

Scenario

You are reviewing a Python program where a basic function definition contains a syntax error.

Requirements

• Provide a Python function add(a, b) with a missing colon

• Use an AI tool to detect the syntax error

• Allow AI to correct the function definition

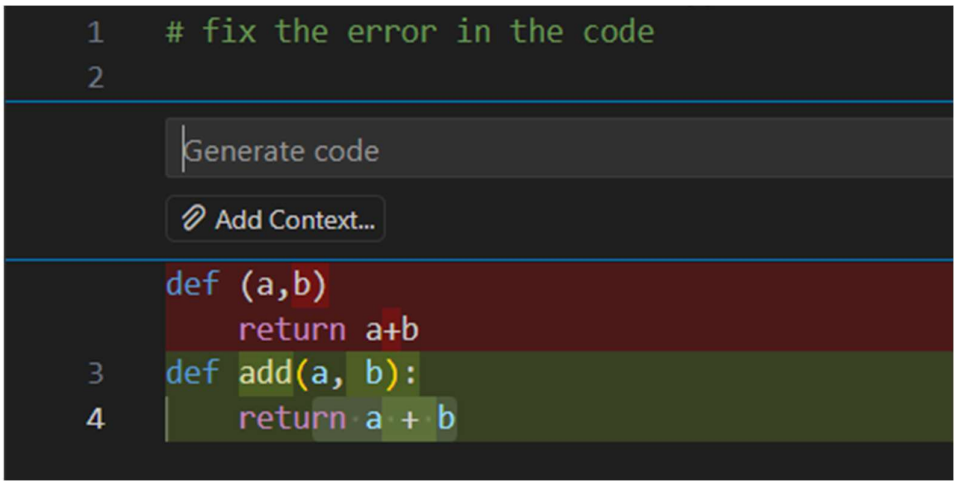
• Observe how AI explains the syntax issue

Expected Output

• Corrected function with proper syntax

• Syntax error resolved successfully

• AI-generated explanation of the fix



Explanation:
The colon : was missing after the function definition. Adding it fixes the syntax error.

Task 2: Debugging Logic Errors in Loops

Scenario

You are debugging a loop that runs infinitely due to a logical mistake.

Requirements

• Provide a loop with an increment or decrement error

• Use AI to identify the cause of infinite iteration

• Let AI fix the loop logic

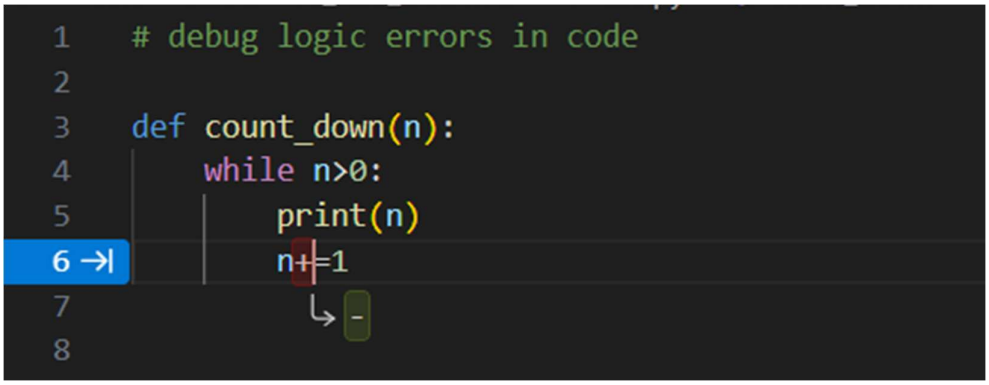
• Analyze the corrected loop behavior

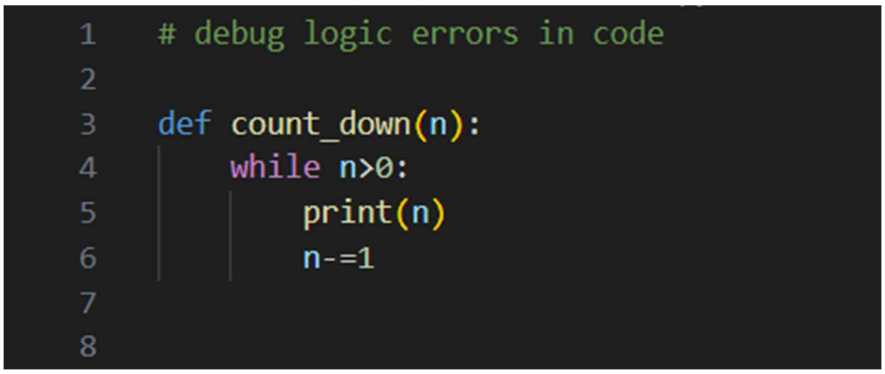
Expected Output

• Infinite loop issue resolved

• Correct increment/decrement logic applied

• AI explanation of the logic error





Explanation:
The loop was running infinitely because n was increasing (n += 1) instead of decreasing.
Changing it to n -= 1 makes n move toward 0, so the loop ends correctly.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

A Python function crashes during execution due to a division by zero error.

Requirements

• Provide a function that performs division without validation

• Use AI to identify the runtime error

• Let AI add try-except blocks for safe execution

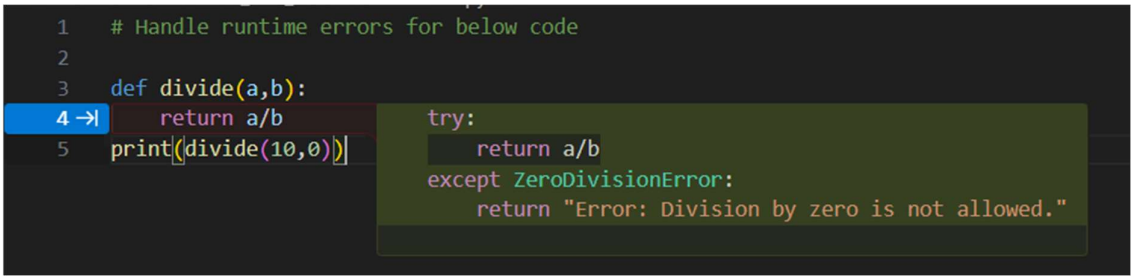
• Review AI's error-handling approach

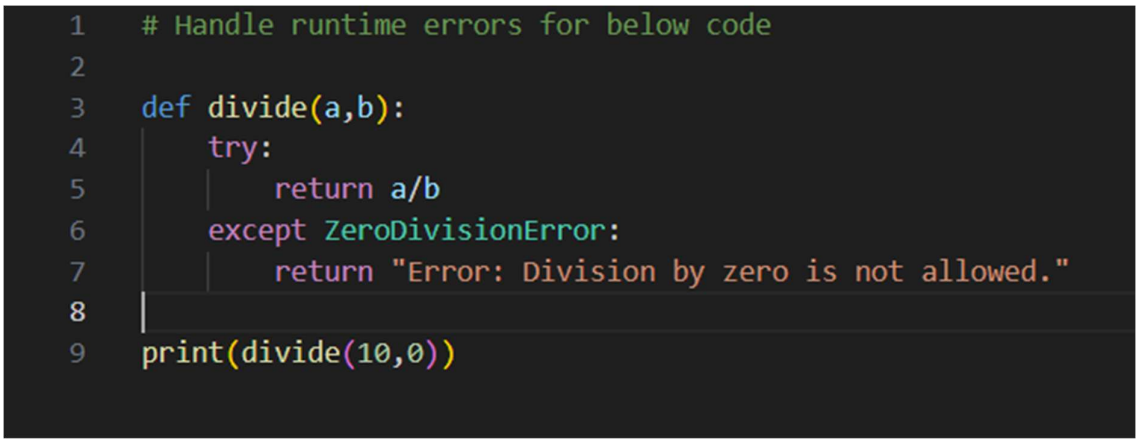
Expected Output

• Function executes safely without crashing

• Division by zero handled using try-except

• Clear AI-generated explanation of runtime error handling





Explanation:

The runtime error occurs when dividing by zero.

Using a try-except block catches the ZeroDivisionError and prevents the program from crashing.

Task 4: Debugging Class Definition Errors

Scenario

You are given a faulty Python class where the constructor is incorrectly defined.

Requirements

• Provide a class definition with missing self-parameter

• Use AI to identify the issue in the __init__() method

• Allow AI to correct the class definition

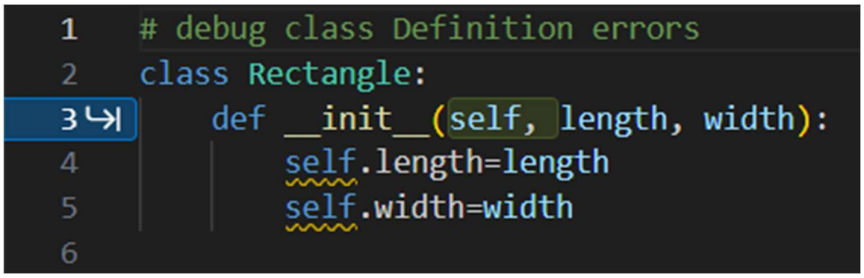
• Understand why self is required

Expected Output

• Corrected __init__() method

• Proper use of self in class definition

• AI explanation of object-oriented error



```
1   # debug class Definition errors
2   class Rectangle:
3       def __init__(self, length, width):
4           self.length=length
5           self.width=width
6
```

Explanation:
The constructor __init__ was missing the self parameter.
self refers to the current object and is required to store and access instance variables inside a class.

Task 5: Resolving Index Errors in Lists

Scenario

A program crashes when accessing an invalid index in a list.

Requirements

• Provide code that accesses an out-of-range list index

• Use AI to identify the Index Error

• Let AI suggest safe access methods

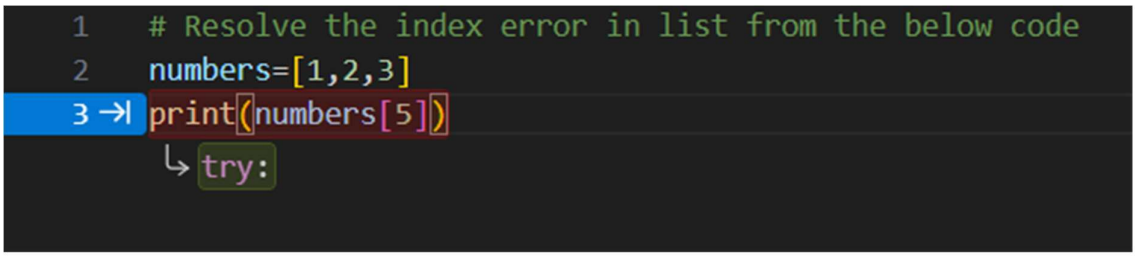
• Apply bounds checking or exception handling

Expected Output

• Index error resolved

• Safe list access logic implemented

• AI suggestion using length checks or exception handling




```
# Resolve the index error in list from the below code
numbers=[1,2,3]
try:
    print(numbers[5])
except IndexError:
    print("Index is out of bounds")

```

Explanation:

The error occurs because index 5 does not exist in the list.

Using try-except catches the Index Error and prevents the program from crashing.