ASSIGNMENT – 7.3

V.Vishnu

2303A52035

Batch – 38

Task 1: Fixing Syntax Errors

Scenario

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

```
1   def add(a, b)
2       return a + b
```

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

OUTPUT:

```
PS C:\Users\vishn> & C:/Users/vishn
  File "c:\Users\vishn\Design a sim
    def add(a, b)
                  ^
SyntaxError: expected ':'
```

You are correct. In Python, function definitions require a colon `:` at the end of the function signature. Here's the corrected code:

```
# ...existing code...
def add(a, b):
    return a + b
# ...existing code...
```

Explanation:

- The colon `:` tells Python that the following indented lines are part of the function body.
- Without the colon, you'll get a `SyntaxError`.

Tip:
Always check for colons after function, class, and control statement headers (`if`, `for`, `while`, etc.).

```
1 def add(a,b):
2     return a+b
```

Task 2: Debugging Logic Errors in Loops

Scenario

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

```
C: > Users > vishn > Design a simple ca
1   def count_down(n):
2       while n >= 0:
3           print(n)
4           n -= 1
```

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

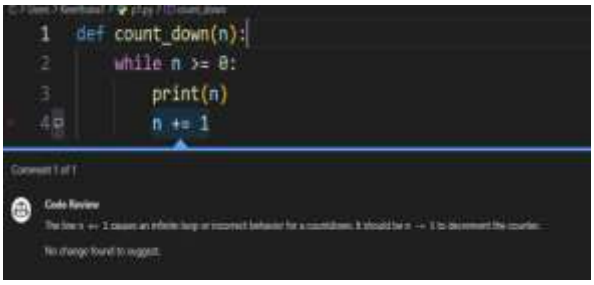
Expected Output

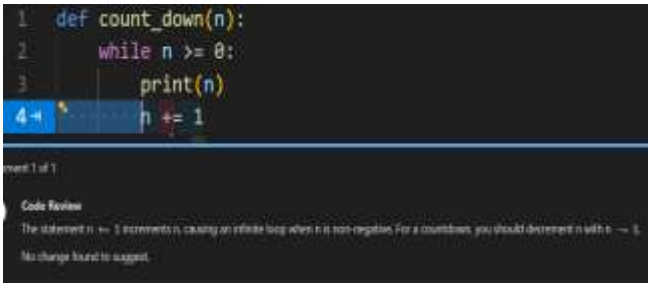
• Infinite loop issue resolved

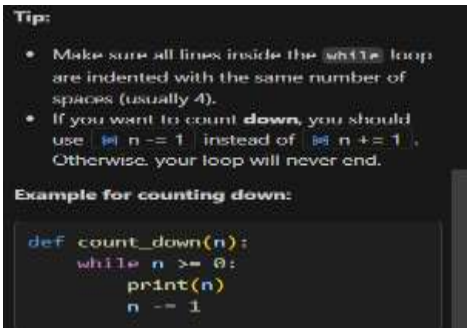
• Correct increment/decrement logic applied

• AI explanation of the logic error

OUTPUT:



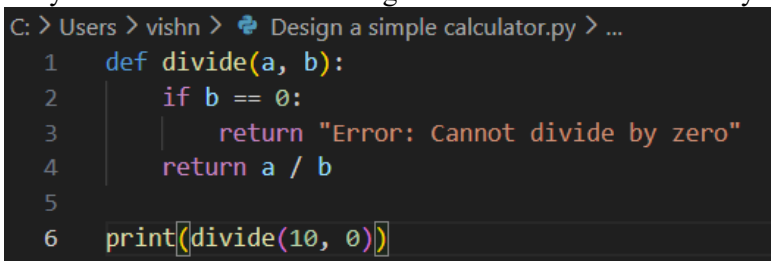




Task 3: Handling Runtime Errors (Division by Zero)

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

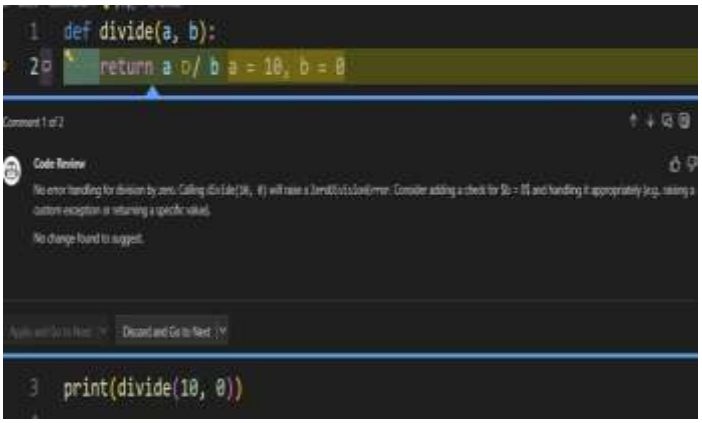
• Review AI's error-handling approach

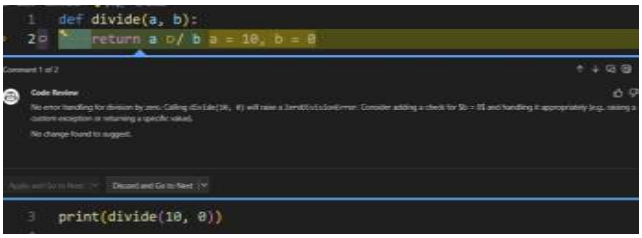
Expected Output

• Function executes safely without crashing

• Division by zero handled using try-except

• Clear AI-generated explanation of runtime error handling





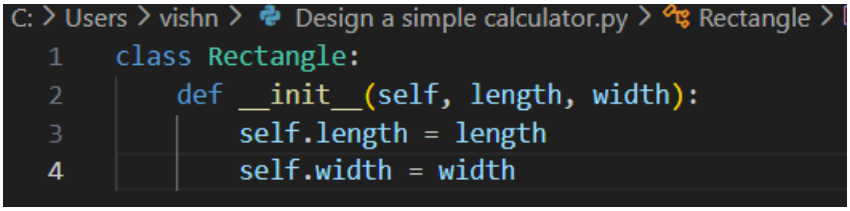
Explanation:
The runtime error occurs when dividing by zero.
Using a try-except block catches the ZeroDivisionError and prevents the program from crashing.

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

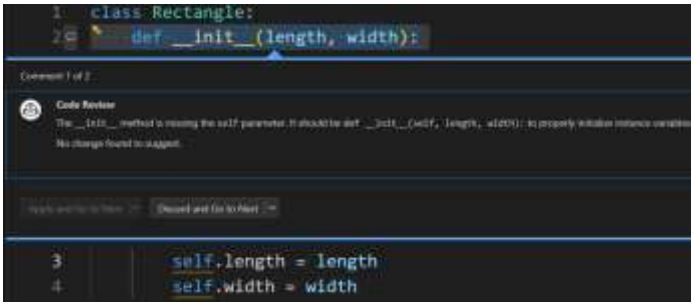
• Understand why self is required

Expected Output

• Corrected __init__() method

• Proper use of self in class definition

• AI explanation of object-oriented error



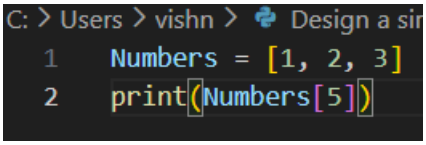
EXPLANATION:
The constructor __init__was missing the self parameter.
self refers to the current object and is required to store and access instance variables inside a class.

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

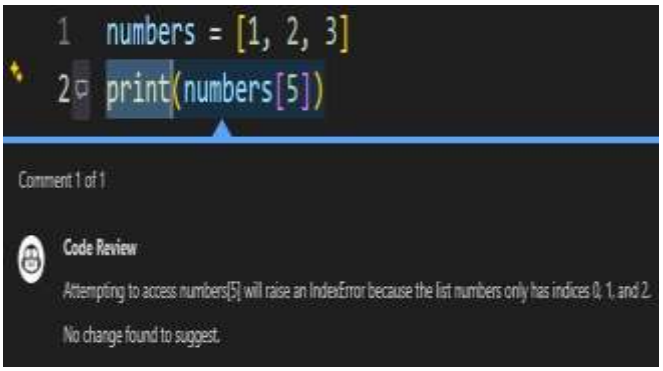
• Apply bounds checking or exception handling

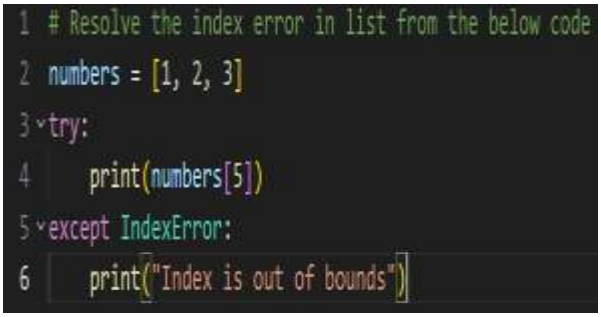
Expected Output

• Index error resolved

• Safe list access logic implemented

• AI suggestion using length checks or exception handling





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.