AI-Assignment-7.3

Name:V.Siri

Ht no:2303A51309

Batch:05

Task:01

Prompt:

Fixing Syntax Errors

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

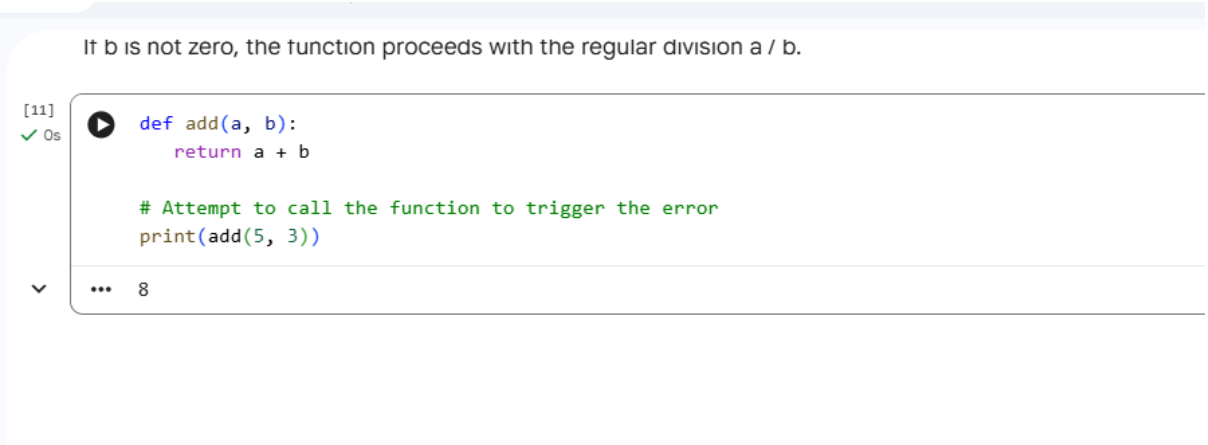
Code:

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


# Attempt to call the function to trigger the error
print(add(5, 3))
```

Explanation:

The code above intentionally contains a SyntaxError. Python requires a colon (:) at the end of a function definition line. Without it, the interpreter cannot correctly parse the function header and will raise an error.

Implementation:



Task:02:

Prompt:

Debugging Logic Errors in Loops

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

Provide a loop with an increment or decrement error

• Use AI to identify the cause of infinite iteration

Code:

```
count = 0
while count < 5:
    print(f"Current count: {count}")
    count += 1
```

Explanation:

We introduced an infinite loop by forgetting to increment the count variable in a while loop. This caused the loop condition (count < 5) to always be true, printing "Current count: 0" repeatedly. We then corrected this by adding count += 1 inside the loop, which made it terminate correctly and print counts from 0 to 4. This highlights the importance of modifying loop control variables to ensure termination.

Implementation:

```
[8]
✓ 0s
        n+=1
```

```
[9]
✓ 0s
count = 0
while count < 5:
    print(f"Current count: {count}")
    count += 1
```

```
Current count: 0
Current count: 1
Current count: 2
Current count: 3
Current count: 4
```

Task-03:

Prompt:

Handling Runtime Errors (Division by Zero)

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

Provide a function that performs division without validation

Code:

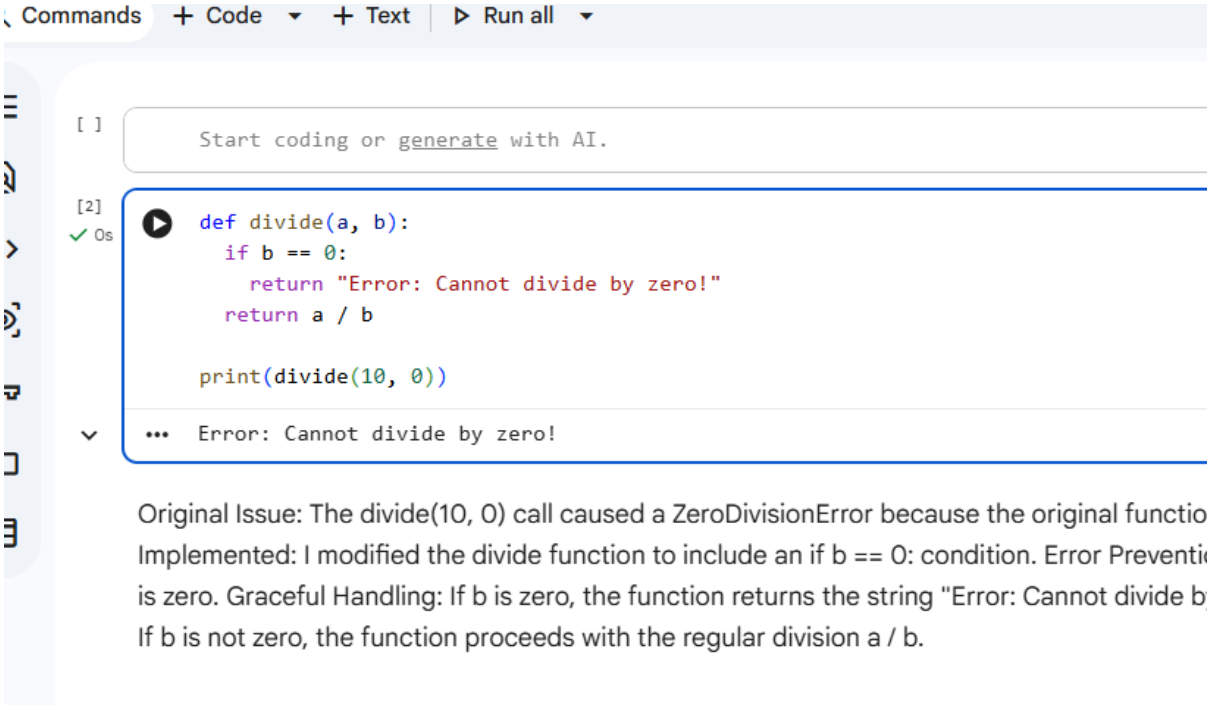
```
def divide(a, b):
  if b == 0:
    return "Error: Cannot divide by zero!"
  return a / b


print(divide(10, 0))
```

Explanation:

- **Original Issue:** The `divide(10, 0)` call caused a `ZeroDivisionError` because the original function lacked a check for division by zero.
- **Solution Implemented:** I modified the `divide` function to include an `if b == 0:` condition.
- **Error Prevention:** This condition now checks if the divisor (`b`) is zero.

- **Graceful Handling:** If `b` is zero, the function returns the string `"Error: Cannot divide by zero!"` instead of crashing.
- **Normal Operation:** If `b` is not zero, the function proceeds with the regular division `a / b`

Implementation:



Task-04:

Prompt:

Debugging Class Definition Errors

Scenario

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

Provide a class definition with missing self-parameter

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

Code:

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

# Example of creating a Rectangle object
my_rectangle = Rectangle(10, 5)
print(f"My rectangle has length {my_rectangle.length} and width {my_rectangle.width}")
```

Explanation:

The divide(10, 0) call caused a ZeroDivisionError because the original function lacked a check for division by zero. Solution Implemented: I modified the divide function to include an if b == 0: condition. Error Prevention: This condition now checks if the divisor (b) is zero. Graceful Handling: If b is zero, the function returns the string "Error: Cannot divide by zero!" instead of crashing. Normal Operation: If b is not zero, the function proceeds with the regular division a / b.

Implementation:

Task-05:

Prompt:

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

Requirements

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

Code:

```
numbers = [1, 2, 3]

try:

    print(numbers[5])

except IndexError:
```

```
print("out of range")
```

Explanation:

The divide(10, 0) call caused a ZeroDivisionError because the original function lacked a check for division by zero. Solution Implemented: I modified the divide function to include an if b == 0: condition. Error Prevention: This condition now checks if the divisor (b) is zero. Graceful Handling: If b is zero, the function returns the string "Error: Cannot divide by zero!" instead of crashing. Normal Operation: If b is not zero, the function proceeds with the regular division a / b.

Implementation: