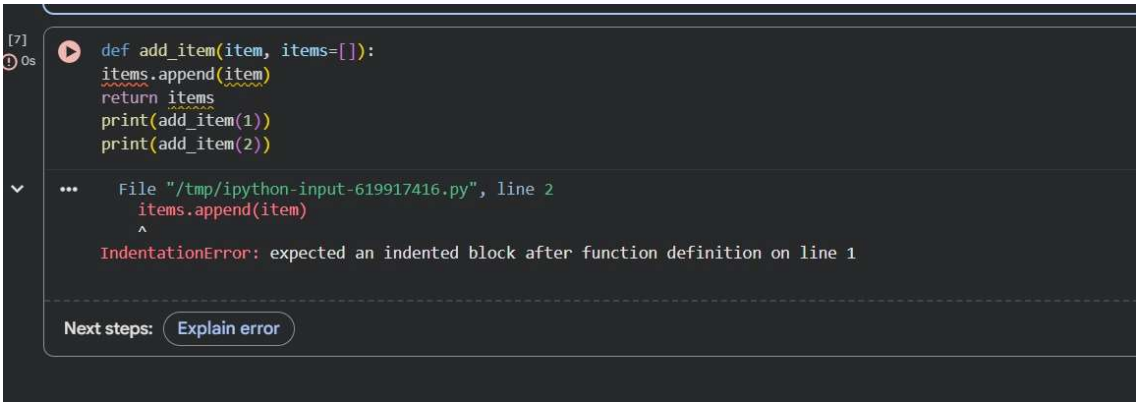
ASSIGNMENT-7.5

RISHAAK **2303A51125**

BATCH NO: 03

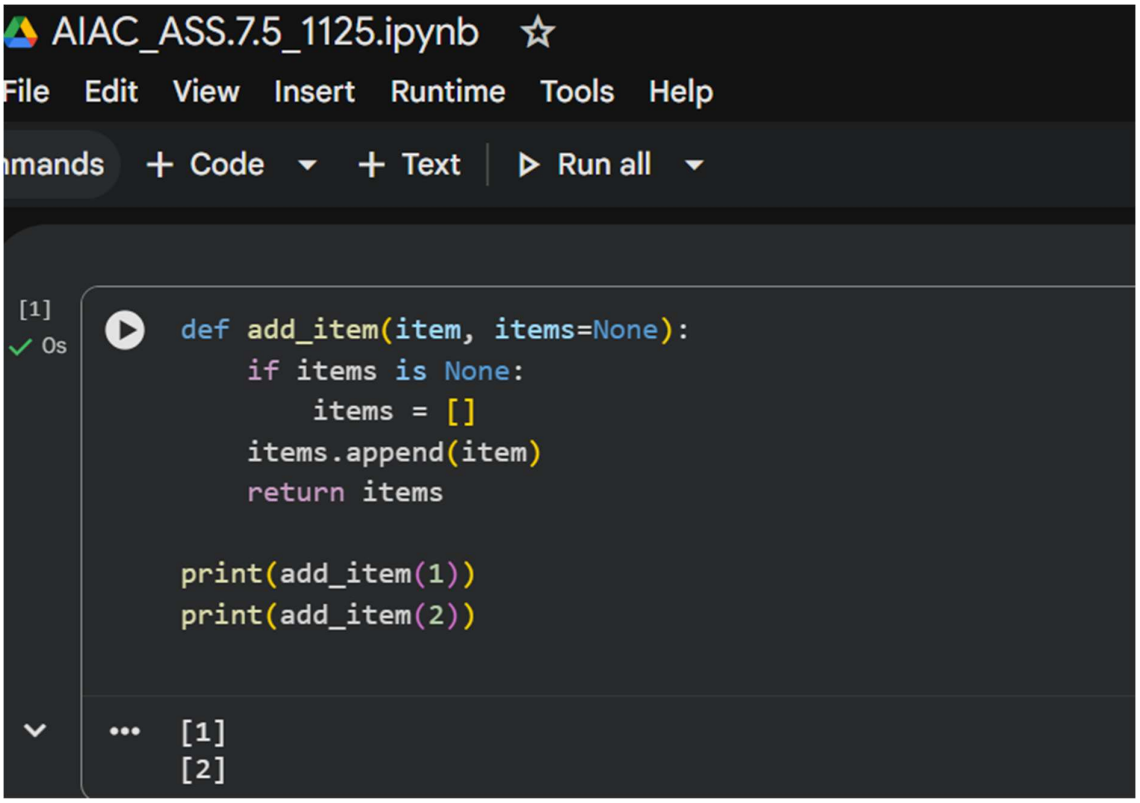
TASK 1: Mutable Default Argument – Function Bug

ACTUAL CODE:



Prompt: Analyze the Python function where a mutable default argument causes shared state between function calls. Fix the bug so each call uses a new list.

CORRECTED CODE:

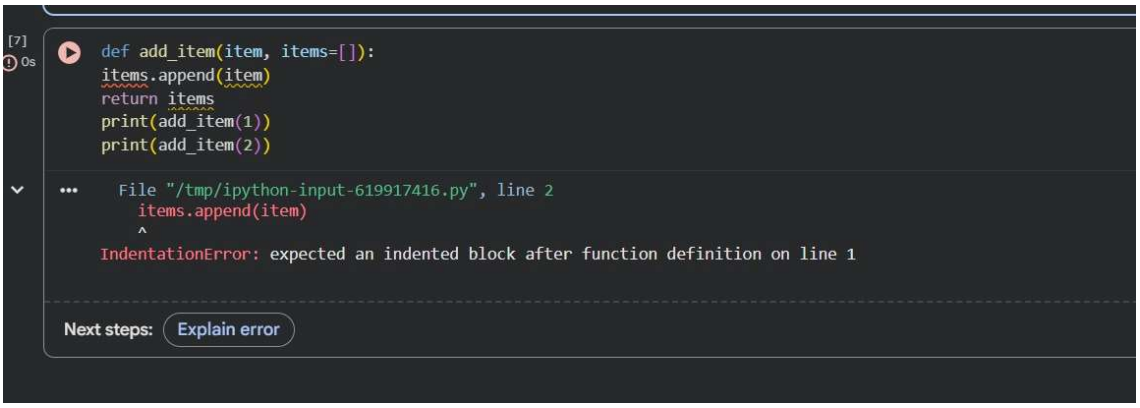


Explanation:

The issue occurs because a mutable object (list) is used as a default argument. In Python, default arguments are created once and reused across function calls, which leads to unexpected shared data. Each function call modifies the same list, causing incorrect results.

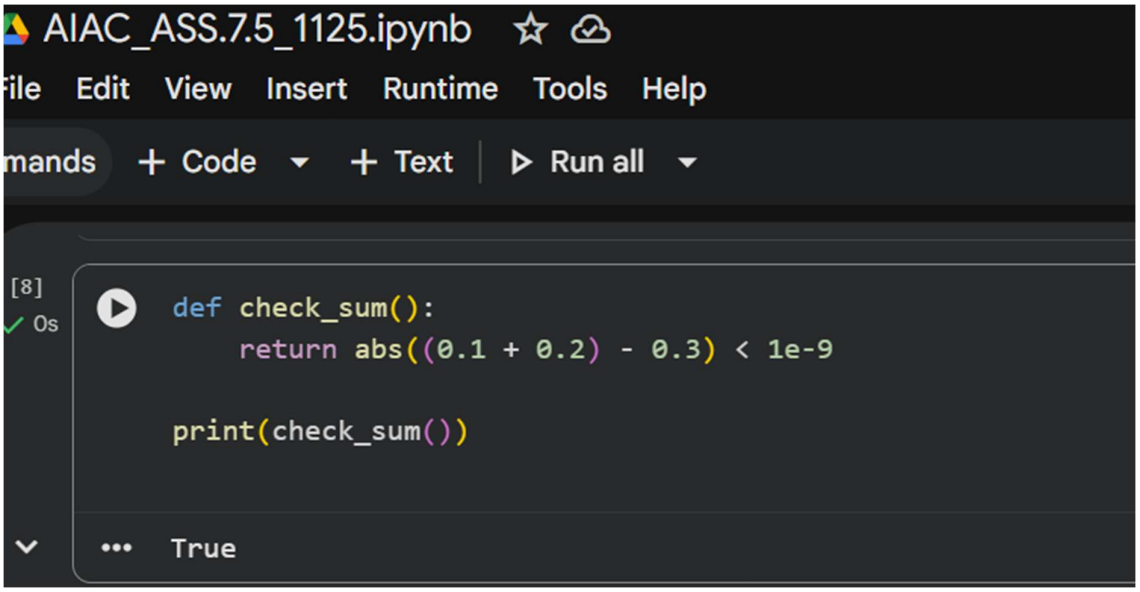
TASK 2: Floating-Point Precision Error

ACTUAL CODE:



Prompt: Identify why direct floating-point comparison fails in Python and correct the function using an appropriate tolerance-based comparison.

CORRECTED CODE:



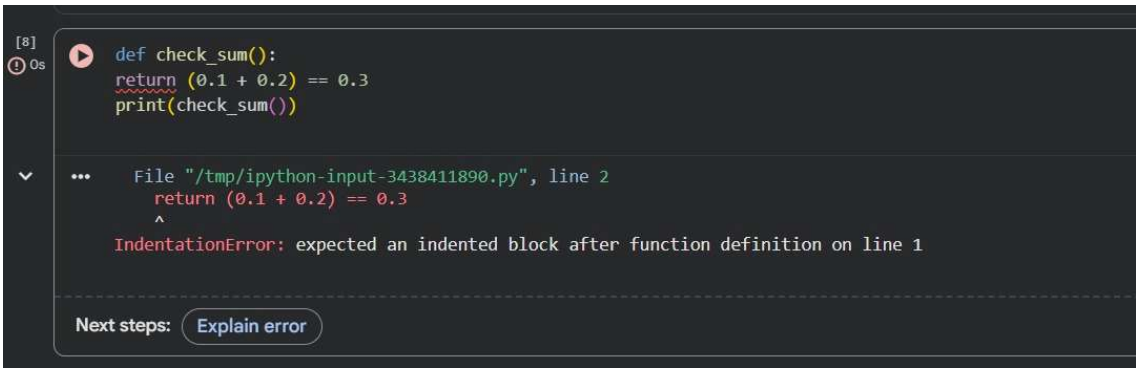
Explanation:

Floating-point numbers are stored in binary form, which can cause small precision errors. Direct equality

comparison fails because the computed result is not exactly equal to the expected value. Using a tolerance-based comparison avoids this issue.

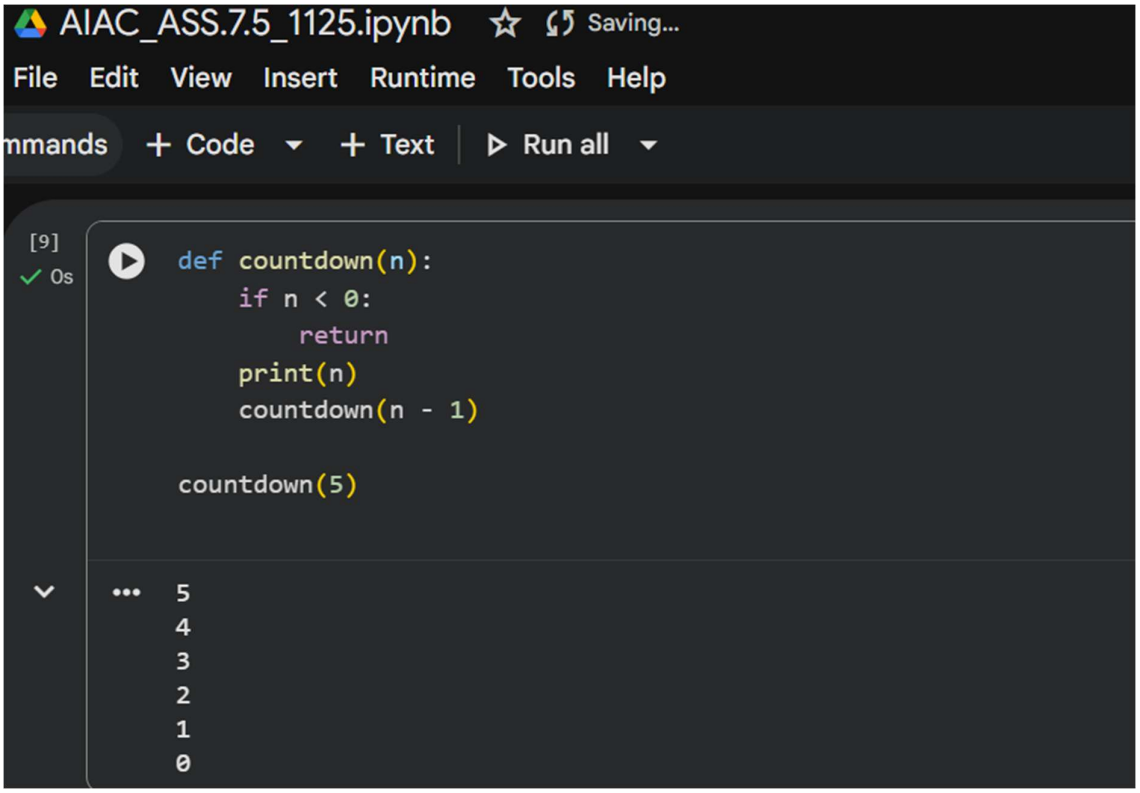
TASK 3: Recursion Error – Missing Base Case

ACTUAL CODE:



Prompt: Debug the recursive function that runs infinitely due to a missing base case. Add a proper stopping condition

CORRECTED CODE:



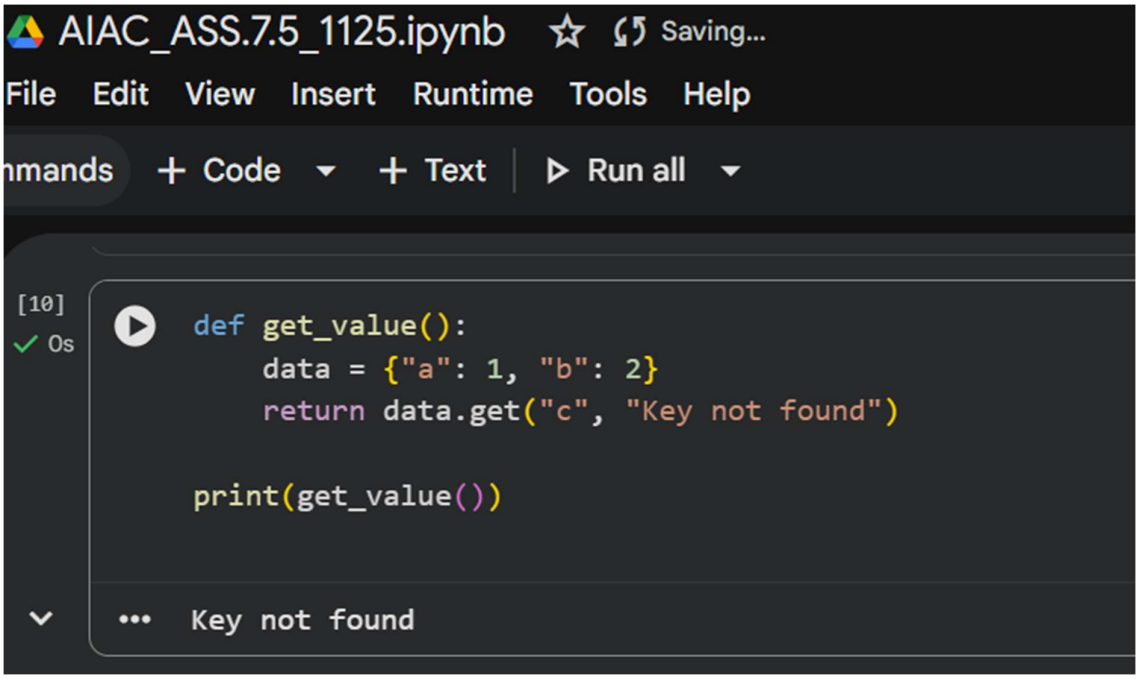
Explanation: The recursive function lacks a base case, so it keeps calling itself indefinitely. This leads to infinite recursion and eventually a runtime error. Adding a proper stopping condition ensures safe termination.

TASK 4: Dictionary Key Error

ACTUAL CODE:

Prompt: Fix the function that raises a KeyError when accessing a non-existing dictionary key by using safe access or error handling.

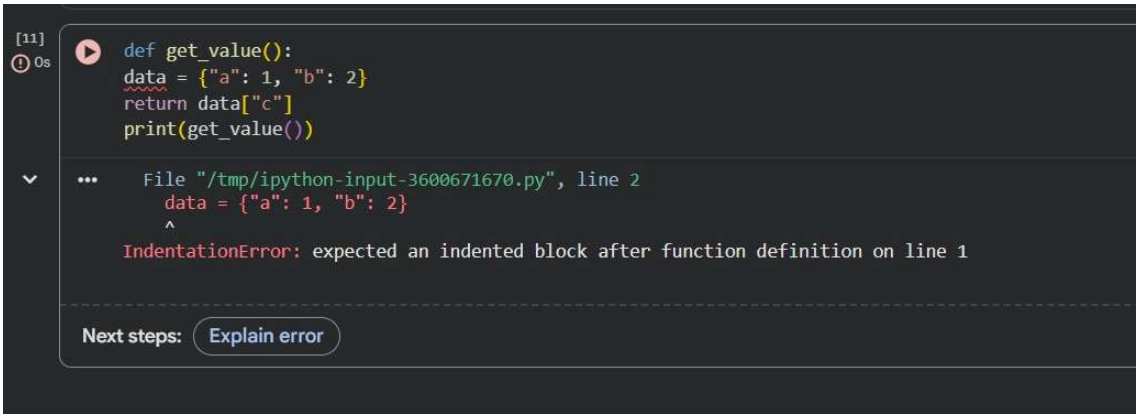
CORRECTED CODE:



Explanation: Accessing a key that does not exist in a dictionary raises a KeyError. Using safe access methods or handling missing keys prevents the program from crashing.

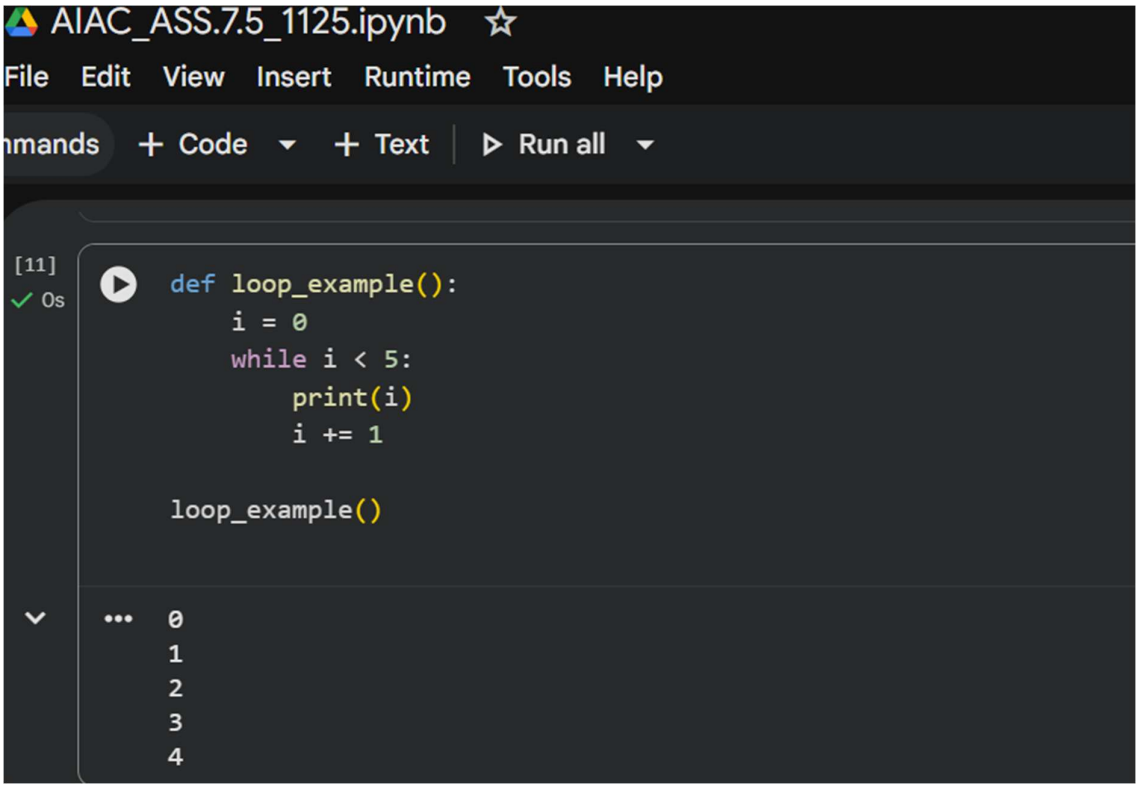
TASK 5: Infinite Loop – Wrong Condition

ACTUAL CODE:



Prompt: Detect and correct the infinite loop caused by an incorrect loop condition so the loop terminates properly.

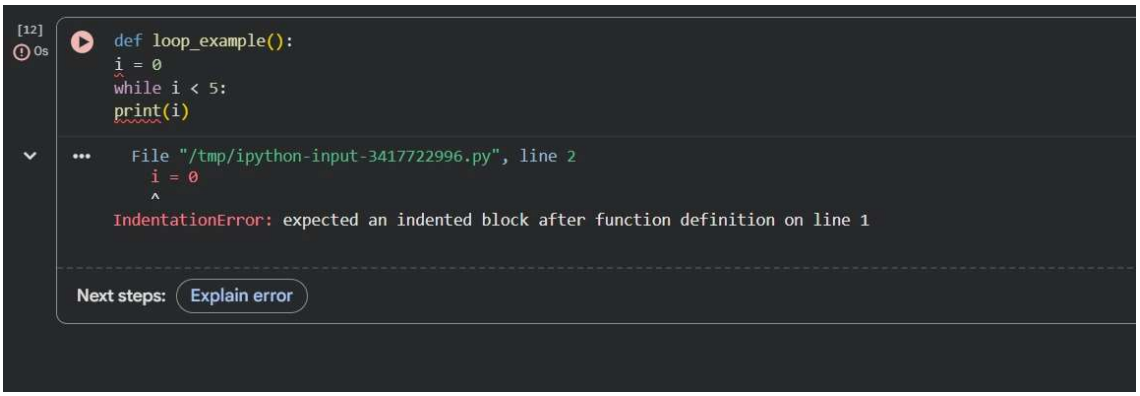
CORRECTED CODE:



Explanation: The loop condition is correct, but the loop variable is never updated. This causes the loop to run endlessly. Incrementing the loop variable allows proper termination.

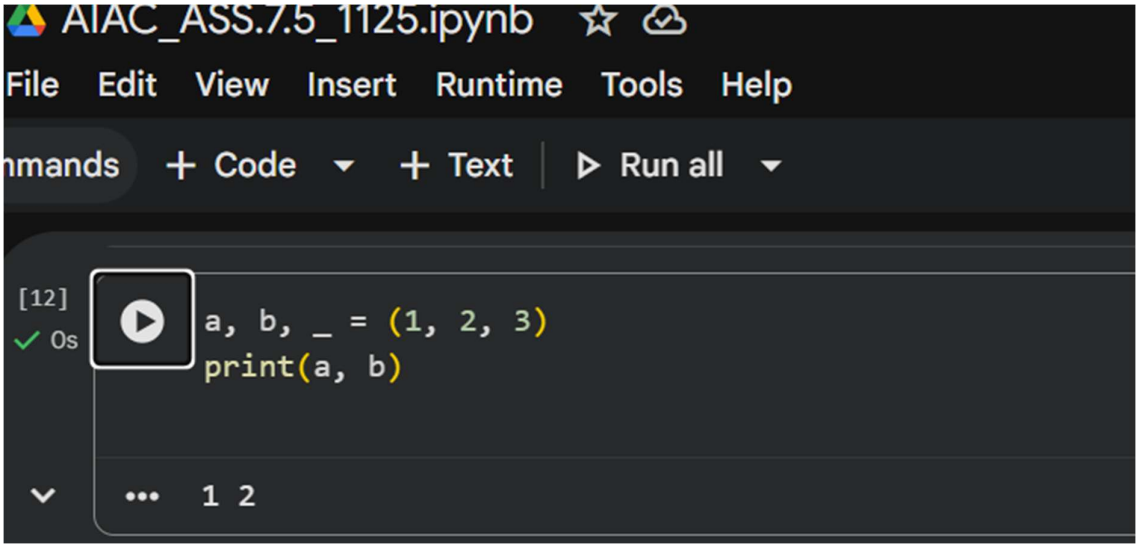
TASK 6: Unpacking Error – Wrong Variables

ACTUAL CODE:



Prompt: Analyze the tuple unpacking error caused by mismatched variables and fix it using proper unpacking.

CORRECTED CODE:



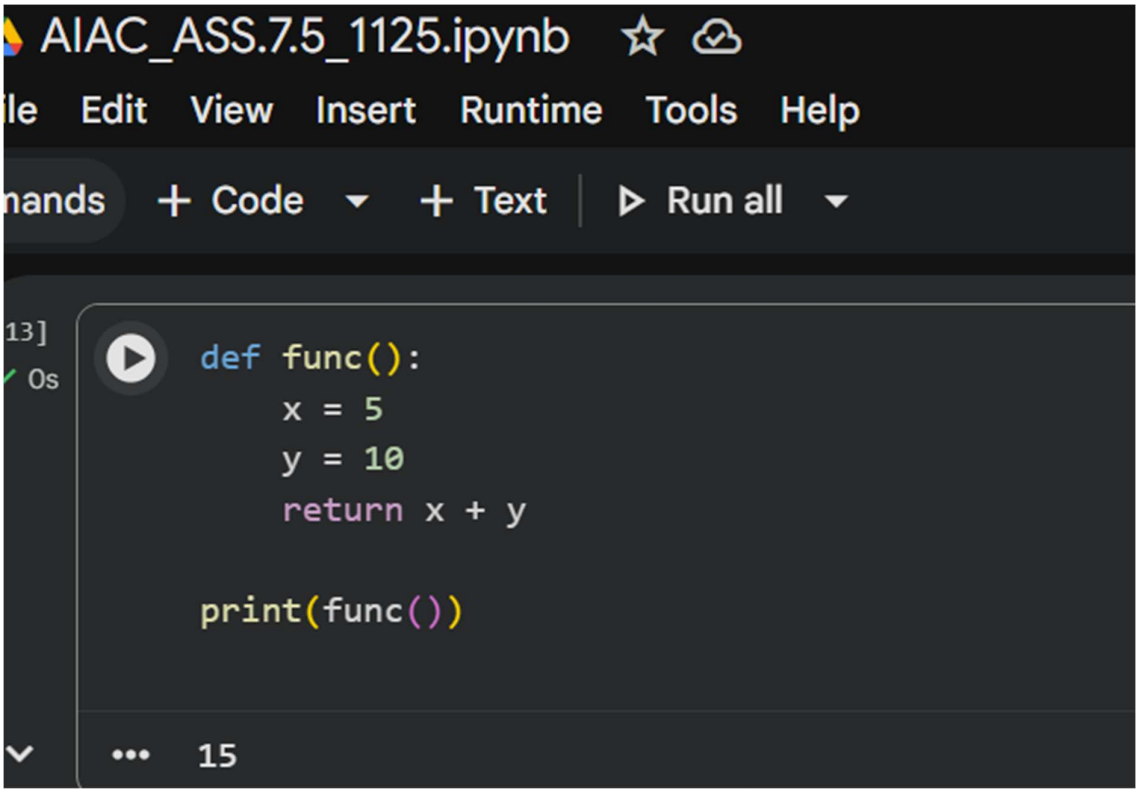
Explanation: Tuple unpacking fails when the number of variables does not match the number of values. Correct unpacking or ignoring extra values resolves the error.

TASK 7: Mixed Indentation – Tabs vs Spaces

ACTUAL CODE:

Prompt: Correct the Python function that fails due to mixed or incorrect indentation by applying consistent indentation.

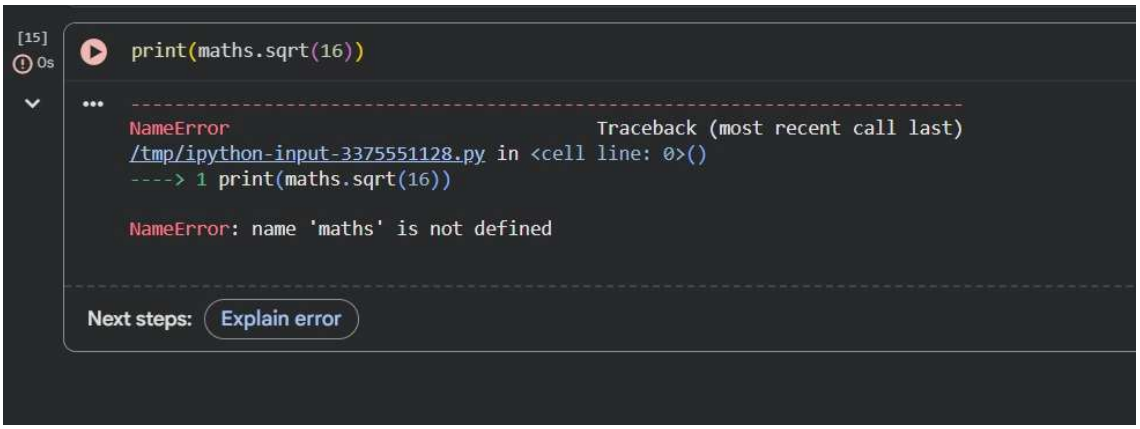
CORRECTED CODE:



Explanation: Python relies on indentation to define code blocks. Mixed or incorrect indentation causes syntax errors. Using consistent spacing fixes the issue

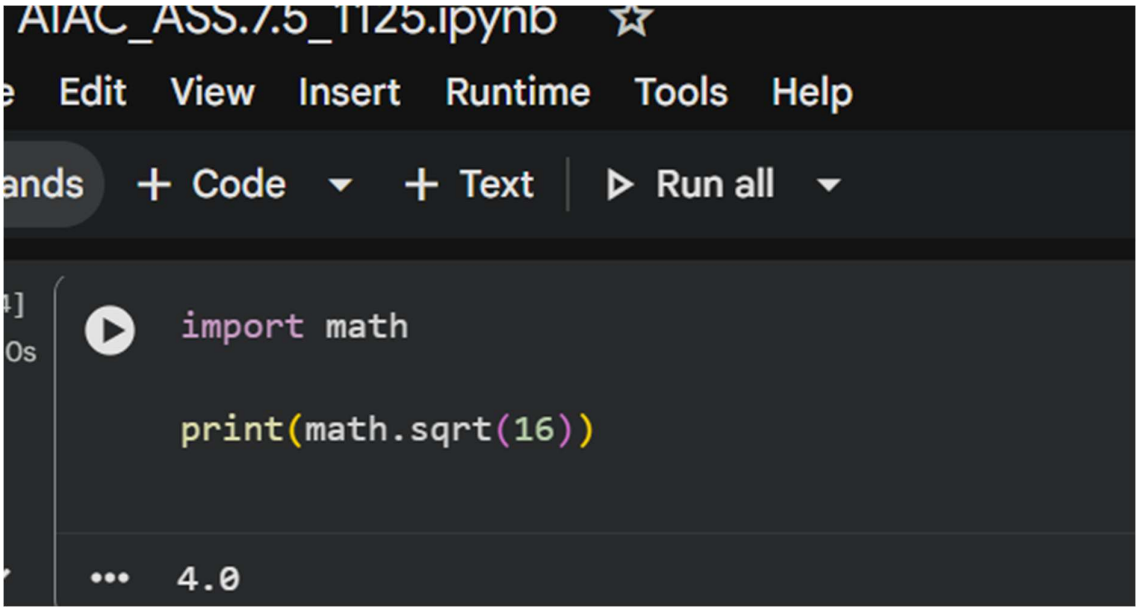
TASK 8: Import Error – Wrong Module Usage

ACTUAL CODE:



Prompt: Correct the Python function that fails due to mixed or incorrect indentation by applying consistent indentation.

CORRECTED CODE:



Explanation: The error occurs due to importing a non-existent module. Using the correct standard library module name resolves the import issue.