Lab exam 2

Prompt: generate a code in python and run the test cases for the scenario: A helper in renewable energy sometimes returns None, causing downstream type errors.

Annotate types and ensure the function always returns str or raises ValueError if not found.

Code:

def check\_membership(corpus: list[int], stream: list[int]) -> list[bool]:

    """

    Checks for the membership of elements from a stream in a corpus.

    This function optimizes the membership check by first converting the

    corpus into a set, which provides an average time complexity of O(1)

    for lookups. It then maps the stream elements to boolean values based

    on their presence in the created set.

    Args:

        corpus: A list of integers representing the large data set.

        stream: A list of integers representing the incoming data stream.

    Returns:

        A list of booleans, where each boolean corresponds to whether the

        respective element in the stream is present in the corpus.

    """

    corpus\_set = set(corpus)

    return [item in corpus\_set for item in stream]

# --- Test Cases ---

# Sample 1: Provided in the problem description

corpus1 = [1, 2, 3, 4, 5]

stream1 = [2, 5, 9]

print(f"Test Case 1: corpus={corpus1}, stream={stream1}")

print(f"Expected: [True, True, False]")

output1 = check\_membership(corpus1, stream1)

print(f"Actual:   {output1}")

print(f"Result:   {'Passed' if output1 == [True, True, False] else 'Failed'}\n")

# Sample 2: Empty stream

corpus2 = [10, 20, 30]

stream2 = []

print(f"Test Case 2: corpus={corpus2}, stream={stream2}")

print(f"Expected: []")

output2 = check\_membership(corpus2, stream2)

print(f"Actual:   {output2}")

print(f"Result:   {'Passed' if output2 == [] else 'Failed'}\n")

# Sample 3: Empty corpus

corpus3 = []

stream3 = [1, 2, 3]

print(f"Test Case 3: corpus={corpus3}, stream={stream3}")

print(f"Expected: [False, False, False]")

output3 = check\_membership(corpus3, stream3)

print(f"Actual:   {output3}")

print(f"Result:   {'Passed' if output3 == [False, False, False] else 'Failed'}\n")

# Sample 4: All members present

corpus4 = [1, 2, 3, 4, 5]

stream4 = [1, 3, 5]

print(f"Test Case 4: corpus={corpus4}, stream={stream4}")

print(f"Expected: [True, True, True]")

output4 = check\_membership(corpus4, stream4)

print(f"Actual:   {output4}")

print(f"Result:   {'Passed' if output4 == [True, True, True] else 'Failed'}\n")

# Sample 5: No members present

corpus5 = [1, 2, 3]

stream5 = [4, 5, 6]

print(f"Test Case 5: corpus={corpus5}, stream={stream5}")

print(f"Expected: [False, False, False]")

output5 = check\_membership(corpus5, stream5)

print(f"Actual:   {output5}")

print(f"Result:   {'Passed' if output5 == [False, False, False] else 'Failed'}\n")

output:

Test Case 1: corpus=[1, 2, 3, 4, 5], stream=[2, 5, 9]

Expected: [True, True, False]

Actual: [True, True, False]

Result: Passed

Test Case 2: corpus=[10, 20, 30], stream=[]

Expected: []

Actual: []

Result: Passed

Test Case 3: corpus=[], stream=[1, 2, 3]

Expected: [False, False, False]

Actual: [False, False, False]

Result: Passed

Test Case 4: corpus=[1, 2, 3, 4, 5], stream=[1, 3, 5]

Expected: [True, True, True]

Actual: [True, True, True]

Result: Passed

Test Case 5: corpus=[1, 2, 3], stream=[4, 5, 6]

Expected: [False, False, False]

Actual: [False, False, False]

Result: Passed

Prompt: generate me a python code that matches scenario: A streaming job in renewable energy checks if IDs are in a large corpus.

Optimize membership checks by converting corpus to a set once, then mapping stream to booleans.

Code:

def find\_renewable\_energy(source: str, sources\_list: List[str]) -> str:

    """

    Checks if a given renewable energy source is in a list.

    Args:

        source: The renewable energy source to find.

        sources\_list: A list of known renewable energy sources.

    Returns:

        A string confirming the source was found.

    Raises:

        ValueError: If the source is not in the list.

    """

    if source in sources\_list:

        return "Value found"

    else:

        raise ValueError("Value not found")

# Test Case (Negative Test)

# This test expects a ValueError to be raised

try:

    find\_renewable\_energy("c", ["a", "b"])

except ValueError as e:

    print(f"Caught expected error: {e}")

output:

Caught expected error: Value not found