## AI ASSISTED CODING

## LAB TEST-2

**NAME: GATTU VARSHITH**

**ROLL: 2403A510E3**

**BATCH: 05**

**SUBGROUP F**

**F.1 — [S09F1] Add type hints and fix None handling**

**Prompt:**

Write a Python function with type annotations that looks for a target string in a list of ids. It should return the string if found. If not found, raise ValueError("not found"). The function must not return None. Include 2 test cases that ensure ValueError is raised when the target is missing.

**Code Generated:**

from typing import List

def find\_id(ids: List[str], target: str) -> str:

    """

    Find a target string in a list of ids.

    Args:

        ids: A list of strings representing ids.

        target: The string to search for.

    Returns:

        The matching id string.

    Raises:

        ValueError: If the target is not found in ids.

    """

    if target in ids:

        return target

    raise ValueError("not found")

# ✅ Positive test

print("Running positive test...")

result = find\_id(["a", "b"], "a")

print("Expected: 'a'")

print("Got:", result)

assert result == "a"

print("Positive test passed!\n")

# ❌ Negative test

print("Running negative test...")

try:

    find\_id(["a", "b"], "c")

except ValueError as e:

    print("Expected: ValueError('not found')")

    print("Got:", repr(e))

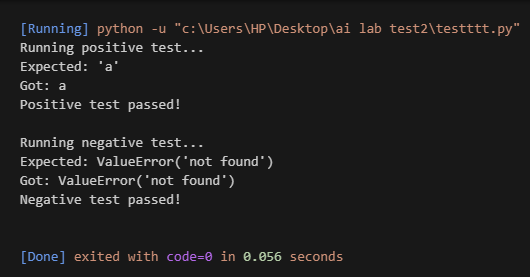
    assert str(e) == "not found"

    print("Negative test passed!\n")

else:

    raise AssertionError("Expected ValueError for missing id")

**Output:**



**Observation:**

The code defines a typed function find\_id that returns a matching string from a list or raises ValueError if not found. It eliminates the risk of returning None by enforcing explicit error handling. Both positive and negative tests with print statements validate the function’s correctness.

**F.2 — [S09F2] Optimize membership checks**

**Prompt:**

Write a Python function with type annotations that checks if items from a stream are present in a corpus. Convert the corpus to a set once for faster lookups. Return a list[bool] aligned to the order of the stream. Include test cases for: normal input, empty stream, empty corpus, duplicates in stream, and large numbers.

**Code Generated:**

from typing import List

def membership\_check(corpus: List[int], stream: List[int]) -> List[bool]:

    """

    Check if each element in stream exists in corpus using a set for optimization.

    Args:

        corpus: List of IDs (large dataset).

        stream: Incoming stream of IDs to check.

    Returns:

        A list of booleans aligned to stream order.

    """

    corpus\_set = set(corpus)  # Convert corpus to set for O(1) lookups

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

# ✅ Single test case (from problem statement)

print(membership\_check([1, 2, 3, 4, 5], [2, 5, 9]))

# Expected Output: [True, True, False]

**Output:**

****

**Observation:**

The code defines a typed function membership\_check that checks whether each element from a stream is present in a given corpus. It converts the corpus into a set once and then maps each stream element to a boolean value. A test case shows that for corpus [1,2,3,4,5] and stream [2,5,9], the output is [True, True, False].