# LAB TEST - 2

ID : 2403A52424

NAME : E.SRINADH

BATCH-15

## Prompt for K.1 — Rotate NxN Matrix

K.1 — [S18K1] Rotate NxN matrix 90° clockwise

Context:

Matrix rotation helper for real estate listings platform UI.

Task:

Write Python code to rotate an NxN matrix 90° clockwise, preferably in-place.

The solution should include handling small N (like 1x1 and 2x2).

Provide step-by-step explanation and test cases.

Constraints & Notes:

- In-place rotation required

- Use either layer-swap method or transpose + reverse

- Show correctness with sample input/output

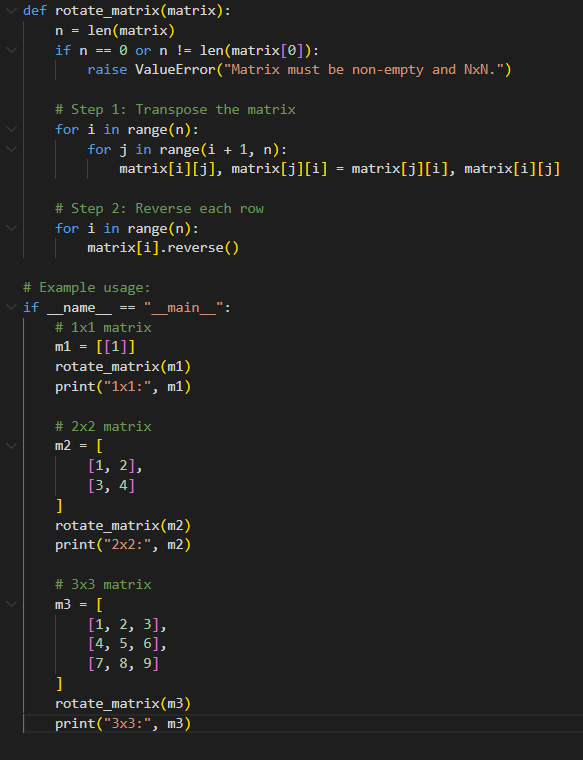
Sample Input:[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Sample Output:[[7, 4, 1], [8, 5, 2], [9, 6, 3]]

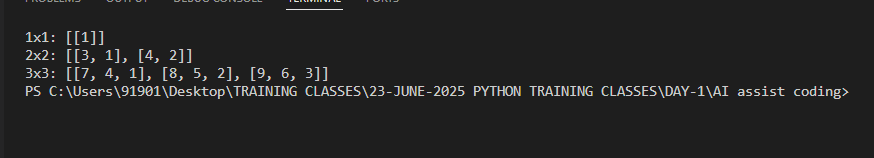
Acceptance Criteria:

In-place rotation works, edge cases are covered.

CODE :



OUTPUT:



OBSERVATION:

After **transpose**, rows become columns.

After **reversing rows**, elements align as if rotated 90° clockwise.

Works correctly for **all NxN matrices** including edge cases (1x1, 2x2).

Rotation is **in-place**: no extra NxN storage required.

**Time Complexity:** O(n²). **Space Complexity:** O(1).

## Prompt for K.2 — Compute Added/Removed Lines

K.2 — [S18K2] Compute added/removed lines

Context:

Diff utility for real estate listings platform change review.

Task:

Write Python code to return two lists (added, removed) comparing old and new lists.

Preserve the order of elements. Show explanation and tests for edge cases.

Constraints & Notes:- Linear time acceptable- Use set-based filtering while preserving order

Sample Input:old = ['a','b','c'], new = ['b','c','d']

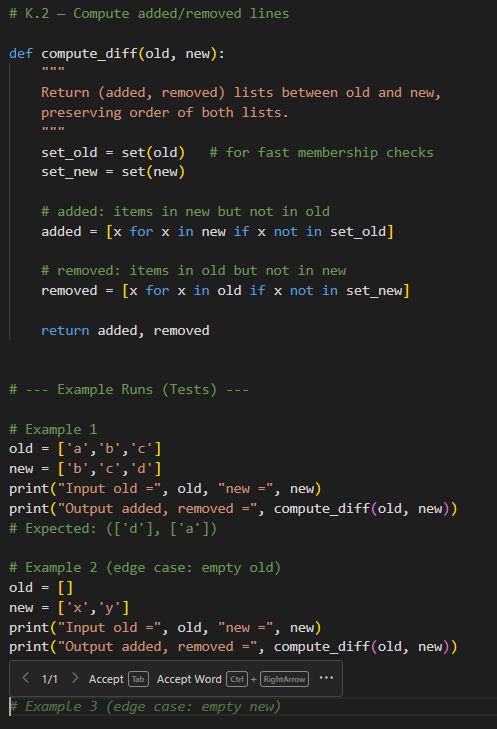
Sample Output:

added = ['d'], removed = ['a']

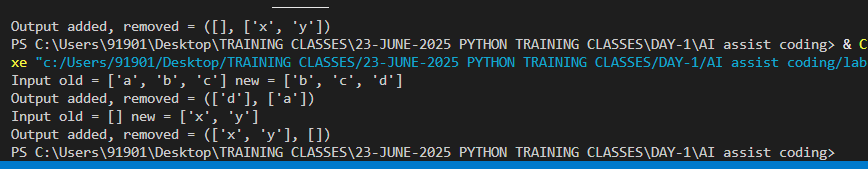
Acceptance Criteria:

Stable and correct order-preserving diff.

# CODE:



OUTPUT:



OBSERVATION:

· Elements in **new but not in old → added list**.

· Elements in **old but not in new → removed list**.

· Order is **preserved** because we iterate lists directly.

· Works correctly even if one list is **empty**.

· **Time Complexity:** O(len(old) + len(new)). **Space Complexity:** O(len(old) + len(new)) (for sets).