Code-1: Detailed Explanation

Step 1: Transpose the Matrix

```
for i in range(n):
    for j in range(i+1, n):
        matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]
```

- This nested loop swaps elements across the diagonal to transpose the matrix.
- For example:Input:
  [[1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]]

  After Transpose:
  [[1, 4, 7],
  [2, 5, 8],
  [3, 6, 9]]

Step 2: Reverse Each Row

```
for i in range(n):
    matrix[i].reverse()
```

- This loop reverses each row of the transposed matrix.
- For example:Transposed Matrix:
  [[1, 4, 7],
  [2, 5, 8],
  [3, 6, 9]]

  After Reversing Rows:
  [[7, 4, 1],
  [8, 5, 2],
  [9, 6, 3]]

Summary of Code-1

- It explicitly performs two steps:
  - Transpose the matrix.
  - Reverse each row.

- The code is verbose because it uses explicit loops to perform these operations.

Code-2: Shortened Version

Single Line Solution

matrix[:] = zip(*matrix[::-1])

This single line replaces the two-step process in **Code-1**. Let's break it down step by step:

Step 1: Reverse the Matrix (matrix[::-1])

- matrix[::-1] reverses the order of rows in the matrix.
- For example:Input:
  [[1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]]

  After Reversing Rows:
  [[7, 8, 9],
  [4, 5, 6],

  [1, 2, 3]]

Step 2: Unpack Rows with zip(...)*

- The * operator unpacks the rows of the reversed matrix as separate arguments to the zip() function.
- zip() groups the first elements of all rows together, the second elements together, and so on. This effectively transposes the reversed matrix.
- For example:Reversed Matrix:
  [[7, 8, 9],
  [4, 5, 6],
  [1, 2, 3]]

After Unpacking and Zipping:
[(7, 4, 1),
(8, 5, 2),

(9, 6, 3)]

Step 3: Assign Back to matrix[:]

- matrix[:] = ... ensures that the original matrix is modified in-place (instead of creating a new matrix).
- The tuples produced by zip() are converted back into lists (implicitly, since matrix expects a list of lists).

Final result:

```
[[7, 4, 1],
 [8, 5, 2],
 [9, 6, 3]]
```

Why Code-2 is Shorter

1. **Reversing and Transposing Together**:
    a. In **Code-1**, reversing rows and transposing are done in two separate steps.
    b. In **Code-2**, reversing (matrix[::-1]) and transposing (zip(*...)) are combined into a single operation.
2. **No Explicit Loops**:
    a. **Code-1** uses explicit loops to iterate over rows and columns.
    b. **Code-2** leverages Python's built-in functions ([::-1] and zip()) to achieve the same result concisely.
3. **In-Place Modification**:
    a. Both codes modify the matrix in-place, but **Code-2** achieves this with minimal syntax using matrix[:].

Key Points

- **Code-2** achieves the same result as **Code-1** but in a single line by combining the reverse and transpose operations.
- It uses Python's slicing ([::-1]) and unpacking (*) features to simplify the logic.
- The zip() function naturally transposes the rows into columns after reversing.

Final Answer

The transformation from **Code-1** to **Code-2** involves combining the explicit transpose and reverse operations into a single line using Python's slicing ([::-1]) and zip(*...). This makes **Code-2** concise while preserving the same functionality of rotating the matrix by 90 degrees clockwise in-place.