118. Pascal's Triangle

Easy | Topics | Companies

Given an integer `numRows`, return the first numRows of **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:

Example 1:

```
Input: numRows = 5
Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]
```

Example 2:

```
Input: numRows = 1
Output: [[1]]
```

Constraints:

- `1 <= numRows <= 30`

Python:

```
from typing import List
```

```
class Solution:
    def generate(self, numRows: int) -> List[List[int]]:
        triangle = []

        for row_num in range(numRows):
            # Start with all 1s
            row = [1] * (row_num + 1)

            # Fill the middle elements
            for j in range(1, row_num):
                row[j] = triangle[row_num - 1][j - 1] + triangle[row_num - 1][j]

            triangle.append(row)

        return triangle
```

JavaScript:

```
/**
 * @param {number} numRows
 * @return {number[][]}
 */
var generate = function(numRows) {
    let result = [];

    for (let i = 0; i < numRows; i++) {
        let row = new Array(i + 1).fill(1); // create row filled with 1s

        for (let j = 1; j < i; j++) {
            row[j] = result[i - 1][j - 1] + result[i - 1][j];
        }

        result.push(row);
    }

    return result;
};
```

Java:

```
import java.util.*;

class Solution {
    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> triangle = new ArrayList<>();
```

```
        // Base case: first row is always [1]
        if (numRows == 0) return triangle;

        triangle.add(new ArrayList<>());
        triangle.get(0).add(1);

        // Build the triangle row by row
        for (int i = 1; i < numRows; i++) {
            List<Integer> prevRow = triangle.get(i - 1);
            List<Integer> row = new ArrayList<>();

            row.add(1); // first element

            // middle elements
            for (int j = 1; j < i; j++) {
                row.add(prevRow.get(j - 1) + prevRow.get(j));
            }

            row.add(1); // last element
            triangle.add(row);
        }

        return triangle;
    }
}
```