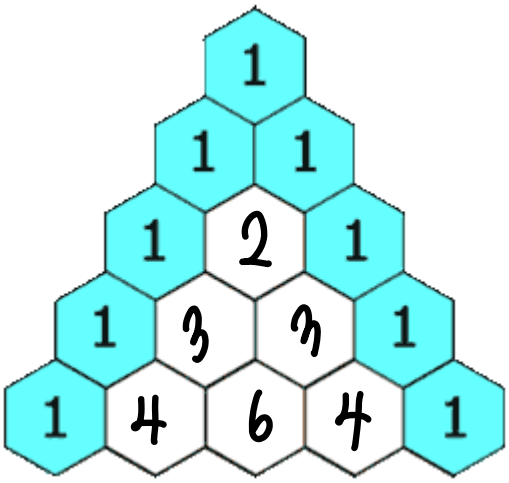
· 45분 이상 걸렸지만, 풀긴 풀음
· 내가 사용한 방법 : memorization, 2D array

Pascal's Triangle
numRows,
Return the numRows of Pascal's Triangle



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

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

for num_row in range(numRows):
Col
[
row [1] -> 1
row [1,1] -> 1,1
1. First index = 1, last index 1
2. num_row[3][1] = num_row[2][1] + num_row[2][0]
=> num_row[row][col] = num_row[row-1][col] + num_row[row-1][col-1]
row [1,2,1]
row [1,3,3,1]
1. First index = 1, last index 1
2.
num_row[4][2] = num_row[3][2] + num_row[3][1]
=> num_row[row][col] = num_row[row-1][col] + num_row[row-1][col-1]
num_row[4][2] = num_row[3][2] + num_row[3][1]
=> num_row[row][col] = num_row[row-1][col] + num_row[row-1][col-1]
[1,4,6,4,1]
]

내가 점화식을 영어로 뭐라하는지 몰라서 재귀라고 함.

Hyochan command.
1. 직관적으로 이해 가능한 변수 설정은 잘 했으나,
2. 좀 더 쉽게 설명하기 "만약 ~라면" 등.

```
        [
row 0   [1],
    1   [1,1],
        [1,2,1],
        [1,3,3,1],
        [1,4,6,4,1]
        ]


pascal_triangle = [ ] # memoize

for num_row in range(numRows):

        row = [ None for _ in range(num_row+1) ]
        row[0], row[-1] = 1, 1


        for col in range(1, len(row)-1):
                num_row[row][col] = num_row[row-1][col-1] + num_row[row-1][col]


        pascal_triangle.append(row) # [1], [1,1]

return pascal_triangle
```

이건 주성이가 hint 줌

'List comprehension' 더 외쳐야 할듯.

범위가 전체 len(row)에서

0 1 -1

len(row)-1 임을 이해...ㅍ방

Hyochan's hint.
output을 위치 맞춰.