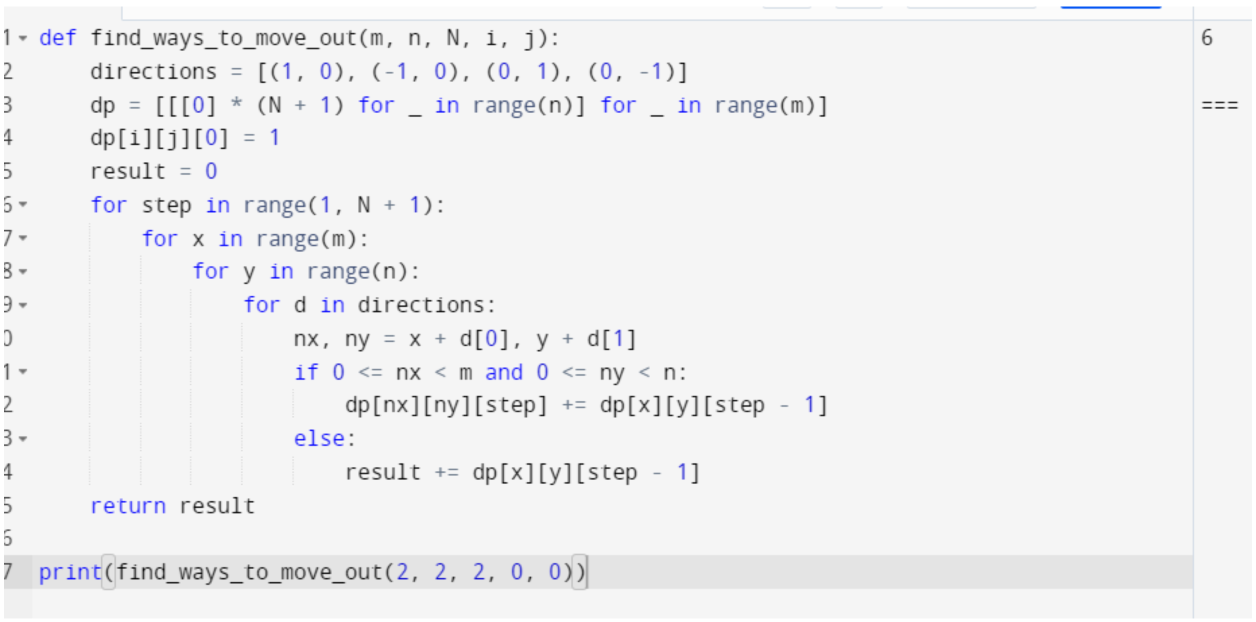
1.



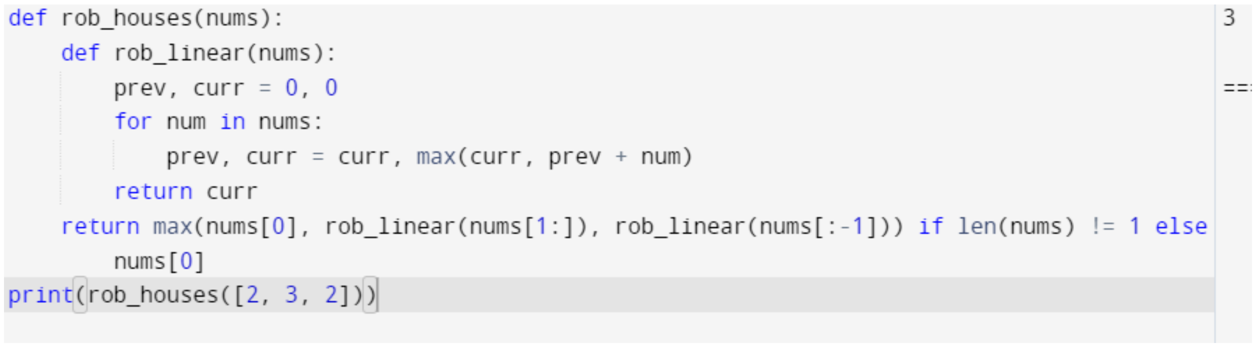
```python
def find_ways_to_move_out(m, n, N, i, j):
    directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
    dp = [[[0] * (N + 1) for _ in range(n)] for _ in range(m)]
    dp[i][j][0] = 1
    result = 0
    for step in range(1, N + 1):
        for x in range(m):
            for y in range(n):
                for d in directions:
                    nx, ny = x + d[0], y + d[1]
                    if 0 <= nx < m and 0 <= ny < n:
                        dp[nx][ny][step] += dp[x][y][step - 1]
                    else:
                        result += dp[x][y][step - 1]
    return result

print(find_ways_to_move_out(2, 2, 2, 0, 0))
```

```
6

===
```

2.

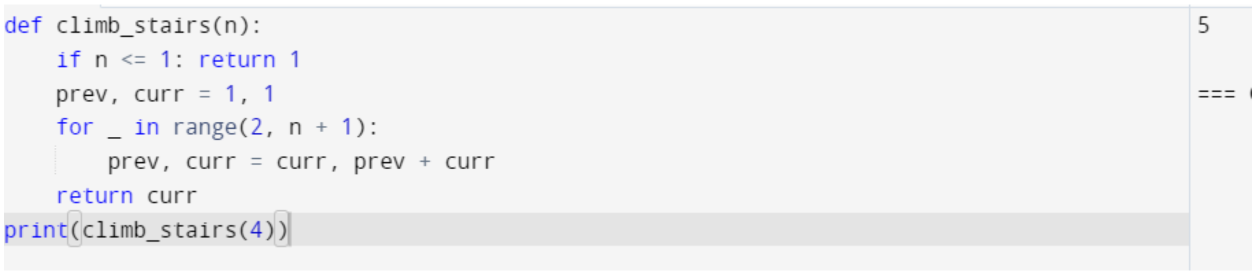


```python
def rob_houses(nums):
    def rob_linear(nums):
        prev, curr = 0, 0
        for num in nums:
            prev, curr = curr, max(curr, prev + num)
        return curr
    return max(nums[0], rob_linear(nums[1:]), rob_linear(nums[:-1])) if len(nums) != 1 else
        nums[0]
print(rob_houses([2, 3, 2]))
```

```
3

==
```

3.

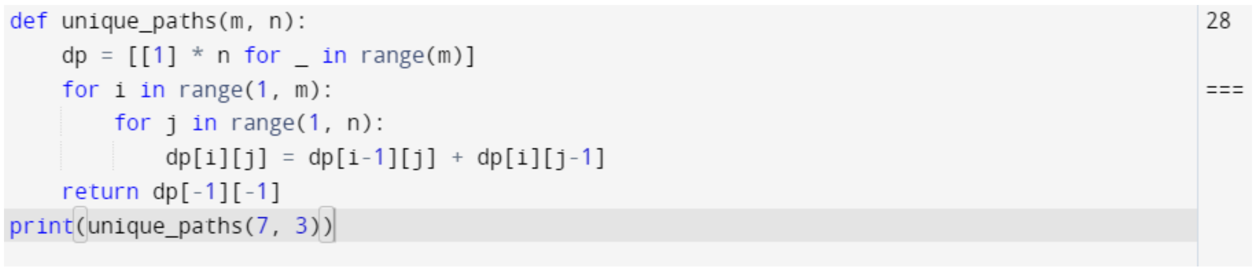


```python
def climb_stairs(n):
    if n <= 1: return 1
    prev, curr = 1, 1
    for _ in range(2, n + 1):
        prev, curr = curr, prev + curr
    return curr
print(climb_stairs(4))
```

```
5

===
```

4.

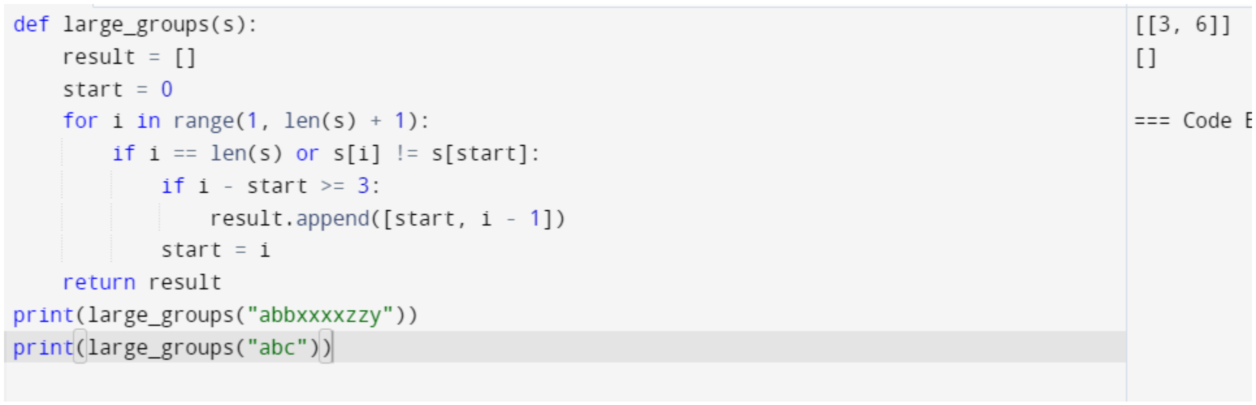


```
def unique_paths(m, n):
    dp = [[1] * n for _ in range(m)]
    for i in range(1, m):
        for j in range(1, n):
            dp[i][j] = dp[i-1][j] + dp[i][j-1]
    return dp[-1][-1]
print(unique_paths(7, 3))
```

```
28

===
```

5.

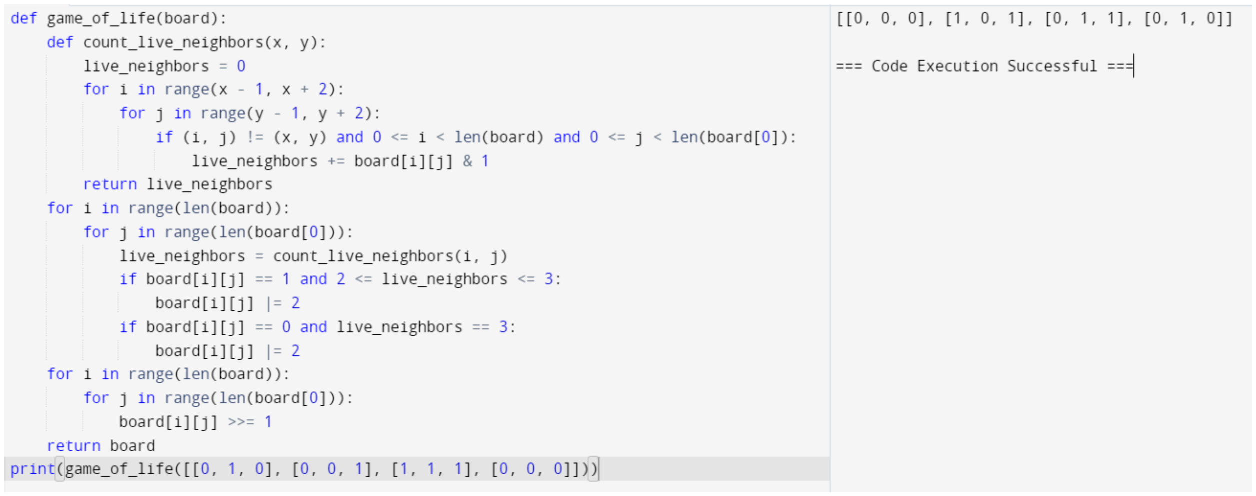


```
def large_groups(s):
    result = []
    start = 0
    for i in range(1, len(s) + 1):
        if i == len(s) or s[i] != s[start]:
            if i - start >= 3:
                result.append([start, i - 1])
            start = i
    return result
print(large_groups("abbxxxxzzy"))
print(large_groups("abc"))
```

```
[[3, 6]]
[]

=== Code E
```

6.



```
def game_of_life(board):
    def count_live_neighbors(x, y):
        live_neighbors = 0
        for i in range(x - 1, x + 2):
            for j in range(y - 1, y + 2):
                if (i, j) != (x, y) and 0 <= i < len(board) and 0 <= j < len(board[0]):
                    live_neighbors += board[i][j] & 1
        return live_neighbors
    for i in range(len(board)):
        for j in range(len(board[0])):
            live_neighbors = count_live_neighbors(i, j)
            if board[i][j] == 1 and 2 <= live_neighbors <= 3:
                board[i][j] |= 2
            if board[i][j] == 0 and live_neighbors == 3:
                board[i][j] |= 2
    for i in range(len(board)):
        for j in range(len(board[0])):
            board[i][j] >>= 1
    return board
print(game_of_life([[0, 1, 0], [0, 0, 1], [1, 1, 1], [0, 0, 0]]))
```

```
[[0, 0, 0], [1, 0, 1], [0, 1, 1], [0, 1, 0]]

=== Code Execution Successful ===
```