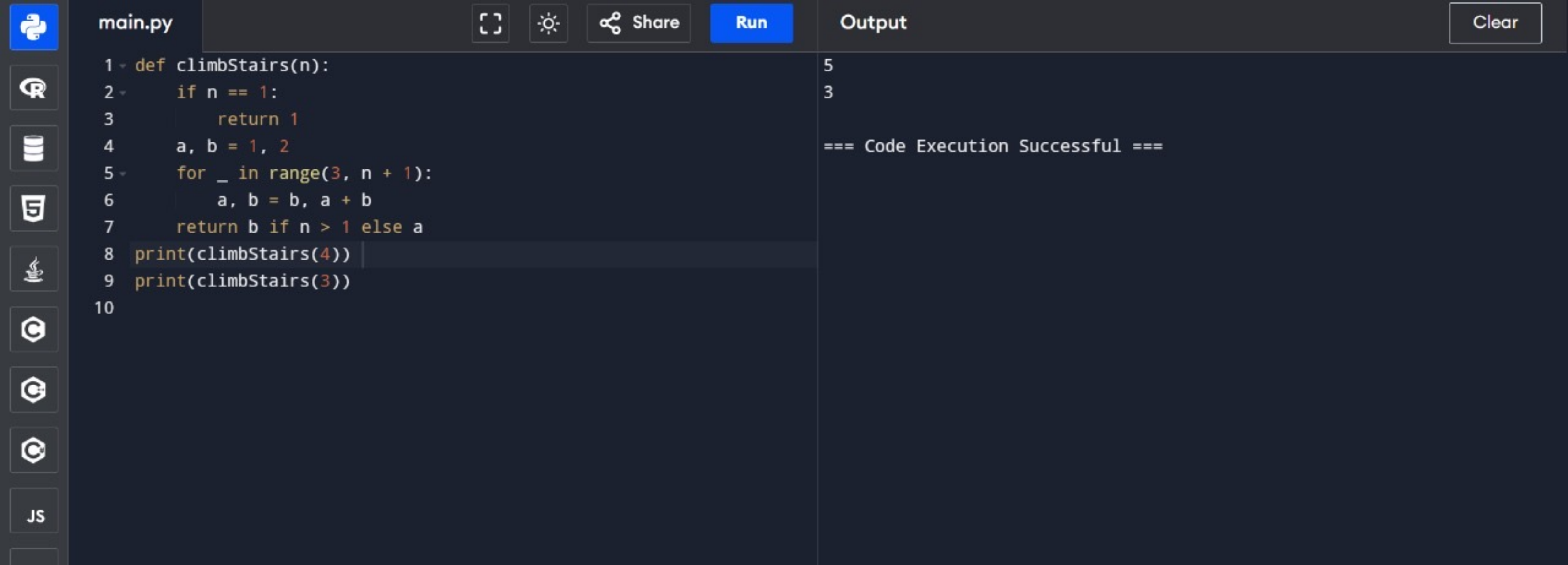
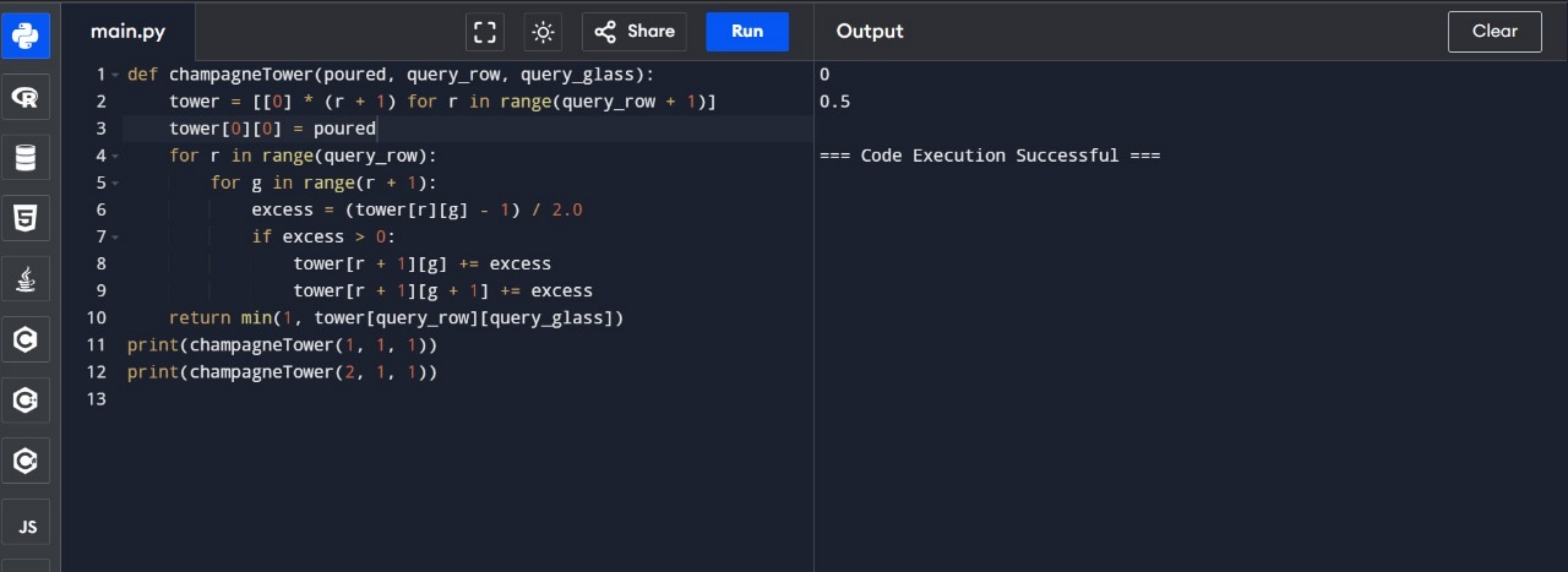


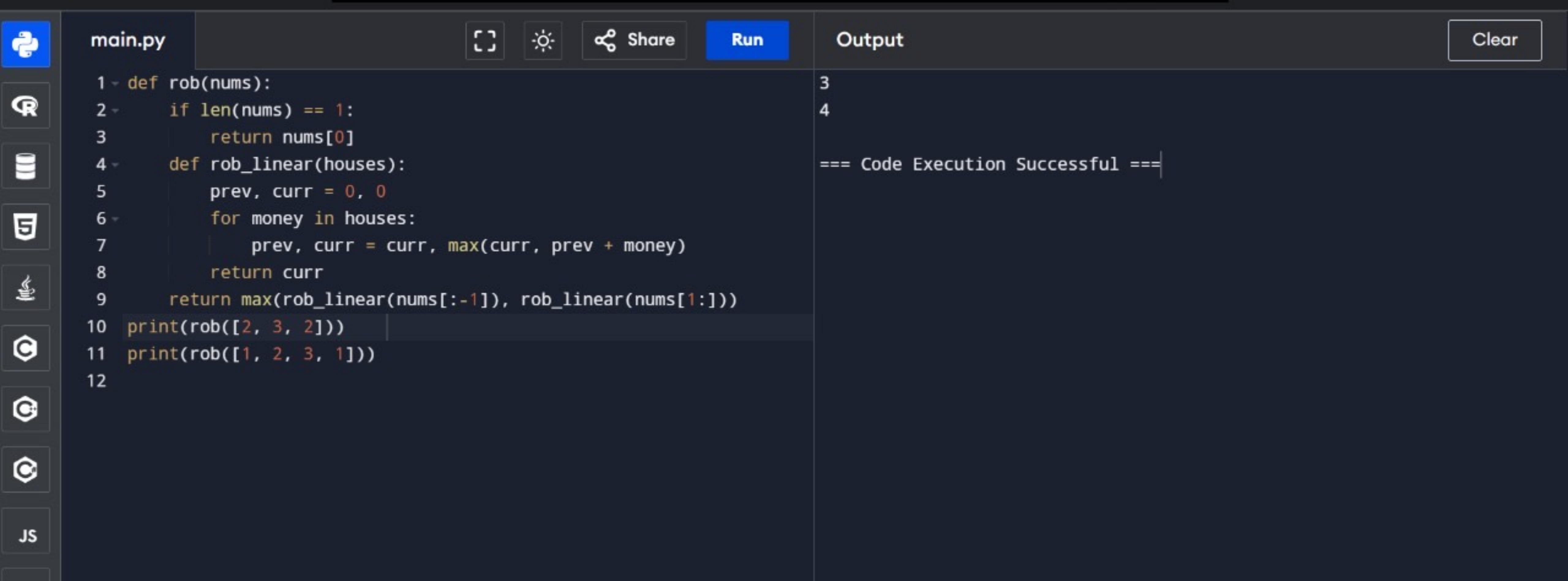
```python
def climbStairs(n):
    if n == 1:
        return 1
    a, b = 1, 2
    for _ in range(3, n + 1):
        a, b = b, a + b
    return b if n > 1 else a
print(climbStairs(4))
print(climbStairs(3))
```

```
5
3

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



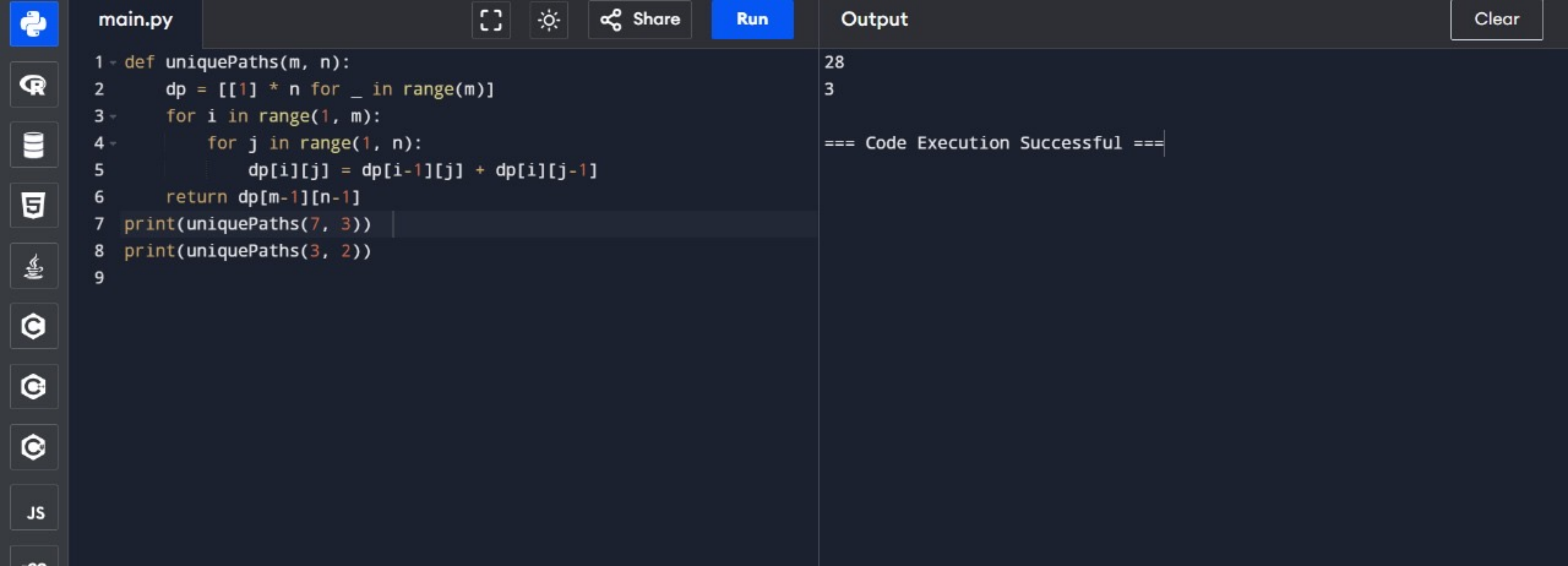
```python
def champagneTower(poured, query_row, query_glass):
    tower = [[0] * (r + 1) for r in range(query_row + 1)]
    tower[0][0] = poured
    for r in range(query_row):
        for g in range(r + 1):
            excess = (tower[r][g] - 1) / 2.0
            if excess > 0:
                tower[r + 1][g] += excess
                tower[r + 1][g + 1] += excess
    return min(1, tower[query_row][query_glass])
print(champagneTower(1, 1, 1))
print(champagneTower(2, 1, 1))
```

```
0
0.5


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



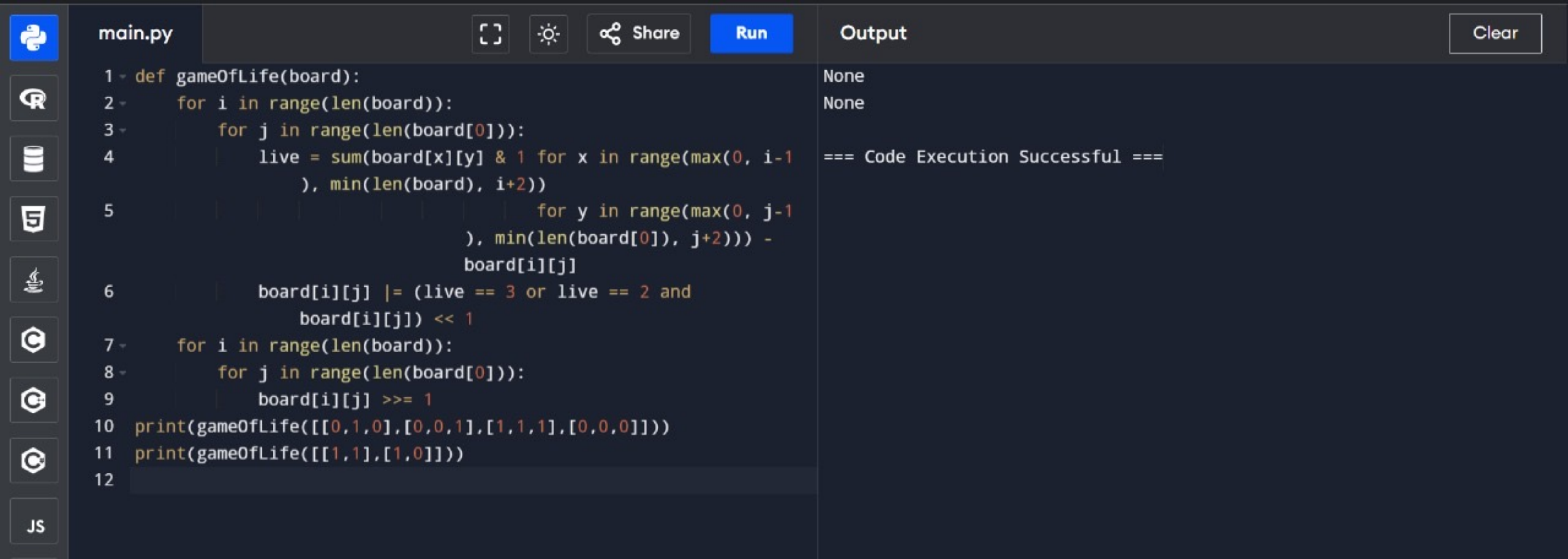
```python
def rob(nums):
    if len(nums) == 1:
        return nums[0]
    def rob_linear(houses):
        prev, curr = 0, 0
        for money in houses:
            prev, curr = curr, max(curr, prev + money)
        return curr
    return max(rob_linear(nums[:-1]), rob_linear(nums[1:]))
print(rob([2, 3, 2]))
print(rob([1, 2, 3, 1]))

```

```
3
4

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



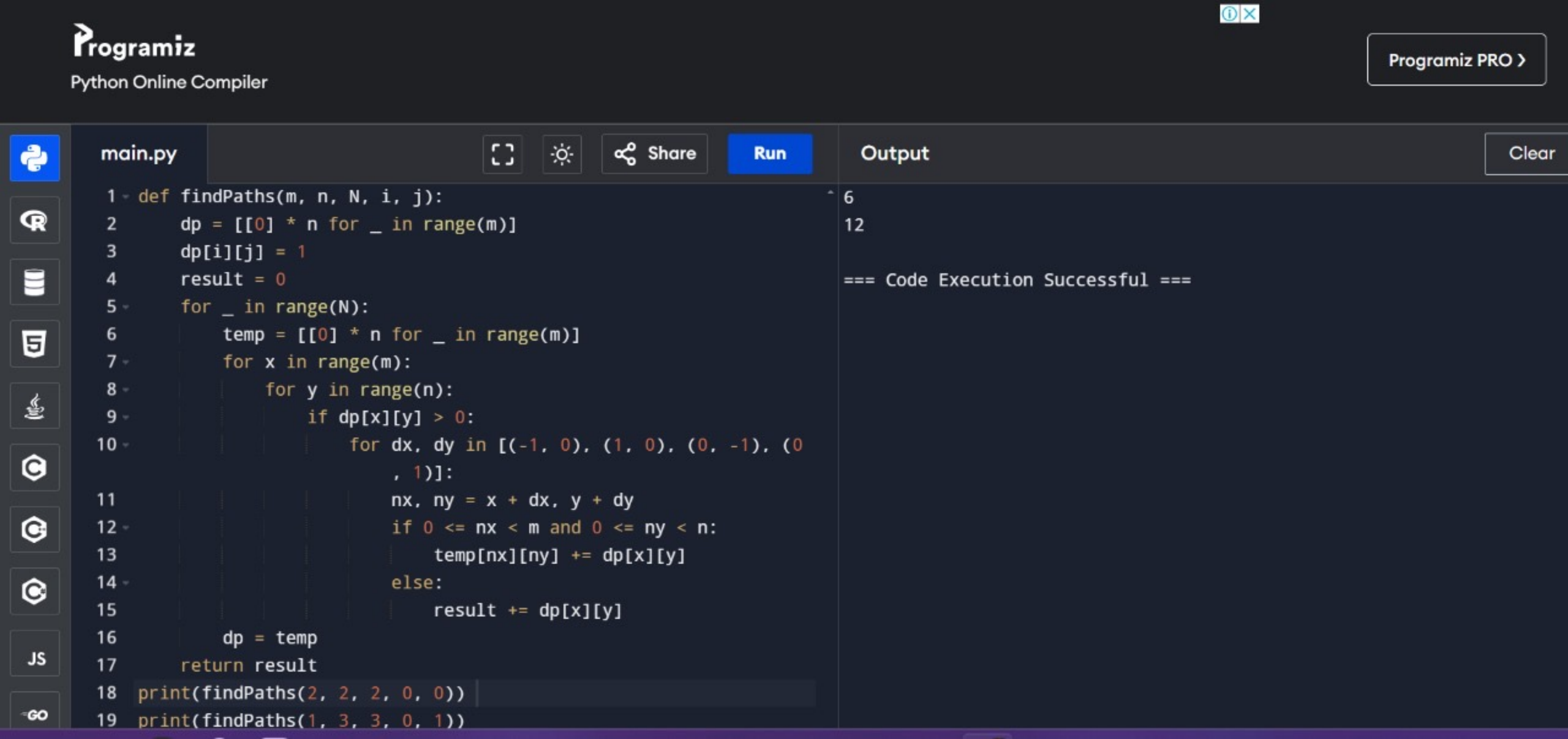
```python
def uniquePaths(m, n):
    dp = [[1] * n for _ in range(m)]
    for i in range(1, m):
        for j in range(1, n):
            dp[i][j] = dp[i-1][j] + dp[i][j-1]
    return dp[m-1][n-1]
print(uniquePaths(7, 3))
print(uniquePaths(3, 2))

```

```
28
3

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



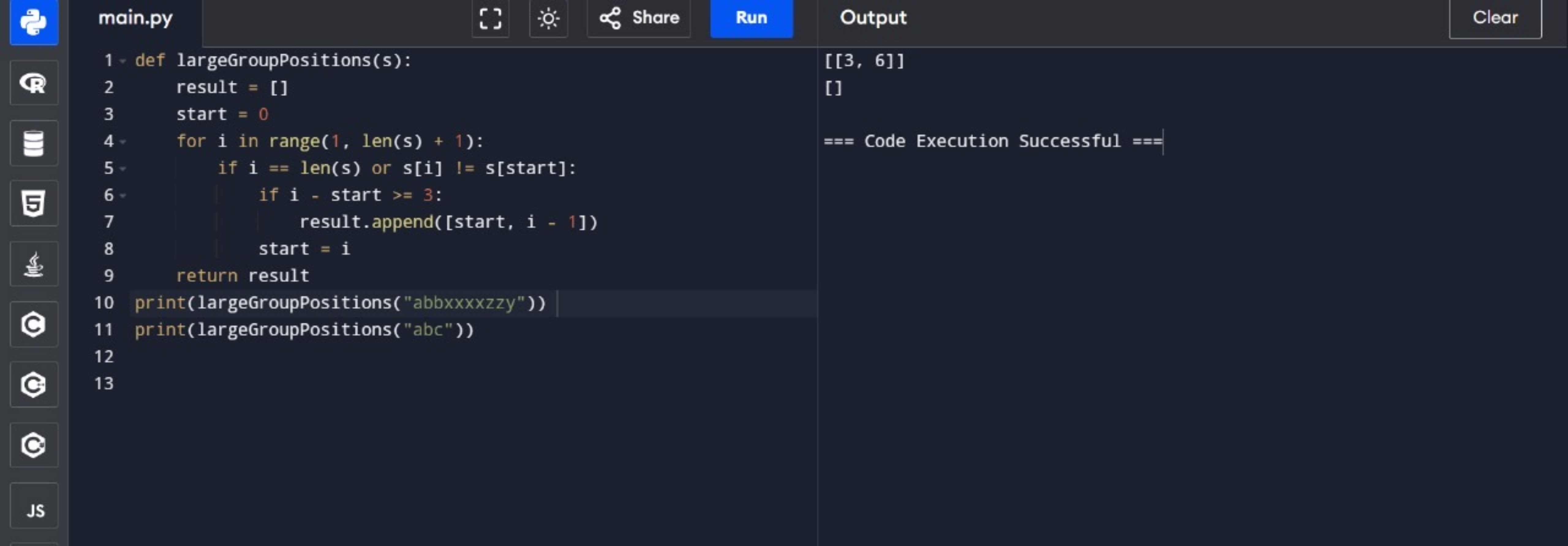
```python
def gameOfLife(board):
    for i in range(len(board)):
        for j in range(len(board[0])):
            live = sum(board[x][y] & 1 for x in range(max(0, i-1), min(len(board), i+2))
                                       for y in range(max(0, j-1), min(len(board[0]), j+2))) - board[i][j]
            board[i][j] |= (live == 3 or live == 2 and board[i][j]) << 1
    for i in range(len(board)):
        for j in range(len(board[0])):
            board[i][j] >>= 1
print(gameOfLife([[0,1,0],[0,0,1],[1,1,1],[0,0,0]]))
print(gameOfLife([[1,1],[1,0]]))

```

```
None
None

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



```python
def findPaths(m, n, N, i, j):
    dp = [[0] * n for _ in range(m)]
    dp[i][j] = 1
    result = 0
    for _ in range(N):
        temp = [[0] * n for _ in range(m)]
        for x in range(m):
            for y in range(n):
                if dp[x][y] > 0:
                    for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
                        nx, ny = x + dx, y + dy
                        if 0 <= nx < m and 0 <= ny < n:
                            temp[nx][ny] += dp[x][y]
                        else:
                            result += dp[x][y]
        dp = temp
    return result
print(findPaths(2, 2, 2, 0, 0))
print(findPaths(1, 3, 3, 0, 1))
```

```
6
12

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



```python
def largeGroupPositions(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(largeGroupPositions("abbxxxxzzy"))
print(largeGroupPositions("abc"))
```

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

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