**DAY-02**

**# If the ball moves out of the grid boundary, it counts as a successful move:-**

N, m, n, startRow, startCol = 3, 4, 4, 1, 1

dp = [[[0] \* n for \_ in range(m)] for \_ in range(N+1)]

dp[0][startRow][startCol] = 1

directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

total\_out\_bound\_ways = 0

for steps in range(N):

for row in range(m):

for col in range(n):

if dp[steps][row][col]:

for d in directions:

newRow, newCol = row + d[0], col + d[1]

if newRow < 0 or newRow >= m or newCol < 0 or newCol >= n:

total\_out\_bound\_ways += dp[steps][row][col]

else:

dp[steps + 1][newRow][newCol] += dp[steps][row][col]

print(total\_out\_bound\_ways)

**output:12**

**# The maximum money you can rob without alerting the**

**Police :-**

nums = [2, 3, 2]

n = len(nums)

if n == 1:

print(nums[0])

else:

rob1, rob2 = [0] \* n, [0] \* n

rob1[0], rob2[1] = nums[0], nums[1]

for i in range(1, n-1): rob1[i] = max(rob1[i-1], rob1[i-2] + nums[i] if i > 1 else nums[i])

for i in range(2, n): rob2[i] = max(rob2[i-1], rob2[i-2] + nums[i])

print(max(rob1[n-2], rob2[n-1]))

**ouput:3**

# **how many distinct ways can you climb to the top :-**

n = 5

if n == 0:

print(1)

elif n == 1:

print(1)

else:

dp = [0] \* (n + 1)

dp[0], dp[1] = 1, 1

for i in range(2, n + 1):

dp[i] = dp[i-1] + dp[i-2]

print(dp[n])

**output:8**

**# How many possible unique paths are there :-**

m, n = 3, 3 # Example grid size

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]

print(dp[m-1][n-1])

**output:6**

# intervals of every large group sorted in increasing order by start index :-

s = "abbxxxxzzy"

n = len(s)

res = []

start = 0

for i in range(1, n):

if s[i] != s[i-1]:

if i - start >= 3:

res.append([start, i-1])

start = i

if n - start >= 3:

res.append([start, n-1])

print(res)

**output:- [[3,6]]**

#**Q16.**

import numpy as np

board = np.array([[0, 1, 0], [0, 1, 0], [0, 1, 0]])

next\_board = np.zeros\_like(board)

for i in range(board.shape[0]):

for j in range(board.shape[1]):

n = np.sum(board[max(0, i-1):i+2, max(0, j-1):j+2]) - board[i, j]

next\_board[i, j] = (n == 3) or (board[i, j] and n in [2, 3])

print(next\_board)

**output:-**

**[[0 0 0]**

**[1 1 1]**

**[0 0 0]]**

**#**Q17.

amount = 5

pyramid = [[0] \* (i + 1) for i in range(20)]

pyramid[0][0] = amount

for i in range(20):

for j in range(i + 1):

if pyramid[i][j] > 1:

overflow = (pyramid[i][j] - 1) / 2

if i + 1 < 20: pyramid[i + 1][j] += overflow; pyramid[i + 1][j + 1] += overflow; pyramid[i][j] = 1

print(min(1, pyramid[10][0]))