def is\_safe(board, row, col, n):

for i in range(row):

if board[i][col] == 'Q':

return False

for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

if board[i][j] == 'Q':

return False

for i, j in zip(range(row, -1, -1), range(col, n)):

if board[i][j] == 'Q':

return False

return True

def solve\_n\_queens(board, row, n):

if row == n:

global count

count += 1

for row in board:

print(" ".join(row))

print("\n")

return

for col in range(n):

if is\_safe(board, row, col, n):

board[row][col] = 'Q'

solve\_n\_queens(board, row + 1, n)

board[row][col] = '.'

def n\_queens(n):

global count

count = 0

board = [['.' for \_ in range(n)] for \_ in range(n)]

solve\_n\_queens(board, 0, n)

print(f"Total solutions: {count}")

n = 4

n\_queens(n)

Output:

. Q . .

. . . Q

Q . . .

. . Q .

. . Q .

Q . . .

. . . Q

. Q . .

Total solutions: 2