# **N-Queen problem:**

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

for i in range(col):

if board[row][i] == 1:

return False

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

if board[i][j] == 1:

return False

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

if board[i][j] == 1:

return False

return True

def solve\_n\_queens\_util(board, col, N):

if col >= N:

return True

for i in range(N):

if is\_safe(board, i, col, N):

board[i][col] = 1

if solve\_n\_queens\_util(board, col + 1, N):

return True

board[i][col] = 0

return False

def solve\_n\_queens(N):

board = [[0 for \_ in range(N)] for \_ in range(N)]

if not solve\_n\_queens\_util(board, 0, N):

print(f"No solution exists for {N}-Queens")

return False

for row in board:

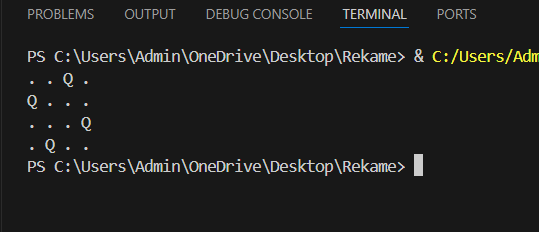
print(' '.join(['Q' if x == 1 else '.' for x in row]))

return True

N = 4

solve\_n\_queens(N)

**Output:**

****