def print\_board(board):

for row in board:

print(" ".join("Q" if col == 1 else "." for col in row))

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

# Check the column for other queens

for i in range(row):

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

return False

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

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

return False

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

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

return False

return True

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

if row >= len(board):

return True

for col in range(len(board)):

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

board[row][col] = 1 # Place the queen

if solve\_n\_queens(board, row + 1):

return True

board[row][col] = 0

return False

def eight\_queens():

N = 8

board = [[0 for \_ in range(N)] for \_ in range(N)] # Create an empty board

if solve\_n\_queens(board, 0):

print\_board(board)

else:

print("Solution does not exist")

eight\_queens()

