def solve\_n\_queens(n):

# Initialiser l'échiquier

board = [['.' for i in range(n)] for j in range(n)]

# Stocker les positions des reines

positions = []

# Fonction récursive pour placer les reines sur l'échiquier

def place\_queen(row):

# Si toutes les reines ont été placées, renvoyer True

if row == n:

return True

# Parcourir toutes les colonnes pour cette rangée

for col in range(n):

# Vérifier si la reine peut être placée dans cette colonne

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

# Placer la reine

board[row][col] = 'Q'

positions.append((row, col))

# Récursivement placer les reines pour la rangée suivante

if place\_queen(row + 1):

return True

# Si aucune solution n'a été trouvée, revenir en arrière et retirer la reine

board[row][col] = '.'

positions.pop()

# Si aucune reine ne peut être placée dans cette rangée, renvoyer False

return False

# Fonction pour vérifier si la reine peut être placée dans la position donnée

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

# Vérifier la colonne

for i in range(n):

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

return False

# Vérifier la diagonale montante

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

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

return False

# Vérifier la diagonale descendante

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

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

return False

# Si toutes les vérifications sont passées, renvoyer True

return True

# Appeler la fonction pour placer les reines

place\_queen(0)

# Afficher la solution

for row, col in positions:

print('({}, {})'.format(row, col))

# Définition d'une fonction pour vérifier si une position est sûre

def isSafe(board, row, col, N):

# Vérifier la rangée de gauche à droite

for i in range(col):

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

return False

# Vérifier la diagonale supérieure de gauche à droite

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

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

return False

# Vérifier la diagonale inférieure de gauche à droite

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

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

return False

# Si la position est sûre, retourner True

return True

# Définition de la fonction pour placer les reines

def placeQueens(board, col, N):

# Si toutes les colonnes sont parcourues, retourner True

if col >= N:

return True

# Parcourir chaque rangée pour cette colonne

for i in range(N):

# Vérifier si la position est sûre

if isSafe(board, i, col, N):

# Si oui, placer la reine à cette position

board[i][col] = 1

# Appeler la fonction récursivement pour placer la reine suivante

if placeQueens(board, col + 1, N):

return True

# Si la position n'est pas sûre, retirer la reine de cette position

board[i][col] = 0

# Si toutes les positions ont été vérifiées sans succès, retourner False

return False

# Définition de la fonction principale pour résoudre le problème des 8 reines

def solveNQueens(N):

# Création d'un tableau vide pour le plateau de jeu

board = [[0 for x in range(N)] for y in range(N)]

# Appel de la fonction récursive pour placer les reines

if placeQueens(board, 0, N) == False:

print("Aucune solution n'a été trouvée.")

return False

# Affichage de la solution trouvée

for i in range(N):

for j in range(N):

print(board[i][j], end=" ")

print()

return True

# Exemple d'utilisation pour résoudre le problème des 8 reines avec un plateau de 8x8

solveNQueens(8)