The **4-Queens Problem** is a classic AI problem where we need to place 4 queens on a **4×4 chessboard** such that no two queens threaten each other. This means:

* No two queens should be in the same **row**.
* No two queens should be in the same **column**.
* No two queens should be on the same **diagonal**.

def print\_solution(board):

for row in board:

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

print("\n")

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

# Check column

for i in range(row):

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

return False

# Check upper-left diagonal

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

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

return False

# Check upper-right diagonal

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

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

return False

return True

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

if row >= N: # All queens placed

print\_solution(board)

return True

for col in range(N):

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

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

if solve\_n\_queens(board, row + 1, N): # Recur for next row

return True

board[row][col] = 0 # Backtrack if no solution found

return False # No valid configuration found

def four\_queens():

N = 4

board = [[0] \* N for \_ in range(N)] # Initialize 4x4 board

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

print("No solution exists.")

# Run the 4-Queens Solver

four\_queens()