**PROGRAM 2:**

Write the python program to solve 8-Queen problem

**AIM:** To write a python to solve 8-Queen problem

**Algorithm**:

Step 1:Board Representation: Represent the chessboard as an 8x8 grid, where each cell can be either empty or occupied by a queen.

Step 2: Place Queen Function: Implement a function that attempts to place a queen in a given row. This function should check if the placement is valid by ensuring that no other queens threaten the newly placed queen.

Step 3: Backtracking Algorithm: a. Start with an empty chessboard. b. For each row from 0 to 7: i. Call the "Place Queen" function for the current row. ii. If a valid placement is found, move to the next row. iii. If no valid placement is found, backtrack to the previous row and continue searching for valid placements. c. If a valid placement is found for all rows, a solution is found.

Step 4: Recursion and Backtracking: a. The "Place Queen" function:

i. Given a row, try placing a queen in each column of that row.

ii. Check if the placement is valid by ensuring that no other queens threaten the newly placed queen.

iii. If the placement is valid, mark the cell as occupied by a queen and move to the next row recursively.

iv. If no valid placement is found in the current row, backtrack by removing the queen from the cell and returning to the previous row.

Step 5: Solution Extraction: Once a solution is found, you can extract the positions of the queens on the board to visualize the solution.

**Program:**

# Taking number of queens as input from user

print ("Enter the number of queens")

N = int(input())

# here we create a chessboard

# NxN matrix with all elements set to 0

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

def attack(i, j):

#checking vertically and horizontally

for k in range(0,N):

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

return True

#checking diagonally

for k in range(0,N):

for l in range(0,N):

if (k+l==i+j) or (k-l==i-j):

if board[k][l]==1:

return True

return False

def N\_queens(n):

if n==0:

return True

for i in range(0,N):

for j in range(0,N):

if (not(attack(i,j))) and (board[i][j]!=1):

board[i][j] = 1

if N\_queens(n-1)==True:

return True

board[i][j] = 0

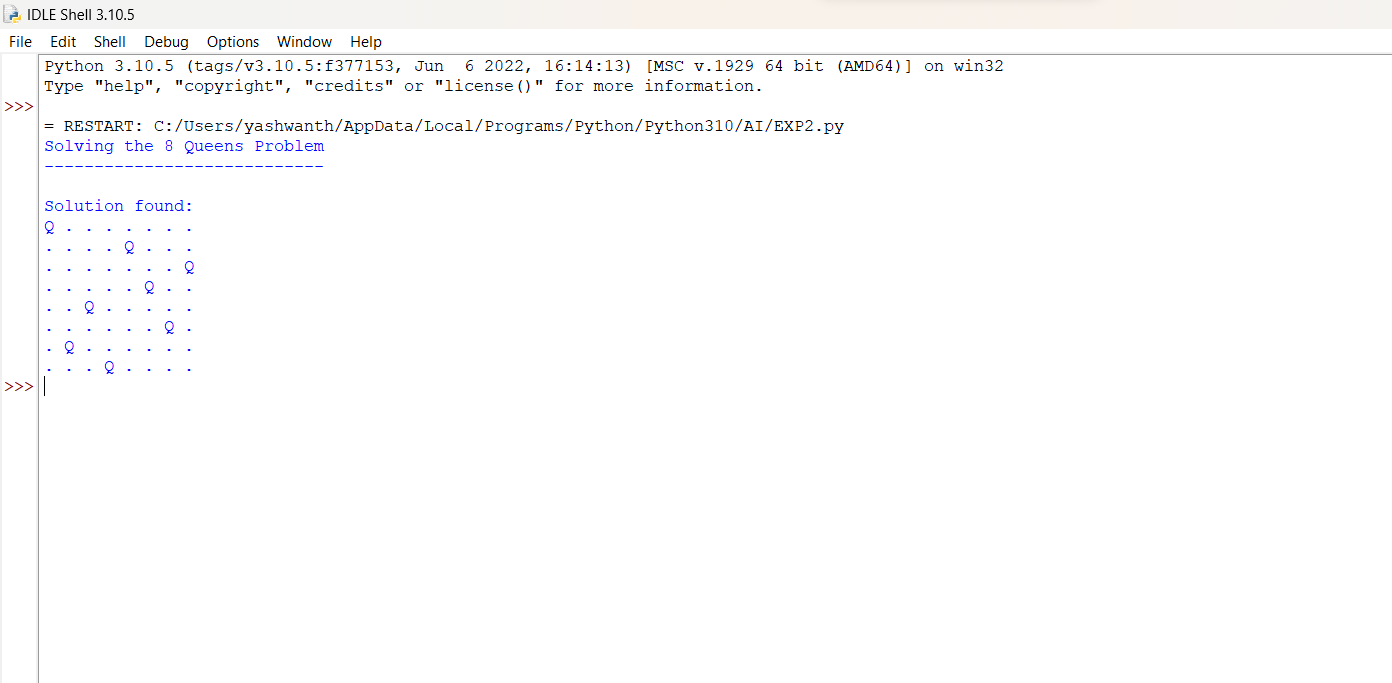
return False

N\_queens(N)

for i in board:

print (i)

**Output**:



**Result:**

The program to execute 8 queen program is executed successfully in python.