SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be **University** *u/s 3 of UGC Act, 1956)*

18CSC305J – ARTIFICIAL INTELLIGENCE LAB

Exp-1: 8 Queen Problem

Submitted by-
Name:- Anannya P. Neog
Reg. No. :- RA1911003010367
Course :- Btech
Section :- F1
Branch:- Computer Science Engineering
Sem:- 6th Sem

AI LAB Ex – 1:- 8 Queens Problem

Team Members:

- ✓ Richa - 357
- ✓ Anannya - 367
- ✓ Pushan - 371
- ✓ Ankit - 372
- ✓ Tanay - 377

Problem:

- The **eight queens puzzle** is the problem of placing 8 chess queens on an 8×8 chessboard so that no two queens threaten each other.
- Thus, a solution requires that no two queens share the same row, column, or diagonal.

Objective:

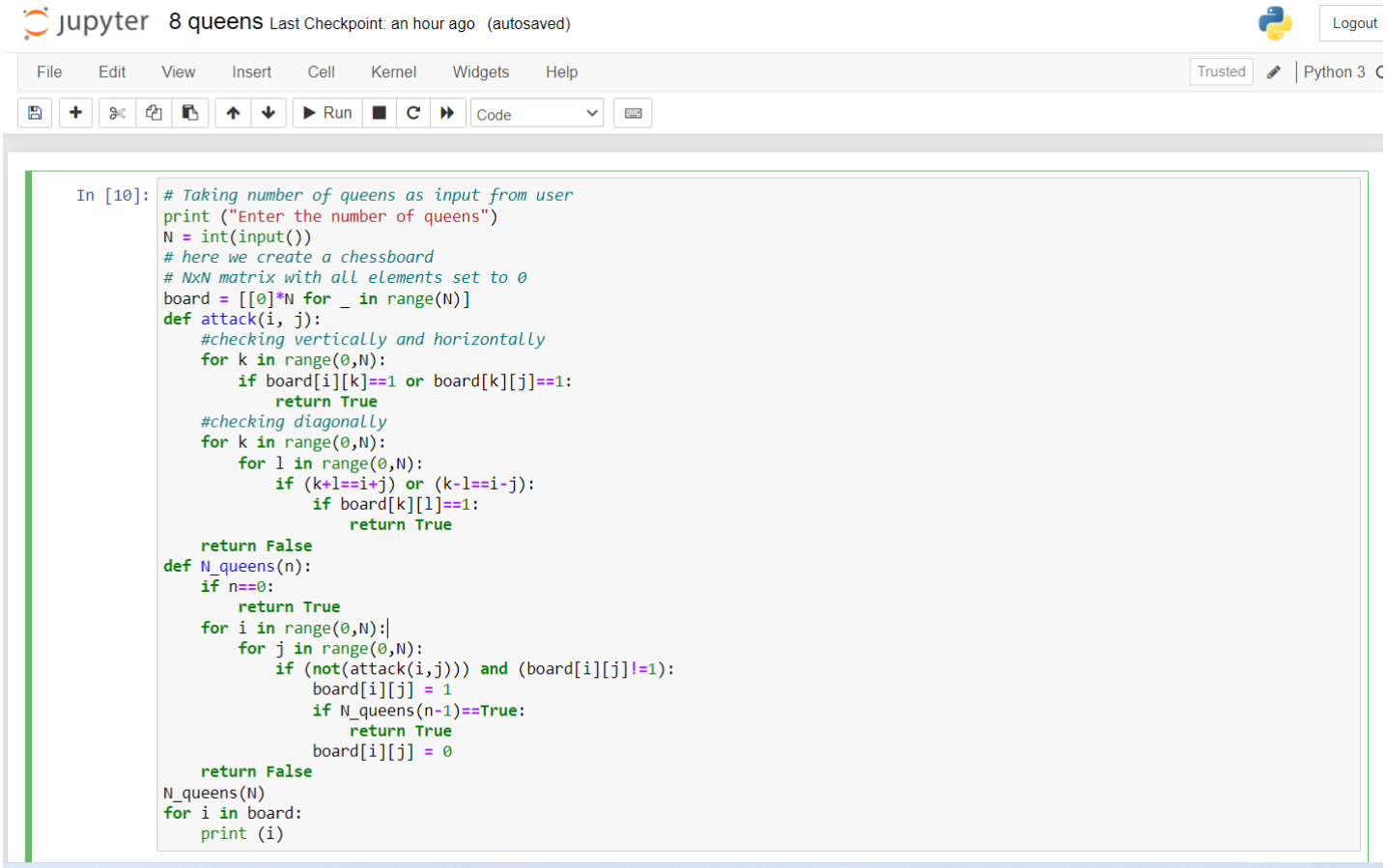
- **Arrangements of 8 queens in 64 sections:** There are 64 possible places, so we need to choose 8 to place the queens there. This can be done in 64C8 ways.
- **Arrangements of 1 queen per row:** If we restrict one queen per row, each queen has 8 possible places, so the total arrangements is 8^8ways.
- **Permutations of 8 queens, 1 queen per row:**If we only take care of the permutations of the numbers 1 to 8, and map the first place to row 1, the second place to row 2, and so on&we do not worry anymore about being in the same row or being in the same column. The total arrangements in this case is 8!

Code:

```
# 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)
```

Code Screenshot:



Output:

```
Enter the number of queens
8
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0]
```

So one of the optimal solution is (1,1) (2,5), (3,8), (4,6), (5,3), (6,7), (7,2), (8,4)