Artificial Inteligence  
Lab Exercise 1

short line

Ashwin Prakash  
RA1911026010048

**The n- Queens Problem:**

The n queen’s problem can be stated as follows. Consider a n \* n chessboard on which we have to place n queens so that no two queens attack each other by being in the same row or in the same column or on the same diagonal

**Aim:** To solve N- Queens Problem.

**Algorithm: Backtracking Algorithm**

1) Start in the leftmost column

2) If all queens are placed

return true

3) Try all rows in the current column.

a) If the queen can be placed safely in this row

then mark this [row, column] as part of the

solution and recursively check if placing

queen here leads to a solution.

b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then

unmark this [row, column] (Backtrack) and go to

step (a) to try other rows.

3) If all rows have been tried and nothing worked,

return false to trigger backtracking.

**Program:**

#Number of queens

N = int(input(("Enter the number of queens")))

#chessboard

#NxN matrix with all elements 0

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

def will\_attack(i, j):

    #checking if there is a queen in row or column

    for k in range(0,N):

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

            return True

    #checking diagonals

    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\_queen(n):

    #if n is 0, solution found

    if n==0:

        return True

    for i in range(0,N):

        for j in range(0,N):

            '''checking if we can place a queen here or not

            queen will not be placed if the place is being attacked

            or already occupied'''

            if (not(will\_attack(i,j))) and (board[i][j]!=1):

                board[i][j] = 1

                #recursion

                #whether we can put the next queen with this arrangment or not

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

                    return True

                board[i][j] = 0

    return False

def printBorad(brd):

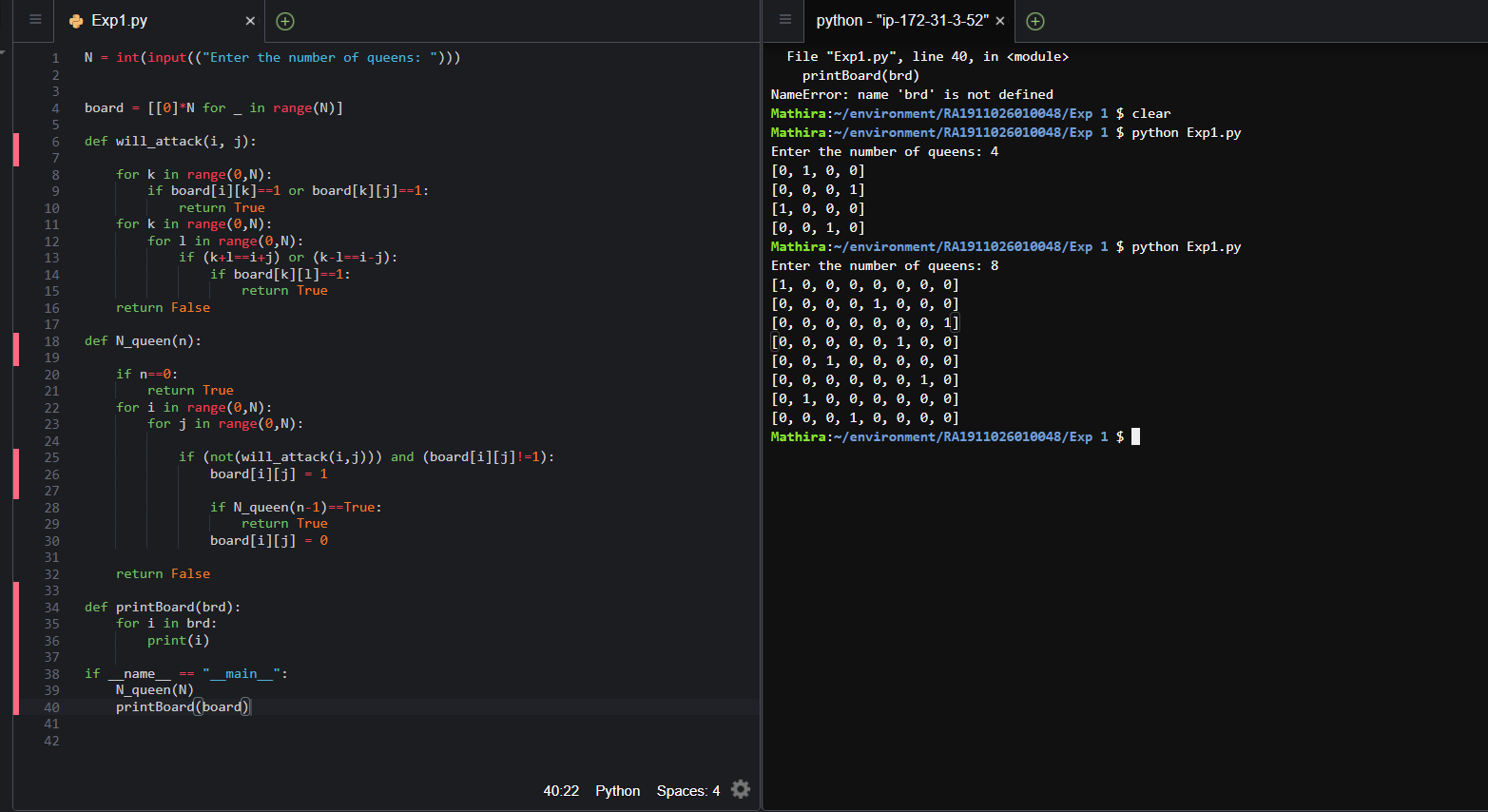
for i in brd:

print(i)

if \_\_name\_\_ == “\_\_main”:

N\_queen(N)

printBoard(board)

**Output:**

**Result:** The n-queens problem (4 queens problem) has been solved using a backtracking algorithm.