Artificial Intelligence
Lab 1

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 and **return true**

print the solution.

3) Try all rows in the current column.

a) If the queen can be placed safely in this rowthen 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.

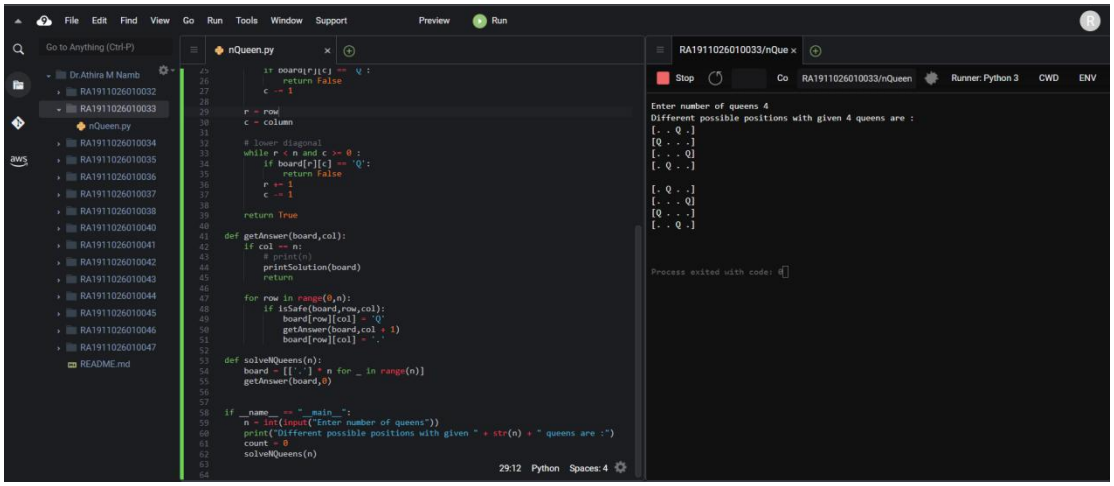
3) If all rows have been tried and nothing worked, **return false** to trigger backtracking.

Code :

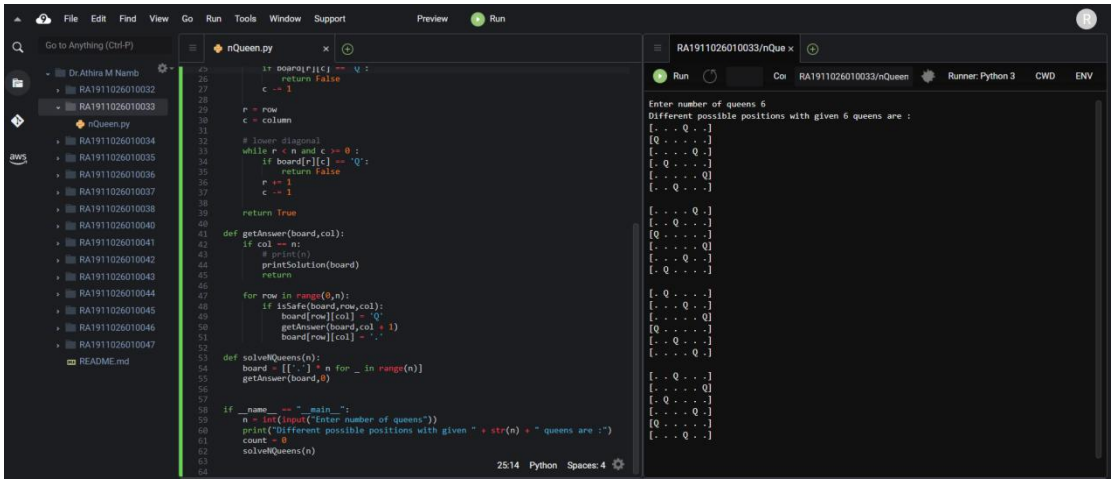
```
global n
global answer

def printSolution(board):
    for r in board:
        print(str(r).replace(',', '').replace('\'', ''))
    print()
def isSafe(board,row,column):
    # print(board)
    r = row
    c = column
    # upper diagonal
    while r >= 0 and c >= 0:
        if board[r][c] == 'Q':
            return False
        r -= 1
        c -= 1

    r = row
    c = column

    # left side
    while c >= 0:
        if board[r][c] == 'Q':
            return False
        c -= 1

    r = row
    c = column

    # lower diagonal
    while r < n and c >= 0 :
        if board[r][c] == 'Q':
            return False
        r += 1
        c -= 1

    return True
def getAnswer(board,col):
    if col == n:
        # print(n)
        printSolution(board)
        return

    for row in range(0,n):
        if isSafe(board,row,col):
            board[row][col] = 'Q'
            getAnswer(board,col + 1)
            board[row][col] = '.'
def solveNQueens(n):
    board = [['.'] * n for _ in range(n)]
    getAnswer(board,0)

if __name__ == "__main__":
    n = int(input("Enter number of queens"))
    print("Different possible positions with given " + str(n) + " queens are :")
    count = 0
    solveNQueens(n)
```

Output :

N = 4



N = 6



Result: The n-queens problem has been solved using a backtracking algorithm.