SUPERIOR UNIVERSITY

Name:

Mahak Farhan

Roll no.:

068

Class:

BSAI

Section:

4B

Subject:

Programming for Artificial Intelligence

Submitted to:

Sir Rasikh Ali

Lab 4

N Queens problem

Code:

```
def position(board, row, col, n):


    for i in range(row):
        if board[i][col] == 'Q':
            return False


    # Check upper-left diagonal
    i, j = row, col
    while i >= 0 and j >= 0:
        if board[i][j] == 'Q':
            return False
        i -= 1
        j -= 1


    # Check upper-right diagonal
    i, j = row, col
    while i >= 0 and j < n:
        if board[i][j] == 'Q':
```

```
            return False
        i -= 1
        j += 1

    return True

def display_board(board,n):
    print("Solution:")
    for row in board:
        print(" ".join(row))
    print()

def solve_n_queens(board, row, n):
    if row == n:
        display_board(board, n)
        return

    for col in range(n):
        if is_safe(board, row, col, n):
            board[row][col] = 'Q'
            solve_n_queens(board, row + 1, n)
```

```
        board[row][col] = '-'


n = int(input("Enter the number of queens: "))
board = [['-' for _ in range(n)] for _ in range(n)]
solve_n_queens(board, 0,n)
```

Description:

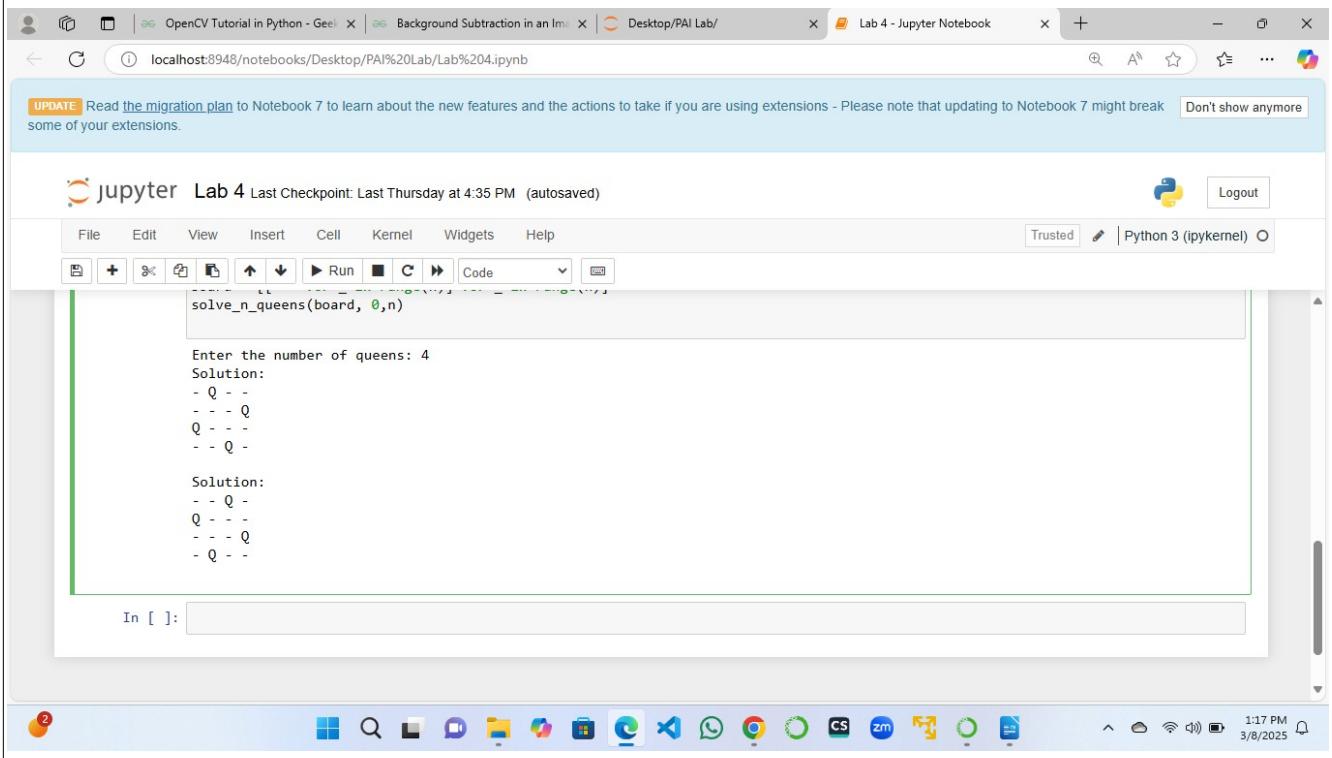
This program is implemented using the backtracking algorithm. The following functions are used:

1. **position(board, row, col, n)**: Checks whether a queen can be placed in the given row and column by checking no other queen exist in the same column, row or diagonals.
2. **display_board(board, n)**: Prints the chessboard structure for each valid solution.
3. **solve_n_queens(board, row, n)**: Recursively places queens row by row, backtracking if a conflict is detected.
4. **Main function**: Initializes the chessboard and calls solve_n_queens to start solving the problem.

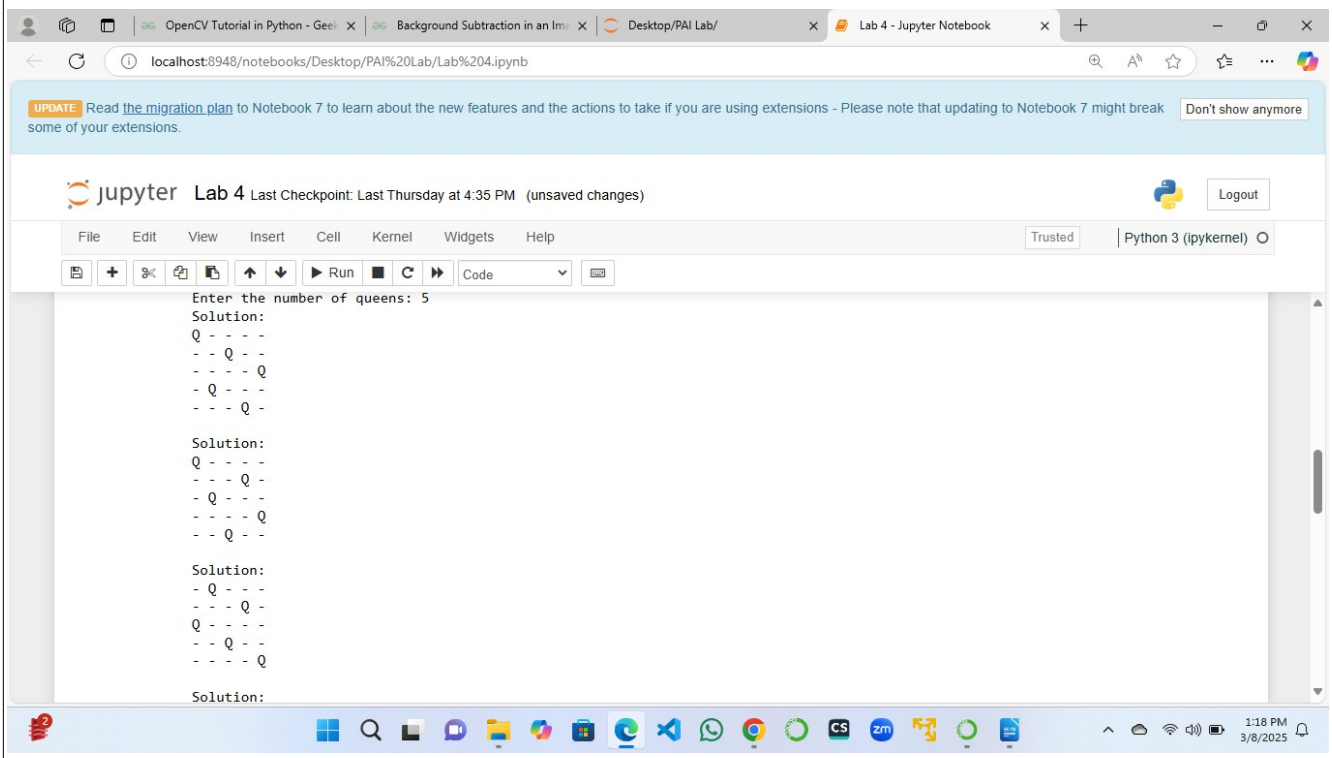
Programming for Artificial Intelligence

Output:

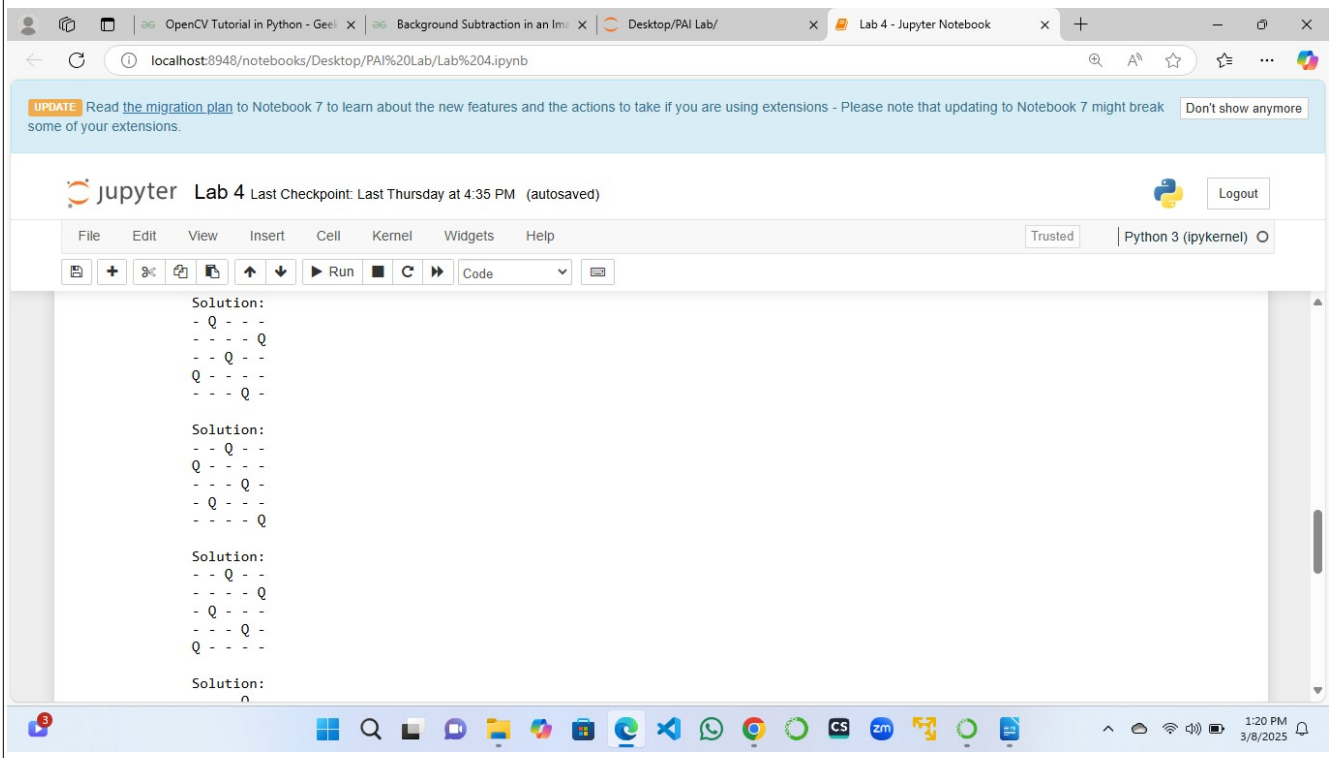
For 4 Queens:

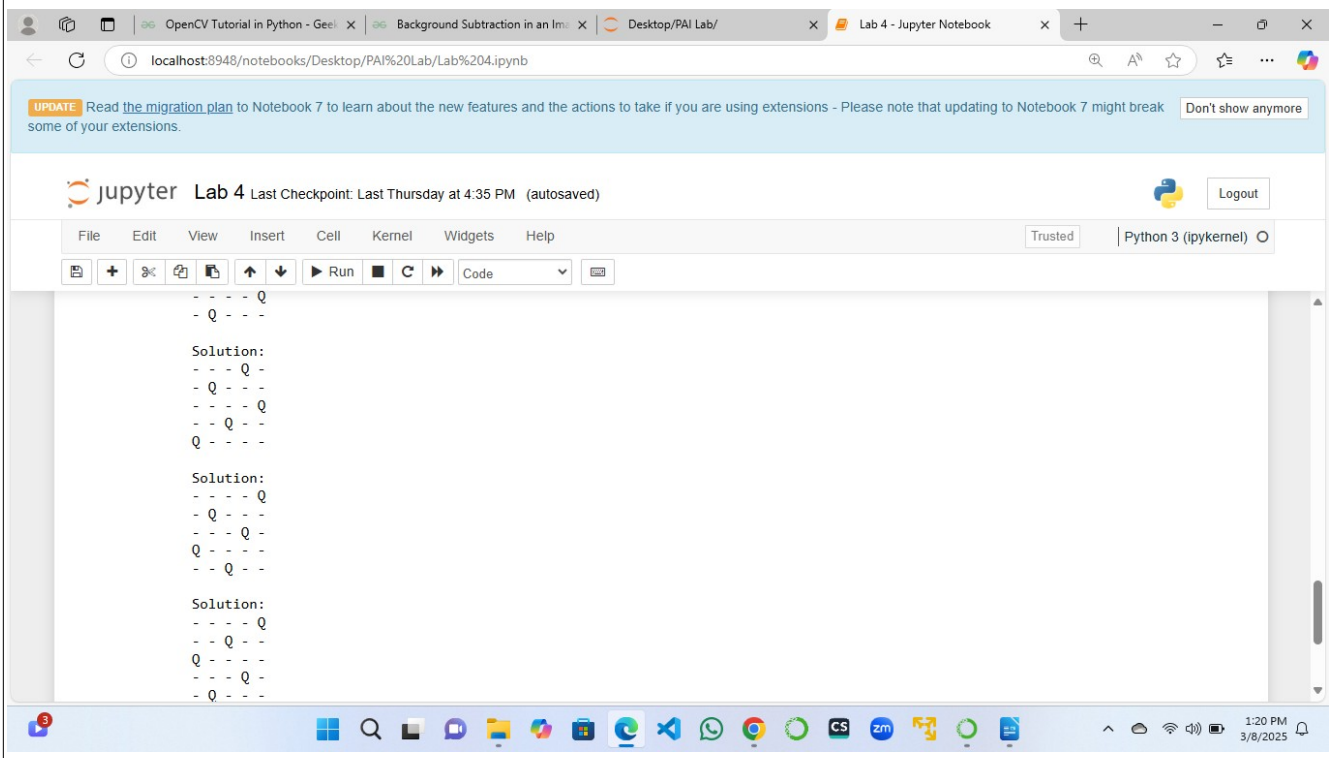


For 5 Queens:



Programming for Artificial Intelligence





Programming for Artificial Intelligence
