

**Practical No: 4**

**Roll No:**

**Subject:** Artificial Intelligence

**Title:** Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph colouring problem

---

**Program Code :**

```
def issafe(arr, x, y, n):
    for row in range(x):
        if arr[row][y] == 1:
            # Checking column attack
            return False

    row = x
    col = y
    # Checking Diagonal Attack
    while row >= 0 and col >= 0:
        if arr[row][col] == 1:
            return False
        row -= 1
        col -= 1

    row = x
    col = y
    # Checking Anti Diagonal Attack
    while row >= 0 and col < n:
        if arr[row][col] == 1:
            return False
        row -= 1
        col += 1

    return True

def nQueen(arr, x, n):
    if x >= n:
        return True

    for col in range(n):
        if issafe(arr, x, col, n):
            arr[x][col] = 1
            if nQueen(arr, x + 1, n):
                return True
            arr[x][col] = 0

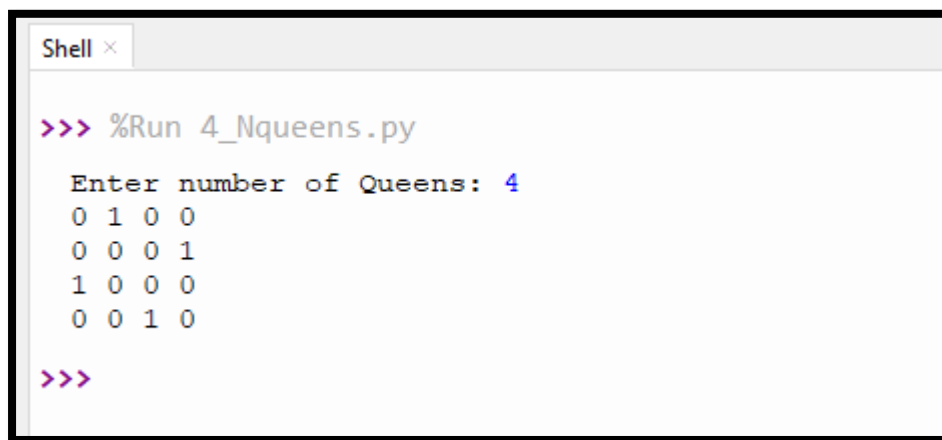
    return False
```

```
def main():
    n = int(input("Enter number of Queens: "))
    arr = [[0] * n for _ in range(n)]

    if nQueen(arr, 0, n):
        for i in range(n):
            for j in range(n):
                print(arr[i][j], end=" ")
            print()

if __name__ == '__main__':
    main()
```

Output :

A screenshot of a Jupyter Notebook's interactive shell. The window title is "Shell x". The prompt is ">>> %Run 4\_Nqueens.py". The program prompts "Enter number of Queens: 4". The output shows a 4x4 matrix of 0s and 1s representing the solutions to the 4-Queens problem. The matrix is:  
0 1 0 0  
0 0 0 1  
1 0 0 0  
0 0 1 0  
The prompt ">>>" is shown at the bottom.