```
def solve(bo):
    find = find_empty(bo)
    if not find:
        return True
    else:
        row, col = find
    for i in range(1, 10):
        if valid(bo, i, (row, col)):
            bo[row][col] = i

            if solve(bo):
                return True
            bo[row][col] = 0
    return False



def valid(bo, num, pos):
    for i in range(len(bo[0])):
        if bo[pos[0]][i] == num and pos[1] != i:
            return False
    for i in range(len(bo)):
        if bo[i][pos[1]] == num and pos[0] != i:
            return False
    box_x = pos[1] // 3
    box_y = pos[0] // 3
    for i in range(box_y*3, box_y*3 + 3):
        for j in range(box_x * 3, box_x*3 + 3):
            if bo[i][j] == num and (i, j) != pos:
                return False
    return True



def print_board(bo):
    for i in range(len(bo)):
        if i % 3 == 0 and i != 0:
            print("- - - - - - - - - - - - - ")
        for j in range(len(bo[0])):
            if j % 3 == 0 and j != 0:
                print(" | ", end="")
            if j == 8:
                print(bo[i][j])
            else:
                print(str(bo[i][j]) + " ", end="")



def find_empty(bo):
    for i in range(len(bo)):
        for j in range(len(bo[0])):
            if bo[i][j] == 0:
                return (i, j)

    return None



board = [
    [7, 8, 0, 4, 0, 0, 1, 2, 0],
    [6, 7, 0, 0, 7, 5, 0, 0, 9],
    [0, 7, 0, 6, 0, 1, 0, 7, 8],
    [0, 0, 7, 0, 4, 0, 2, 6, 0],
    [0, 0, 1, 0, 5, 0, 9, 3, 0],
    [9, 0, 4, 0, 6, 0, 0, 0, 5],
    [0, 7, 0, 3, 0, 0, 0, 1, 2],
    [1, 2, 0, 0, 0, 7, 4, 0, 0],
    [0, 4, 9, 2, 0, 6, 0, 0, 7]
]
print("Sudoku: \n".strip())
print_board(board)
solve(board)
print("\n")
print("Solution: \n".strip())
print_board(board)
```