```
def is_valid(board, row, col, num):
    # Check if the number is already in the row
    for x in range(9):
        if board[row][x] == num:
            return False

    # Check if the number is already in the column
    for x in range(9):
        if board[x][col] == num:
            return False

    # Check if the number is already in the 3x3 box
    start_row = row - row % 3
    start_col = col - col % 3
    for i in range(3):
        for j in range(3):
            if board[i + start_row][j + start_col] == num:
                return False

    return True

def find_empty_location(board):
    for row in range(9):
        for col in range(9):
            if board[row][col] == 0:
                return row, col
    return None, None

def solve_sudoku(board):
    row, col = find_empty_location(board)

    if row is None and col is None:
        return True  # Puzzle solved successfully

    for num in range(1, 10):
        if is_valid(board, row, col, num):
            board[row][col] = num

            if solve_sudoku(board):
                return True

            board[row][col] = 0  # Backtrack

    return False

def print_board(board):
    for row in board:
        print(" ".join(map(str, row)))

def main():
    # Example Sudoku puzzle (0 represents empty cells)
    puzzle = [
        [5, 3, 0, 0, 7, 0, 0, 0, 0],
        [6, 0, 0, 1, 9, 5, 0, 0, 0],
```

```
        [0, 9, 8, 0, 0, 0, 0, 6, 0],
        [8, 0, 0, 0, 6, 0, 0, 0, 3],
        [4, 0, 0, 8, 0, 3, 0, 0, 1],
        [7, 0, 0, 0, 2, 0, 0, 0, 6],
        [0, 6, 0, 0, 0, 0, 2, 8, 0],
        [0, 0, 0, 4, 1, 9, 0, 0, 5],
        [0, 0, 0, 0, 8, 0, 0, 7, 9]
    ]

    if solve_sudoku(puzzle):
        print("Sudoku puzzle solved successfully:")
        print_board(puzzle)
    else:
        print("No solution exists.")

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