65. 37Sudoku Solver

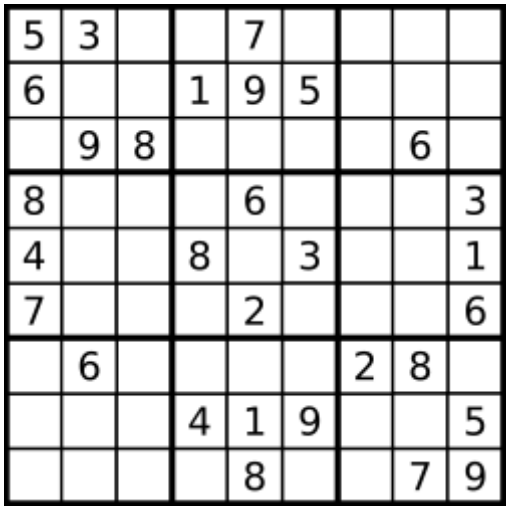
Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy all of the following rules:

1. **Each of the digits 1-9 must occur exactly once in each row.**
2. **Each of the digits 1-9 must occur exactly once in each column.**
3. **Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.**

The '.' character indicates empty cells.

Example 1:



Input: board =
[["5","3",".",".","7",".",".",".","."],["6",".",".","1","9","5",".",".","."],[".","9","8",".",".",".",".","6","."],["8",".",".",".","6",".",".",".","3"],["4",".",".","8",".","3",".",".","1"],["7",".",".",".",".","2",".",".","6"],[".","6",".",".",".",".",".","2","8","."],[".",".",".","4","1","9",".",".","5"],[".",".",".",".",".","8",".",".","7","9"]]

Output:
[["5","3","4","6","7","8","9","1","2"],["6","7","2","1","9","5","3","4","8"],["1","9","8","3","4","2","5","6","7"],["8","5","9","7","6","1","4","2","3"],["4","2","6","8","5","3","7","9","1"],["7","1","3","9","2","4","8","5","6"],["9","6","1","5","3","7","2","8","4"],["2","8","7","4","1","9","6","3","5"],["3","4","5","2","8","6","1","7","9"]]

PROGRAM:-

```
def solveSudoku(board):
    def isValid(board, row, col, num):
        for i in range(9):
            # Check if the number is already in the row or column
```

```
            if board[row][i] == num or board[i][col] == num:
                return False
            # Check if the number is already in the 3x3 sub-box
            if board[row//3*3 + i//3][col//3*3 + i%3] == num:
                return False
        return True

    def solve(board):
        for row in range(9):
            for col in range(9):
                if board[row][col] == '.':
                    for num in '123456789':
                        if isValid(board, row, col, num):
                            board[row][col] = num
                            if solve(board):
                                return True
                            board[row][col] = '.'
                    return False
        return True

    solve(board)

# Example usage and output
board = [
    ["5", "3", ".", ".", "7", ".", ".", ".", "."],
    ["6", ".", ".", "1", "9", "5", ".", ".", "."],
    [".", "9", "8", ".", ".", ".", ".", "6", "."],
    ["8", ".", ".", ".", "6", ".", ".", ".", "3"],
    ["4", ".", ".", "8", ".", "3", ".", ".", "1"],
    ["7", ".", ".", ".", "2", ".", ".", ".", "6"],
    [".", "6", ".", ".", ".", ".", "2", "8", "."],
    [".", ".", ".", "4", "1", "9", ".", ".", "5"],
    [".", ".", ".", ".", "8", ".", ".", "7", "9"]
]
```

solveSudoku(board)

print(board)

 # The board should now be solved

OUTPUT:-

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

=== Code Execution Successful ===
```

TIME COMPLEXITY:-O(n^2)