37. Sudoku Solver

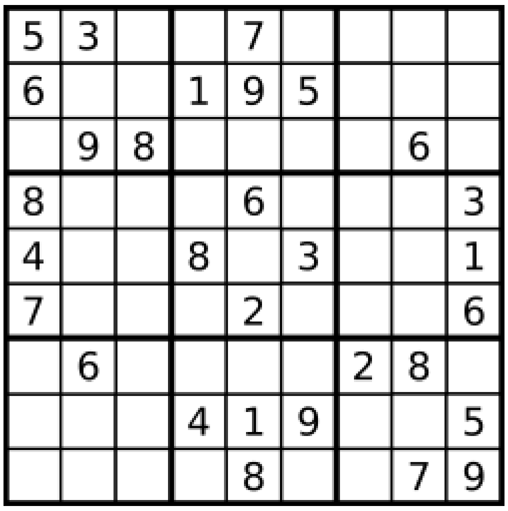
Write a program to solve a Sudoku puzzle by filling the empty cells.

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"]]
Explanation: The input board is shown above and the only valid solution is
shown below:

![](_resources/aae2a7b8ba27486eacd1f6ac53cd261f.png)
```

```
class Solution:
    def solveSudoku(self, board: List[List[str]]) -> None:
        """
        Do not return anything, modify board in-place instead.
        """
        board2 = board
        res =[]
        self.sudokuSolver(board,0,0,res)
        # print(res[0])
        # return res[0]
        # return board

    def sudokuSolver(self,board,i,j,res):
        if i==len(board):
            # res.append(board[:])
            return True

        new_row = 0
        new_col = 0
        if j==len(board)-1:
            new_row = i+1
            new_col = 0
        else:
            new_row = i
            new_col = j+1

        if board[i][j]!='.':
            return self.sudokuSolver(board,new_row,new_col,res)
        else:
            for val in range(1,10):
                if self.isValid(board,i,j,str(val)):
                    board[i][j] =  str(val)
                    if self.sudokuSolver(board,new_row,new_col,res):
                        return True
                    board[i][j] = '.'
```

```python
            return False

    def isValid(self,board,x,y,val):
        for i in range(0,9):
            if board[i][y]==val:
                return False
        for j in range(0,9):
            if board[x][j]==val:
                return False

        baseRow = 3*(x//3)
        baseCol = 3*(y//3)
        for i in range(0,3):
            for j in range(0,3):
                if board[baseRow+i][baseCol+j]==val:
                    return False

        return True
```