In [1]:

```
def print_grid(grid):
    for row in grid:
        print(" ".join(str(num) if num != 0 else '.' for num in row))
```

In [2]:

```
def is_valid(grid, row, col, num):
    # Check if num is not in the current row
    if num in grid[row]:
        return False

    # Check if num is not in the current column
    if num in (grid[i][col] for i in range(9)):
        return False

    # Check if num is not in the current 3x3 sub-grid
    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for i in range(start_row, start_row + 3):
        for j in range(start_col, start_col + 3):
            if grid[i][j] == num:
                return False

    return True

def find_empty_location(grid):
    for i in range(9):
        for j in range(9):
            if grid[i][j] == 0:
                return i, j
    return None

def solve_sudoku(grid):
    empty_location = find_empty_location(grid)
    if not empty_location:
        return True  # Puzzle solved

    row, col = empty_location

    for num in range(1, 10):
        if is_valid(grid, row, col, num):
            grid[row][col] = num
            if solve_sudoku(grid):
                return True
            grid[row][col] = 0  # Undo the move

    return False
```

In [3]:

```
if __name__ == "__main__":
    grid = [
        [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]
    ]

    print("Original Sudoku:")
    print_grid(grid)

    if solve_sudoku(grid):
        print("\nSolved Sudoku:")
        print_grid(grid)
    else:
        print("No solution exists.")
```

```
Original Sudoku:
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

Solved Sudoku:
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
```

In [ ]: