What is a Sudoku Puzzle?

The goal of the Sudoku puzzle is to fill in every empty box with an integer between 1 and 9, making sure that each number from 1 to 9 appears once in each row, column, and each of the tiny 3 by 3 boxes with thick borders.

Steps to solve the Sudoku Puzzle in Python

- In order to begin using this strategy to solve the sudoku puzzle, we first tell the variable M (M*M) how big the 2D matrix is.
- The grid is then printed using the utility function (puzzle).
- Later it will assign num to the row and col.
- 'false' will be returned if the same number is found in the same row, column, or 3*3 matrix specifically.
- Then, in order to prevent further backtracking, we will determine if we have reached the 8th row and 9th column and return true.
- Next, we will check if the column value becomes 9 then we move to the next row and column.
- Further, we check to see if the value in the grid's current position is larger than 0, and if it is, we iterate for the subsequent column.
- After checking if it is a safe place, we move to the next column and then assign the num in the current (row, col) position of the grid. Later we check for the next possibility with the next column.
- As our assumption was wrong, we discard the assigned num and then we go for the next assumption with a different num value

Implementing the Sudoku Solver in Python

- We'll use the backtracking method to create our sudoku solver in Python.
- Backtracking means switching back to the previous step as soon as we determine that our current solution cannot be continued into a complete one.
- We use this principle of backtracking to implement the sudoku algorithm. It's also called the brute force algorithm way to solve the sudoku puzzle.

```
M = 9

def puzzle(a):
    for i in range(M):
        for j in range(M):
            print(a[i][j],end = " ")
        print()


def solve(grid, row, col, num):
    for x in range(9):
        if grid[row][x] == num:
            return False

    for x in range(9):
        if grid[x][col] == num:
            return False


    startRow = row - row % 3
    startCol = col - col % 3
    for i in range(3):
        for j in range(3):
            if grid[i + startRow][j + startCol] == num:
                return False
    return True


def Suduko(grid, row, col):

    if (row == M - 1 and col == M):
        return True
    if col == M:
        row += 1
        col = 0
    if grid[row][col] > 0:
        return Suduko(grid, row, col + 1)
    for num in range(1, M + 1, 1):

        if solve(grid, row, col, num):

            grid[row][col] = num
            if Suduko(grid, row, col + 1):
                return True
        grid[row][col] = 0
    return False

'''0 means the cells where no value is assigned'''
```

```
grid = [[2, 5, 0, 0, 3, 0, 9, 0, 1],
        [0, 1, 0, 0, 0, 4, 0, 0, 0],
    [4, 0, 7, 0, 0, 0, 2, 0, 8],
    [0, 0, 5, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 9, 8, 1, 0, 0],
    [0, 4, 0, 0, 0, 3, 0, 0, 0],
    [0, 0, 0, 3, 6, 0, 0, 7, 2],
    [0, 7, 0, 0, 0, 0, 0, 0, 3],
    [9, 0, 3, 0, 0, 0, 6, 0, 4]]
print(" The input puzzle is: \n")
for i in grid:
  print(i)
print("\n")
if (Suduko(grid, 0, 0)):
    print("The Solution is given below: ")
    print("\n")
    puzzle(grid)
else:
    print("Solution does not exist:(")
```

```
 The input puzzle is:

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


The Solution is given below:


2 5 8 7 3 6 9 4 1
6 1 9 8 2 4 3 5 7
4 3 7 9 1 5 2 6 8
3 9 5 2 7 1 4 8 6
7 6 2 4 9 8 1 3 5
8 4 1 6 5 3 7 2 9
1 8 4 3 6 9 5 7 2
5 7 6 1 4 2 8 9 3
9 2 3 5 8 7 6 1 4
```