LACS-Elite-Part012

Solved Challenges 0/1

Back To Challenges List

Solve Sudoku

ID:12946 **Solved By 1 Users**

The program must accept an integer matrix of size **9x9** representing a sudoku as the input. The sudoku matrix contains the integers from 0 to 9 where **0** represents the **empty cells**. If the sudoku matrix is valid, the program must fill in the empty cells of the sudoku matrix and print it as the output. Else the program must print **Not Solved** as the output.

Sudoku:
Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid contain all of the digits from 1 to 9.

Input Format:
The first 9 lines each contain 9 integers separated by a space.

Output Format:
The first 9 lines each contain 9 integers separated by a space or the first line contains Not Solved.

Example Input/Output 1:
Input:
0 0 0 2 6 0 7 0 1
6 8 0 0 7 0 0 9 0
1 9 0 0 0 4 5 0 0
8 2 0 1 0 0 0 4 0
0 0 4 6 0 2 9 0 0
0 5 0 0 0 3 0 2 8
0 0 9 3 0 0 0 7 4
0 4 0 0 5 0 0 3 6
7 0 3 0 1 8 0 0 0

Output:
4 3 5 2 6 9 7 8 1
6 8 2 5 7 1 4 9 3
1 9 7 8 3 4 5 6 2
8 2 6 1 9 5 3 4 7
3 7 4 6 8 2 9 1 5
9 5 1 7 4 3 6 2 8

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

Example Input/Output 2:
Input:
0 6 0 3 0 0 8 0 4
5 3 7 0 9 0 0 0 0
0 4 0 0 0 6 3 0 7
0 9 0 0 5 1 2 3 8
0 0 0 0 0 0 0 0 0
7 1 3 6 2 0 0 4 0
3 0 6 4 0 0 0 1 0
0 0 0 0 6 0 5 2 3
1 0 2 0 0 3 0 8 0

Output:
Not Solved

Max Execution Time Limit: 500 millisecs

Ambiance

Python3 (3.x)

Reset

```
class Slot:

    def __init__(self):
        self.r=0
        self.c=0

def getFreeSlot(matrix):
    for row in range(R):
        for col in range(C):
            if(matrix[row][col]==0):
                slot = Slot()
                slot.r = row
                slot.c = col
                return slot
    return null


def solve(matrix):
    slot = Slot()
    slot = getFreeSlot(matrix)
    if(slot == null):
```

```
22          return True
23      for digit in range(1,10):
24          if(canFillRow(matrix, slot,digit) and canFillCol(matrix,s
                and canFillSubmatrix(matrix,slot, digit)):
25              matrix[slot.r][slot.c]=digit
26
27              if(solve(matrix)):
28                  return True
29              else:
30                  matrix[slot.r][slot.c] = 0
31
32      return False
33
34  def canFillRow(matrix, slot,digit):
35      for col in range(C):
36          if(matrix[slot.c][col] == digit):
37              return False
38      return True
39
40  def canFillCol(matrix,slot,digit):
41      for row in range(R):
42          if(matrix[row][slot.c] == digit):
43              return False
44      return True
45
46  def canFillSubmatrix(matrix,slot,digit):
47      startRow = (slot.r/3)*3
48      startCol = (slot.c/3)*3
49      for row in range(startRow, startRow+3):
50          for col in range(startCol, startCol+3):
51              if(matrix[row][col] == digit):
52                  return False
53
54      return True
55
56
57
58  R = 9
59  C = 9
60
61  matrix = []
62
63  for r in range(R):
64      row = list(map(int, input().strip().split()))
65      matrix.append(row)
66
67
68  if(solve(matrix)):
69      for row in range(R):
70          for col in range(C):
```

```
71              print(matrix[row][col], end=" ")
72          print()
73  else:
74      print("Not Solved")
75
76
77
78
79
```

Code did not pass the execution

TestCase ID: 86160

Input:

```
0 0 0 2 6 0 7 0 1
6 8 0 0 7 0 0 9 0
1 9 0 0 0 4 5 0 0
8 2 0 1 0 0 0 4 0
0 0 4 6 0 2 9 0 0
0 5 0 0 0 3 0 2 8
0 0 9 3 0 0 0 7 4
0 4 0 0 5 0 0 3 6
7 0 3 0 1 8 0 0 0
```

Expected Output:

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

Your Program Output:

```
Traceback (most recent call last):
  File "Hello.py", line 68, in
    if(solve(matrix)):
  File "Hello.py", line 20, in solve
    slot = slot.getFreeSlot(matrix)
AttributeError: 'Slot' object has no attribute 'getFreeSlot'
```

Save Run

Run with a custom test case (Input/Output)