

AI Lab Exp - 9

IMPLEMENTATION OF UNCERTAIN METHODS FOR AN APPLICATION

Submitted By

Name: VINOTH S

Reg No: RA1911030010103

Date: 05-04-2022

GitHub Link: <https://github.com/vk1308>

Aim:

To study the implementation of uncertain methods for an application and solve Sudoku using Python language.

Description:

Sudoku is a well-known puzzle game and popular for explaining search problems. Given an initial 9x9 grid of cells containing numbers between 1 and 9 or blanks, all blanks must be filled with numbers. You win Sudoku if you find all values such that every row, column, and 3x3 sub square contains the numbers 1–9, each with a single occurrence.

Diagram :

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

Code:

```
size = 9
```

```
#empty cells have value zero
```

```
matrix = [  
    [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 sudoku
```

```
def print_sudoku():
```

```
    for i in matrix:
```

```
        print (i)
```

```
#assign cells and check
```

```
def number_unassigned(row, col):
```

```
    num_unassign = 0
```

```
    for i in range(0,size):
```

```
        for j in range (0,size):
```

```
            #cell is unassigned
```

```
if matrix[i][j] == 0:
```

```
    row = i
```

```
    col = j
```

```
    num_unassign = 1
```

```
    a = [row, col, num_unassign]
```

```
    return a
```

```
a = [-1, -1, num_unassign]
```

```
return a
```

```
#check validity of number
```

```
def is_safe(n, r, c):
```

```
    #checking in row
```

```
    for i in range(0,size):
```

```
        #there is a cell with same value
```

```
        if matrix[r][i] == n:
```

```
            return False
```

```
    #checking in column
```

```
    for i in range(0,size):
```

```
        #there is a cell with same value
```

```
        if matrix[i][c] == n:
```

```
            return False
```

```
row_start = (r//3)*3
```

```
col_start = (c//3)*3;
```

```
#checking submatrix
```

```
for i in range(row_start,row_start+3):
```

```
    for j in range(col_start,col_start+3):  
        if matrix[i][j]==n:  
            return False  
    return True
```

#check validity of number

```
def solve_sudoku():
```

```
    row = 0
```

```
    col = 0
```

```
    #if all cells are assigned then the sudoku is already solved
```

```
    #pass by reference because number_unassigned will change the values of row and col
```

```
    a = number_unassigned(row, col)
```

```
    if a[2] == 0:
```

```
        return True
```

```
    row = a[0]
```

```
    col = a[1]
```

```
    #number between 1 to 9
```

```
    for i in range(1,10):
```

```
        #if we can assign i to the cell or not
```

```
        #the cell is matrix[row][col]
```

```
        if is_safe(i, row, col):
```

```
            matrix[row][col] = i
```

```
            #backtracking
```

```
            if solve_sudoku():
```

```
return True
```

```
#f we can't proceed with this solution
```

```
#reassign the cell
```

```
matrix[row][col]=0
```

```
return False
```

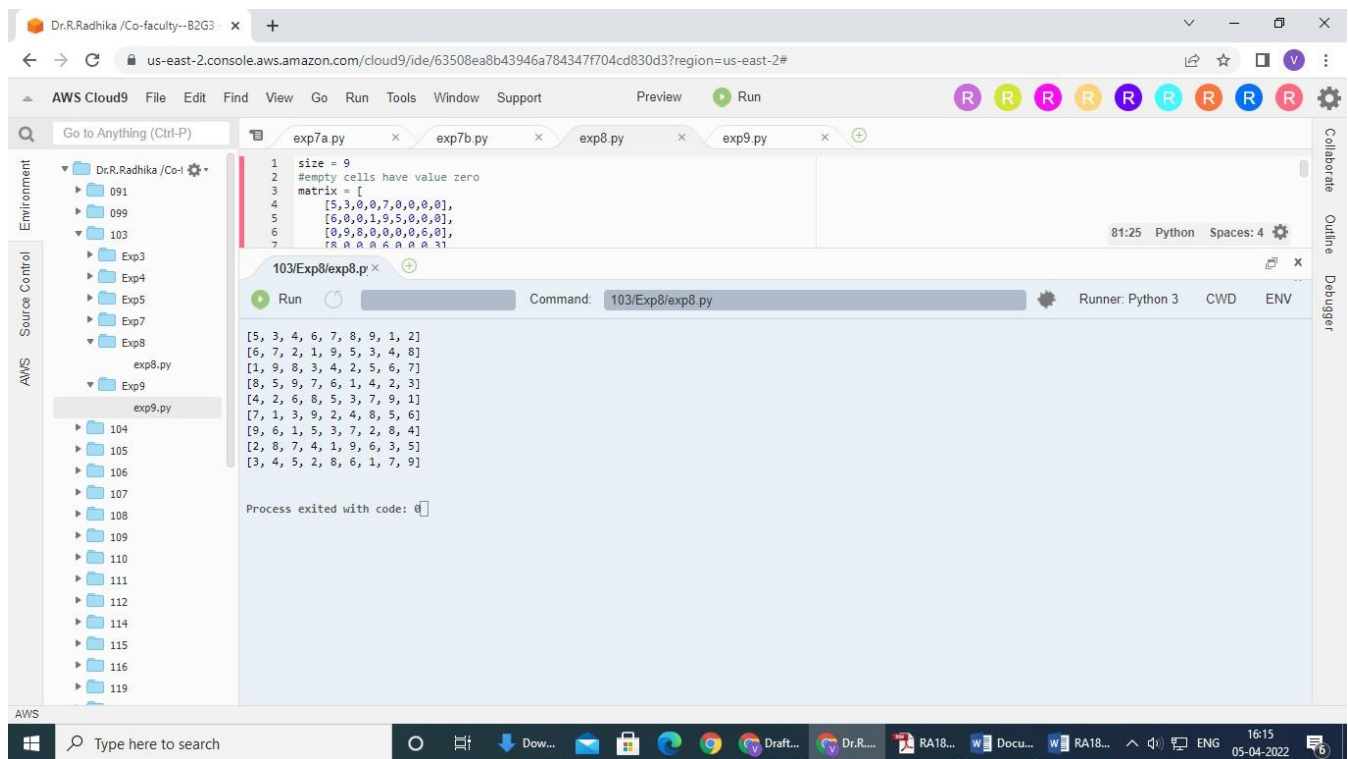
```
if solve_sudoku():
```

```
    print_sudoku()
```

```
else:
```

```
    print("No solution")
```

Output:



```
1 size = 9
2 #empty cells have value zero
3 matrix = [
4     [5,3,0,0,7,0,0,0,0],
5     [6,0,0,1,9,5,0,0,0],
6     [0,9,8,0,0,0,0,6,0],
7     [8,0,0,6,0,0,0,0,3],
8     [4,2,6,8,5,3,7,9,1],
9     [7,1,3,9,2,4,8,5,6],
10    [9,6,1,5,3,7,2,8,4],
11    [2,8,7,4,1,9,6,3,5],
12    [3,4,5,2,8,6,1,7,9]]
13
14 def solve_sudoku(matrix):
15     for row in range(9):
16         for col in range(9):
17             if matrix[row][col] == 0:
18                 for num in range(1,10):
19                     if is_valid(matrix, row, col, num):
20                         matrix[row][col] = num
21                         if solve_sudoku(matrix):
22                             return True
23                 matrix[row][col] = 0
24     return False
25
26 def is_valid(matrix, row, col, num):
27     for i in range(9):
28         if matrix[i][col] == num:
29             return False
30     for i in range(9):
31         if matrix[row][i] == num:
32             return False
33     sub_row = (row//3)*3
34     sub_col = (col//3)*3
35     for i in range(3):
36         for j in range(3):
37             if matrix[sub_row+i][sub_col+j] == num:
38                 return False
39     return True
40
41 if solve_sudoku(matrix):
42     print_sudoku(matrix)
43 else:
44     print("No solution")
```

103/Exp8/exp8.py

```
[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]
```

Process exited with code: 0

```
1 size = 9
2 #empty cells have value zero
3 matrix = [
4     [5,3,0,0,7,0,0,0,0],
5     [6,0,0,1,9,5,0,0,0],
6     [0,9,8,0,0,0,0,6,0],
7     [8,0,0,0,6,0,0,0,3],
8     [4,0,0,0,0,3,0,0,1],
9     [7,0,0,0,2,0,0,0,6],
10    [0,6,0,0,0,0,2,8,0],
11    [0,0,0,4,1,9,0,0,5],
12    [0,0,0,0,8,0,0,7,9]]
13
14 #print sudoku
15 def print_sudoku():
16     for i in matrix:
17         print(i)
18
19 #assign cells and check
20 def number_unassigned(row, col):
21     num_unassign = 0
22     for i in range(0,size):
23         for j in range(0,size):
24             #cell is unassigned
25             if matrix[i][j] == 0:
26                 row = i
27                 col = j
28                 num_unassign = 1
29                 a = [row, col, num_unassign]
30                 return a
31     a = [-1, -1, num_unassign]
32     return a
33
34 #check validity of number
35 def is_safe(n, r, c):
36     #checking in row
37     for i in range(0,size):
38         #check if a cell with same value
```

```
48     for j in range(col_start,col_start+3):
49         if matrix[i][j]==n:
50             return False
51     return True
52
53 #check validity of number
54 def solve_sudoku():
55     row = 0
56     col = 0
57     #if all cells are assigned then the sudoku is already solved
58     #pass by reference because number_unassigned will change the values of row and col
59     a = number_unassigned(row, col)
60     if a[2] == 0:
61         return True
62     row = a[0]
63     col = a[1]
64     #number between 1 to 9
65     for i in range(1,10):
66         #if we can assign i to the cell or not
67         #the cell is matrix[row][col]
68         if is_safe(i, row, col):
69             matrix[row][col] = i
70             #backtracking
71             if solve_sudoku():
72                 return True
73             #if we can't proceed with this solution
74             #reassign the cell
75             matrix[row][col]=0
76     return False
77
78 if solve_sudoku():
79     print_sudoku()
80 else:
81     print("No solution")
```

Result:

The given sudoku problem is solved using python language.