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

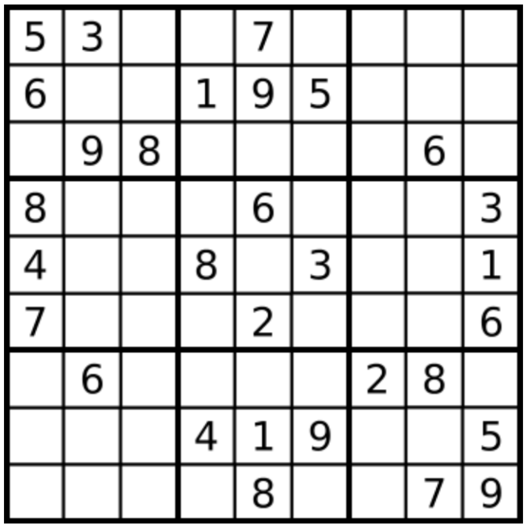
Aim:

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 :

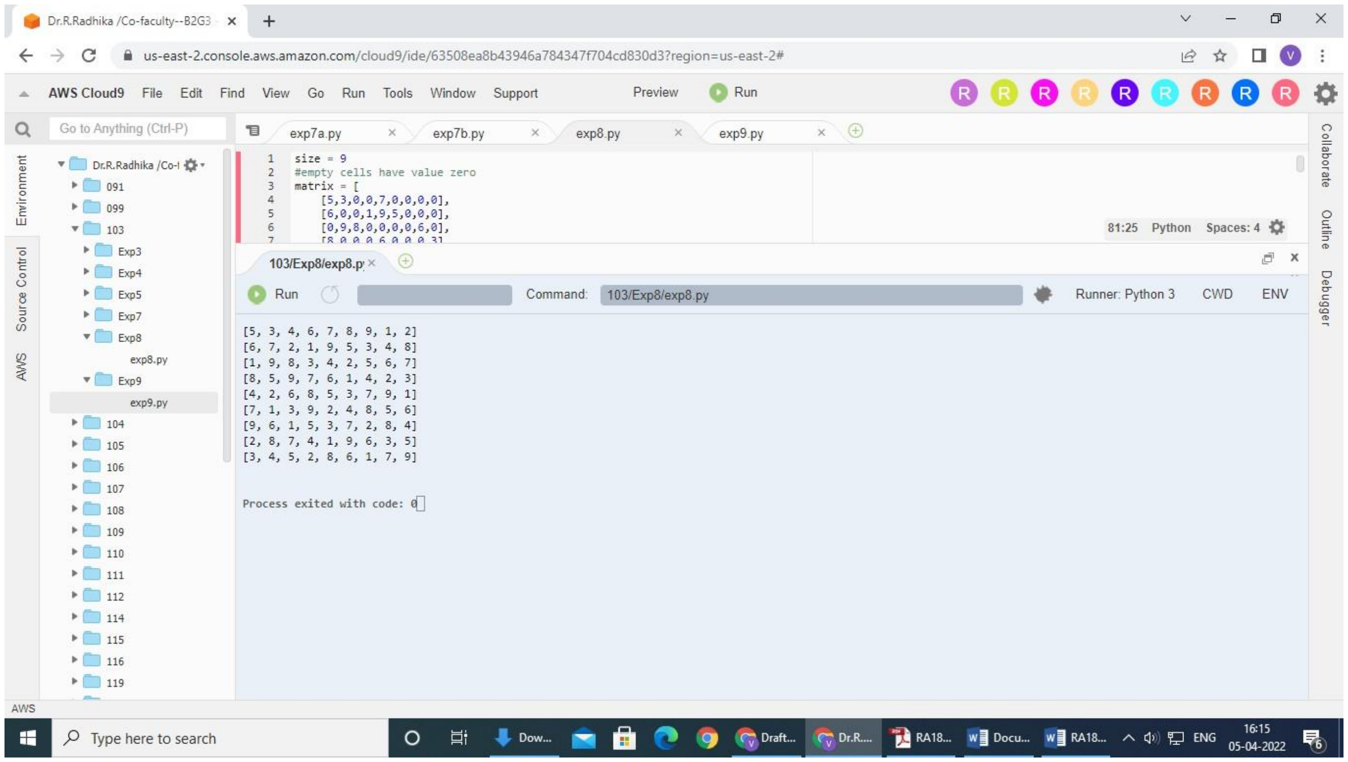


5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

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")
```

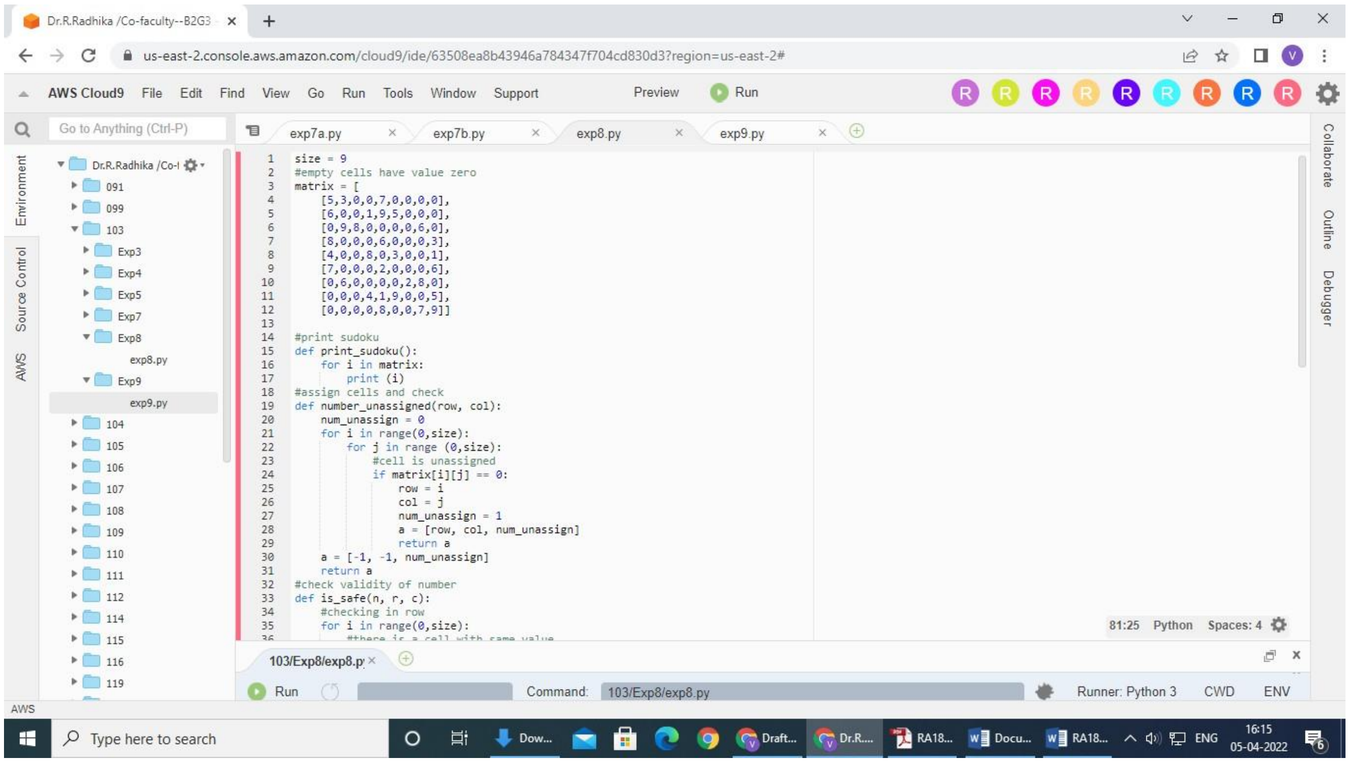
Output:



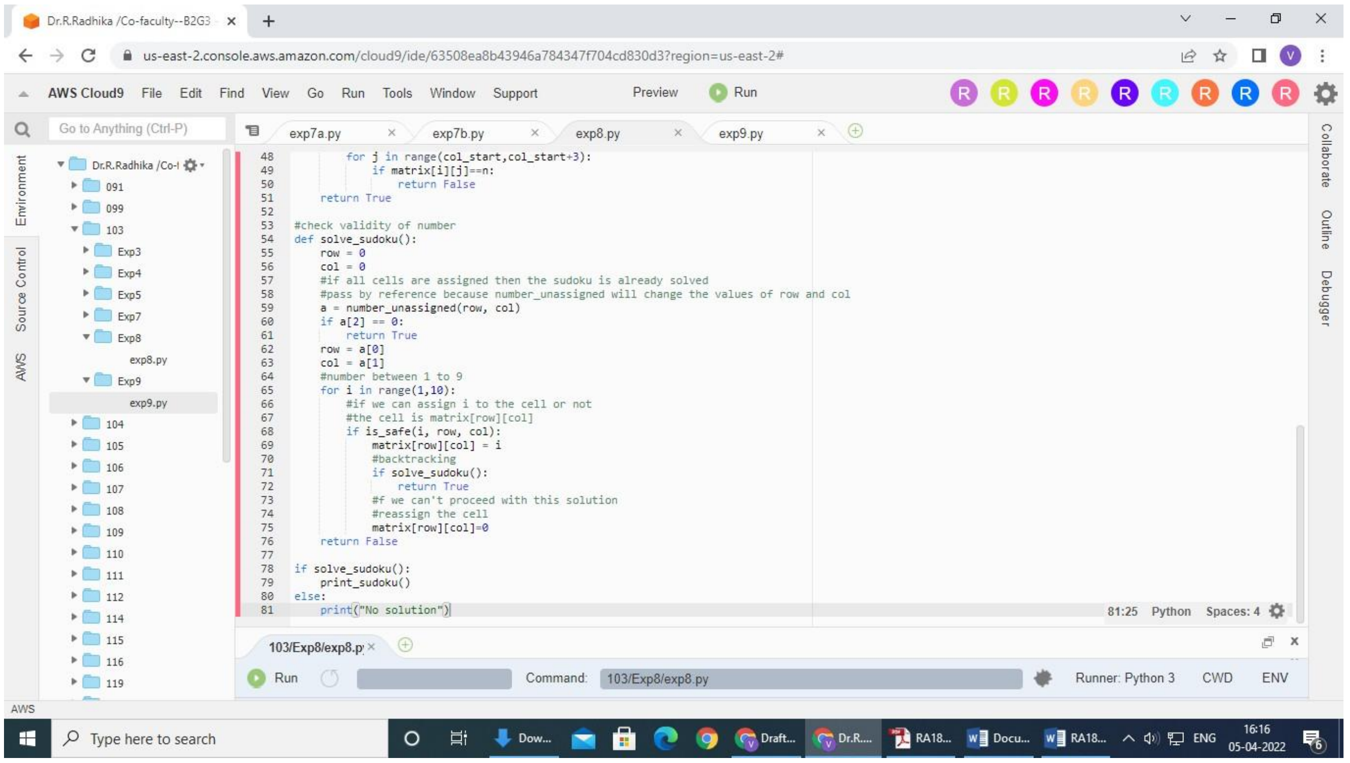
```
size = 9
#empty cells have value zero
matrix = [
    [5,3,0,0,7,0,0,0,0],
    [6,0,0,1,9,5,0,0,0],
    [0,9,8,0,0,0,0,6,0],
```

Command: 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
```



```python
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):
```



```python
        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")
```

Result:

The given sudoku problem is solved using python language.