

Experiment 9

Implementation of uncertain methods for an application

Submitted by-Rahul Goel

Register Number- RA1911030010094

Aim- To study Implementation of uncertain methods for an application.

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):

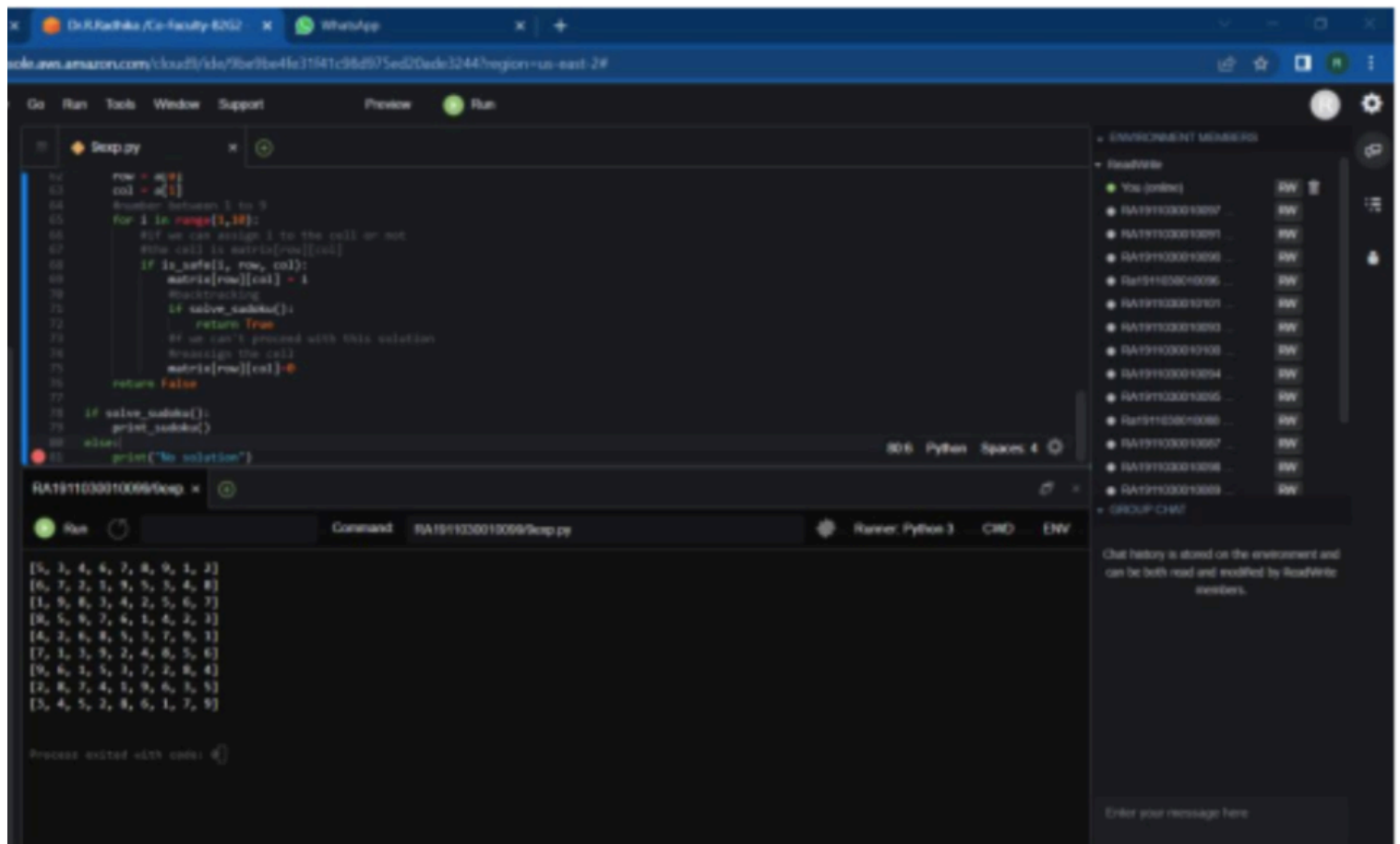
```

```
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:-



The screenshot displays a Jupyter Notebook environment with a dark theme. The main area shows a Python script for solving a Sudoku puzzle using a backtracking algorithm. The script defines a 9x9 grid and a function to check if a number can be placed at a specific position. The output of the script is a 9x9 grid representing the solved Sudoku puzzle.

```
def print_sudoku():
    row = 0
    col = 0
    # Number between 1 to 9
    for i in range(1,10):
        # If we can assign i to the cell or not
        # then call is matrix[row][col]
        if is_safe(i, row, col):
            matrix[row][col] = i
            # Backtracking
            if solve_sudoku():
                return True
            # If 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")
```

The output of the script is a 9x9 grid representing the solved Sudoku puzzle:

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

RESULT:-

Implementation of uncertain methods for an application was studied and implemented.