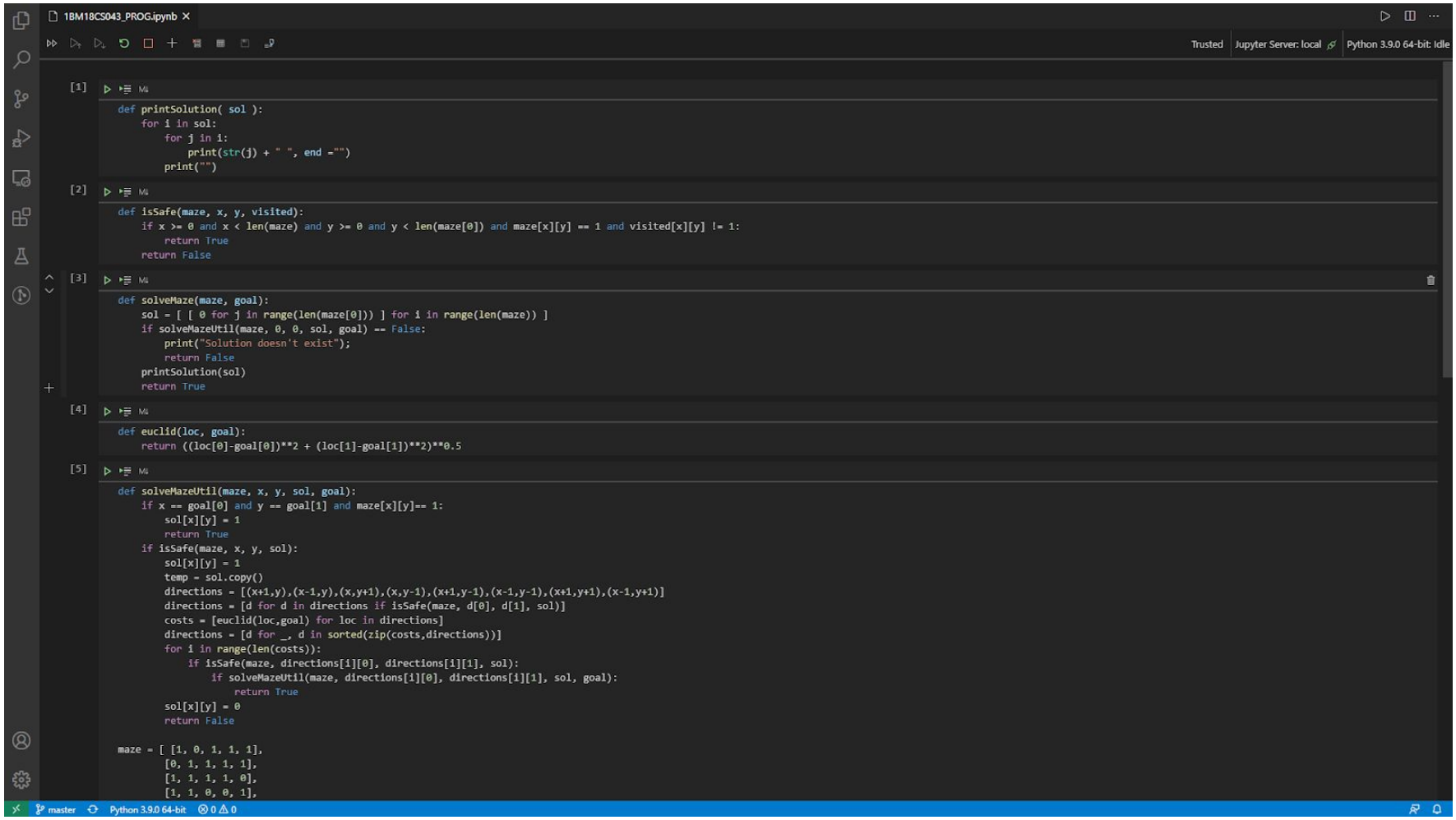
1BM18CS043_PROG.ipynb
Trusted
Jupyter Server: local
Python 3.9.0 64-bit: Idle
[1]
def printSolution(sol):
for i in sol:
for j in i:
print(str(j) + " ", end ="")
print("")
[2]
def isSafe(maze, x, y, visited):
if x >= 0 and x < len(maze) and y >= 0 and y < len(maze[0]) and maze[x][y] == 1 and visited[x][y] != 1:
return True
return False
[3]
def solveMaze(maze, goal):
sol = [[0 for j in range(len(maze[0]))] for i in range(len(maze))]
if solveMazeUtil(maze, 0, 0, sol, goal) == False:
print("Solution doesn't exist");
return False
printSolution(sol)
return True
[4]
def euclid(loc, goal):
return ((loc[0]-goal[0])**2 + (loc[1]-goal[1])**2)**0.5
[5]
def solveMazeUtil(maze, x, y, sol, goal):
if x == goal[0] and y == goal[1] and maze[x][y]== 1:
sol[x][y] = 1
return True
if isSafe(maze, x, y, sol):
sol[x][y] = 1
temp = sol.copy()
directions = [(x+1,y),(x-1,y),(x,y+1),(x,y-1),(x+1,y-1),(x-1,y-1),(x+1,y+1),(x-1,y+1)]
directions = [d for d in directions if isSafe(maze, d[0], d[1], sol)]
costs = [euclid(loc,goal) for loc in directions]
directions = [d for _, d in sorted(zip(costs,directions))]
for i in range(len(costs)):
if isSafe(maze, directions[i][0], directions[i][1], sol):
if solveMazeUtil(maze, directions[i][0], directions[i][1], sol, goal):
return True
sol[x][y] = 0
return False
maze = [[1, 0, 1, 1, 1],
[0, 1, 1, 1, 1],
[1, 1, 1, 1, 0],
[1, 1, 0, 0, 1],
master
Python 3.9.0 64-bit
0 0

1BM18CS043_PROG.ipynb

Trusted | Jupyter Server: local | Python 3.9.0 64-bit: Idle

```
    return ((loc[0]-goal[0])**2 + (loc[1]-goal[1])**2)**0.5
```

[5]

```
def solveMazeUtil(maze, x, y, sol, goal):
    if x == goal[0] and y == goal[1] and maze[x][y]== 1:
        sol[x][y] = 1
        return True
    if isSafe(maze, x, y, sol):
        sol[x][y] = 1
        temp = sol.copy()
        directions = [(x+1,y),(x-1,y),(x,y+1),(x,y-1),(x+1,y-1),(x-1,y-1),(x+1,y+1),(x-1,y+1)]
        directions = [d for d in directions if isSafe(maze, d[0], d[1], sol)]
        costs = [euclid(loc,goal) for loc in directions]
        directions = [d for _, d in sorted(zip(costs,directions))]
        for i in range(len(costs)):
            if isSafe(maze, directions[i][0], directions[i][1], sol):
                if solveMazeUtil(maze, directions[i][0], directions[i][1], sol, goal):
                    return True
        sol[x][y] = 0
        return False

maze = [ [1, 0, 1, 1, 1],
         [0, 1, 1, 1, 1],
         [1, 1, 1, 1, 0],
         [1, 1, 0, 0, 1],
         [1, 0, 1, 1, 1] ]

solveMaze(maze, (4,2))
```

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 1 0 0 0
0 0 1 0 0

True
```

[-]

master | Python 3.9.0 64-bit | ⊗ 0 ⚠ 0