def print\_grid(arr):

for i in range(9):

for j in range(9):

# 注意，在py3.x中，print函数默认都有换行

print(arr[i][j], end="")

print()

# 找出目前没有被赋值的位置，若全部都被填满，则返回False

def find\_empty\_location(arr, l):

for row in range(9):

for col in range(9):

if arr[row][col] == 0:

l[0] = row

l[1] = col

# print("empty: row="+str(row)+" col="+str(col))

return True

return False

# 找出num在该arr的row行是否出现过

def used\_in\_row(arr, row, num):

for i in range(9):

if arr[row][i] == num:

return True

return False

# 找出num在该arr的col列是否出现过

def used\_in\_col(arr, col, num):

for i in range(9):

if arr[i][col] == num:

return True

return False

# 找出num在该arr的3x3-box是否出现过，更应注意的是，传参技巧！

def used\_in\_box(arr, row, col, num):

for i in range(3):

for j in range(3):

if arr[row+i][col+j] == num:

return True

return False

def check\_location\_is\_safe(arr, row, col, num):

return not used\_in\_row(arr, row, num) and not used\_in\_col(arr, col, num) and not used\_in\_box(arr, row - row % 3, col - col % 3, num)

def solve\_sudoku(arr):

# 当前搜索的第几行、第几列

l = [0, 0]

# 找出还未被填充的位置

if not find\_empty\_location(arr, l):

return True

# 未被填充的位置，赋值给row，col

row = l[0]

col = l[1]

for num in range(1, 10):

if check\_location\_is\_safe(arr, row, col, num):

arr[row][col] = num

#print\_grid(arr)

if solve\_sudoku(arr):

return True

# 若当前num导致未来并没有结果，则当前所填充的数无效，置0后选下一个数测试

arr[row][col] = 0

return False

if \_\_name\_\_ == "\_\_main\_\_":

grid = [[0 for x in range(9)] for y in range(9)]

grid = [[3, 0, 6, 5, 0, 8, 4, 0, 0],

[5, 2, 0, 0, 0, 0, 0, 0, 0],

[0, 8, 7, 0, 0, 0, 0, 3, 1],

[0, 0, 3, 0, 1, 0, 0, 8, 0],

[9, 0, 0, 8, 6, 3, 0, 0, 5],

[0, 5, 0, 0, 9, 0, 6, 0, 0],

[1, 3, 0, 0, 0, 0, 2, 5, 0],

[0, 0, 0, 0, 0, 0, 0, 7, 4],

[0, 0, 5, 2, 0, 6, 3, 0, 0]]

if solve\_sudoku(grid):

print\_grid(grid)

else:

print("No solution exists\n")

# output

# 3 1 6 5 7 8 4 9 2

# 5 2 9 1 3 4 7 6 8

# 4 8 7 6 2 9 5 3 1

# 2 6 3 4 1 5 9 8 7

# 9 7 4 8 6 3 1 2 5

# 8 5 1 7 9 2 6 4 3

# 1 3 8 9 4 7 2 5 6

# 6 9 2 3 5 1 8 7 4

# 7 4 5 2 8 6 3 1 9