数独（sudoku）盘面是个九宫，每一宫又分为九个小格。在这八十一格中给出一定的已知数字和解题条件，利用逻辑和推理，在其他的空格上填入1-9的数字。使1-9每个数字在每一行、每一列和每一宫中都只出现一次，所以又称“九宫格”。

明白了原理之后就很简单了，首先我们在代码里生成一个9x9的二维数组。如下图所示，因为第1、第5、第9宫格之间是没有关联的，只需要保证这三个宫格之中都包含1到9的数字就只可以了，所以我们可以生成3组1-9的数字，分别将3组数字打乱顺序后填入蓝色格式，如下图所示。

9x9二维数组

接着我们只需要遍历每个个格子，排除不可填的数字之后，随机给它填入一个数字，如果发现当前格子没有可填的值，则退回去上一个格子，将此格子原有的值排除后再重新选一个新的值填入，如果发现也没有合适的值，再退一个格子....在不停的重复这个过程之后，就可以得到一个正确的数独矩阵，这里用到了深度优先算法和回溯算法。

以下为示例代码：

import random

import math

matrix = []

# 生成一个随机的数组

def get\_random\_unit():

\_num\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]

random.shuffle(\_num\_list)

return \_num\_list

def print\_grid(arr):

for i in range(9):

print(arr[i])

def get\_row(row):

row\_arr = []

for v in matrix[row]:

if v == 0:

continue

row\_arr.append(v)

return row\_arr

def get\_col(col):

col\_arr = []

for i in range(9):

val = matrix[i][col]

if val == 0:

continue

col\_arr.append(matrix[i][col])

return col\_arr

def get\_block(num):

col\_arr = []

seq = num % 3

col\_end = 9 if seq == 0 else seq \* 3

row\_end = int(math.ceil(num / 3) \* 3)

for i in range(row\_end - 3, row\_end):

for j in range(col\_end - 3, col\_end):

val = matrix[i][j]

if val != 0:

col\_arr.append(matrix[i][j])

return col\_arr

def get\_block\_seq(row, col):

col\_seq = int(math.ceil((col + 0.1) / 3))

row\_seq = int(math.ceil((row + 0.1) / 3))

return 3 \* (row\_seq - 1) + col\_seq

def get\_enable\_arr(row, col):

avail\_arr = get\_random\_unit()

seq = get\_block\_seq(row, col)

block = get\_block(seq)

row = get\_row(row)

col = get\_col(col)

unable\_arr = list(set(block + row + col))

for v in unable\_arr:

if v in avail\_arr:

avail\_arr.remove(v)

return avail\_arr

def main():

can\_num = {}

count = 0

# 初始化一个9行9列的数组

for i in range(9):

matrix.append([0] \* 9)

num\_list = get\_random\_unit()

for row in range(3):

for col in range(3):

matrix[row][col] = num\_list.pop(0)

num\_list = get\_random\_unit()

for row in range(3, 6):

for col in range(3, 6):

matrix[row][col] = num\_list.pop(0)

num\_list = get\_random\_unit()

for row in range(6, 9):

for col in range(6, 9):

matrix[row][col] = num\_list.pop(0)

box\_list = []

for row in range(9):

for col in range(9):

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

box\_list.append({'row': row, 'col': col})

i = 0

while i < len(box\_list):

count += 1

position = box\_list[i]

row = position['row']

col = position['col']

key = '%dx%d' % (row, col)

if key in can\_num:

enable\_arr = can\_num[key]

else:

enable\_arr = get\_enable\_arr(row, col)

can\_num[key] = enable\_arr

if len(enable\_arr) <= 0:

i -= 1

if key in can\_num:

del (can\_num[key])

matrix[row][col] = 0

continue

else:

matrix[row][col] = enable\_arr.pop()

i += 1

print\_grid(matrix)

print(count)

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

main()