**实验一 熟悉Numpy库**

**姓名：李坤璘 班级学号：20智能03 2019202216**

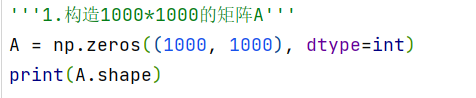
**一、实验目的：熟悉Numpy的数组操作**

**二、实验条件：PC计算机一台以及Python3.8环境**

**三、实验原理：Numpy是Python中的科学计算计算包，利用通函数可以快速地进行各种运算以及随机过程模拟**

**四、实验内容**

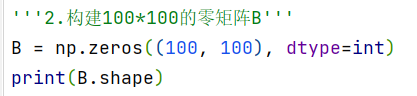
**1.** **构造1000\*1000的矩阵A**





输出结果证明为1000\*1000大小的矩阵。

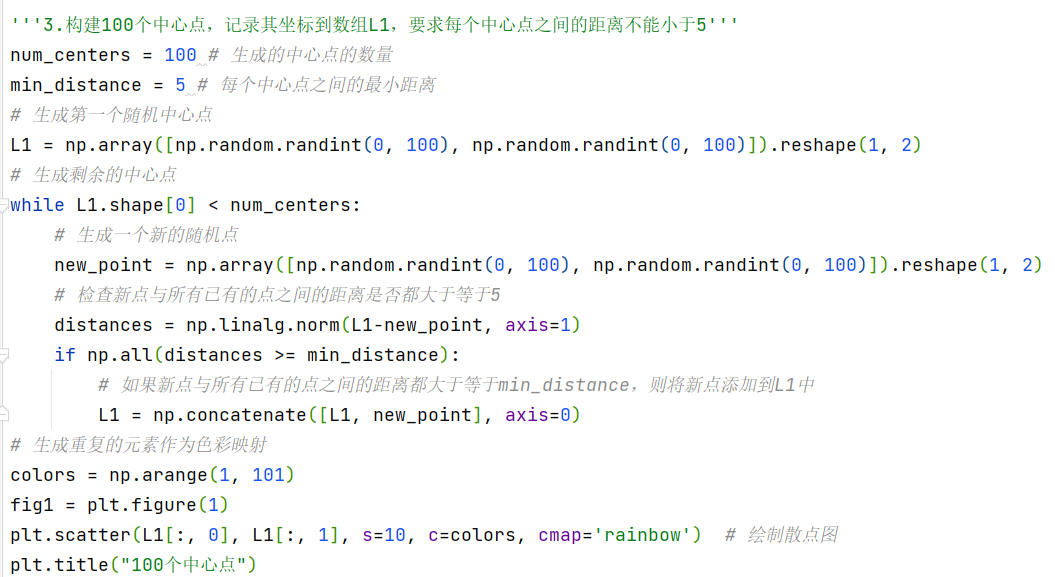
**2.** **构建100\*100的零矩阵B**

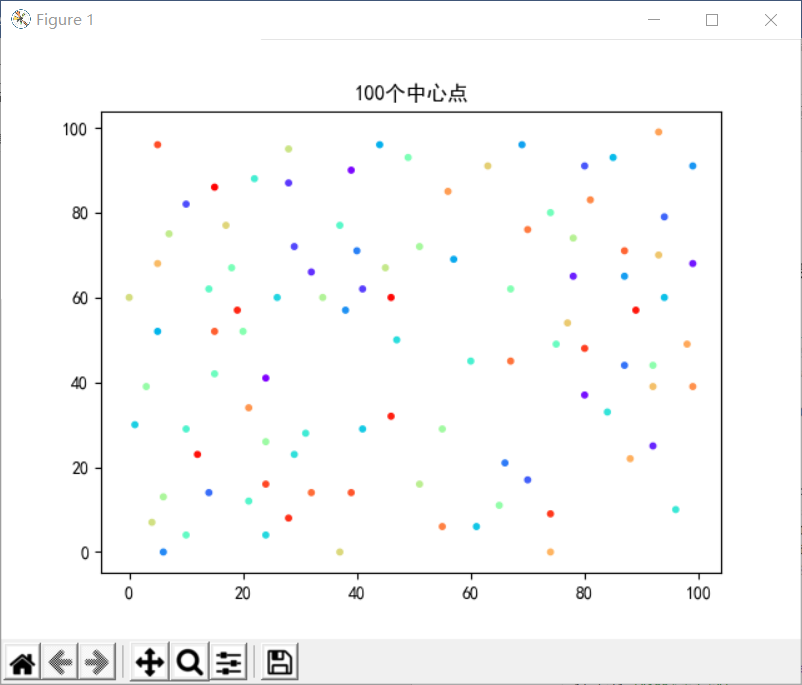




输出结果证明为100\*100大小的矩阵。

**3.** **构建100个中心点，记录其坐标到数组L1，要求每个中心点之间的距离不能小于5。**



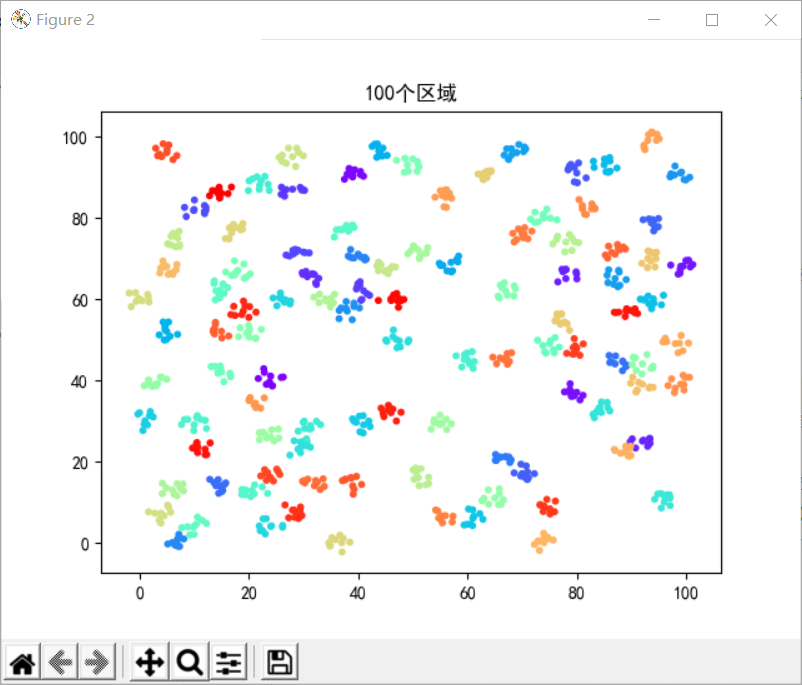


100个中心点如上图所示

**4.** **在每个中心点附近各生成9个点，使其与对应中心的距离不大于2.5**



100个中心点以及其生成的聚类如下图所示：

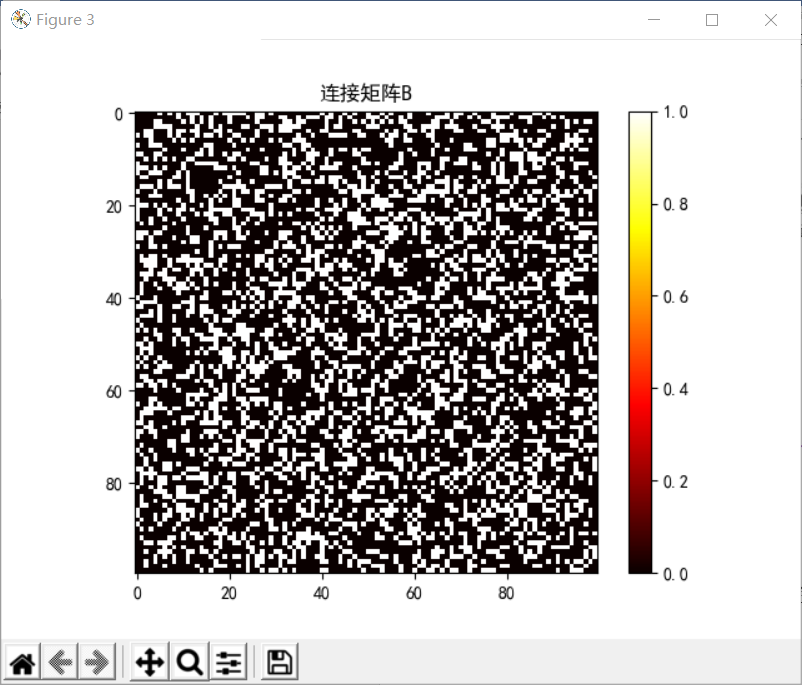


**5. 随机生成中心点对称的双向连接**



下方将矩阵可视化可以看出白色为连接，黑色为未连接；

输出True证明矩阵对称

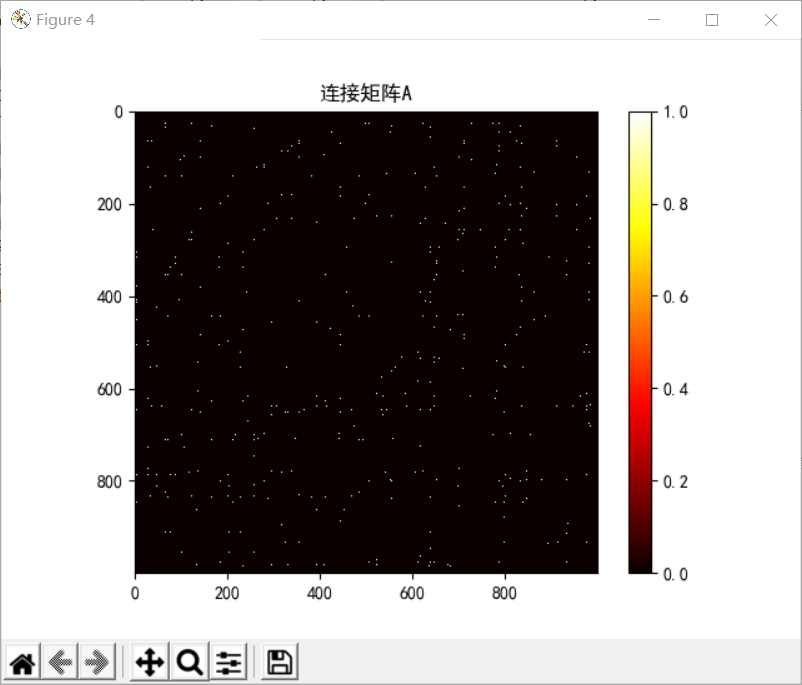


**6. 获得连接矩阵A'**

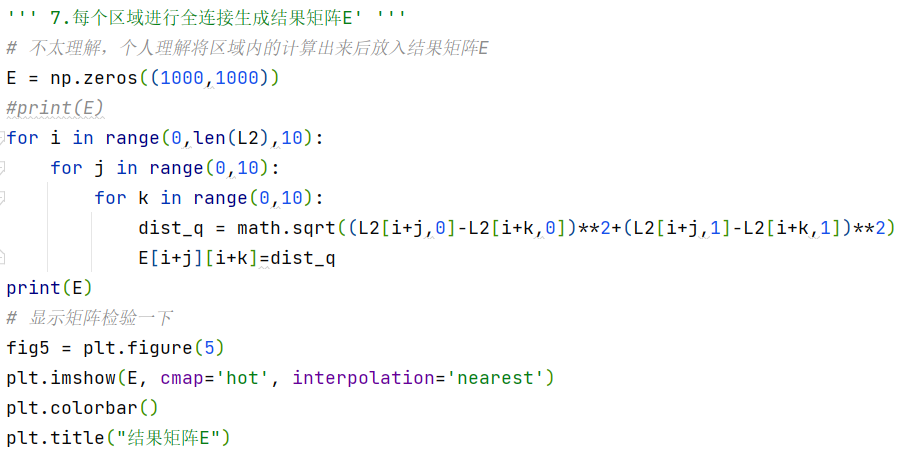


下方将矩阵可视化可以看出白色为连接，黑色为未连接；

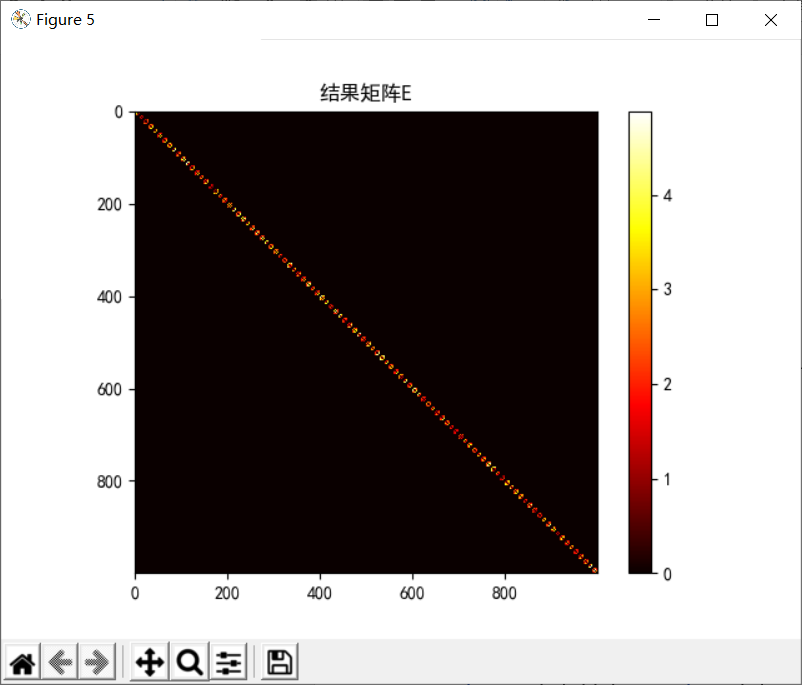
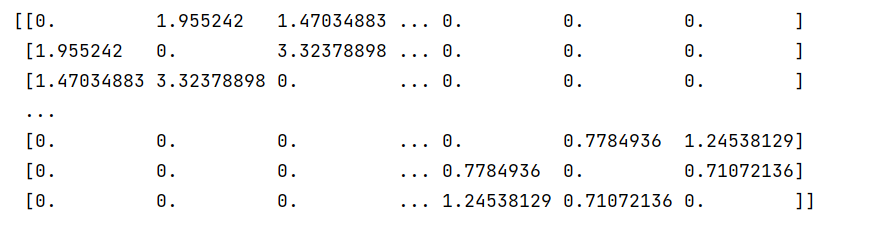
输出True证明矩阵对称

**7.每个区域进行全连接生成结果矩阵E’**



输出全连接的结果矩阵以及可视化如下所示：



**五、完整代码**

**import numpy as np  
from matplotlib import pyplot as plt  
import scipy.spatial.distance as dist  
import math  
plt.rcParams['font.sans-serif'] = ['SimHei'] *# 显示中文标题*'''1.构造1000\*1000的矩阵A'''  
A = np.zeros((1000, 1000), dtype=int)  
print(A.shape)  
  
'''2.构建100\*100的零矩阵B'''  
B = np.zeros((100, 100), dtype=int)  
print(B.shape)  
  
'''3.构建100个中心点，记录其坐标到数组L1，要求每个中心点之间的距离不能小于5'''  
num\_centers = 100 *# 生成的中心点的数量*min\_distance = 5 *# 每个中心点之间的最小距离  
# 生成第一个随机中心点*L1 = np.array([np.random.randint(0, 100), np.random.randint(0, 100)]).reshape(1, 2)  
*# 生成剩余的中心点*while L1.shape[0] < num\_centers:  
 *# 生成一个新的随机点* new\_point = np.array([np.random.randint(0, 100), np.random.randint(0, 100)]).reshape(1, 2)  
 *# 检查新点与所有已有的点之间的距离是否都大于等于5* distances = np.linalg.norm(L1-new\_point, axis=1)  
 if np.all(distances >= min\_distance):  
 *# 如果新点与所有已有的点之间的距离都大于等于min\_distance，则将新点添加到L1中* L1 = np.concatenate([L1, new\_point], axis=0)  
*# 生成重复的元素作为色彩映射*colors = np.arange(1, 101)  
fig1 = plt.figure(1)  
plt.scatter(L1[:, 0], L1[:, 1], s=10, c=colors, cmap='rainbow') *# 绘制散点图*plt.title("100个中心点")  
  
'''4.在每个中心点附近各生成9个点，使其与对应中心的距离不大于2.5'''  
*# 初始化L2数组和编号变量*L2 = np.zeros((1000, 2))  
count = 0  
*# 循环遍历每个中心点*for i in range(100):  
 L2[count] = L1[i] *# 第1个，11个，21个以此类推的点均为L1中的中心点* count += 1  
 *# 中心周围总共生成9个点* for j in range(9):  
 while True:  
 *# 在中心点附近随机生成一个点(这里选择D8距离为2.5周边的点)* x, y = np.random.uniform(L1[i, 0] - 2.5, L1[i, 0] + 2.5), np.random.uniform(L1[i, 1] - 2.5, L1[i, 1] + 2.5)  
 *# 判断该点与中心点之间的距离是否小于等于2.5* if np.linalg.norm(L1[i, :] - [x, y]) <= 2.5:  
 *# 如果小于等于2.5，则将该点存储到L2数组中，并为其赋予一个编号* L2[count] = [x, y]  
 count += 1  
 break  
*# 生成重复的元素作为色彩映射*colors = np.repeat(np.arange(1, 101), 10)  
fig2 = plt.figure(2)  
plt.scatter(L2[:, 0], L2[:, 1], s=10, c=colors, cmap='rainbow') *# 绘制散点图*plt.title("100个区域")  
print(L2)  
  
'''5.随机生成中心点对称的双向连接'''  
a = 0.7 *# 矩阵B中元素为1的比例  
# 生成随机连接矩阵B*B = np.triu(np.random.random((100, 100)) <= a/2, k=1) *# 生成上三角随机矩阵*B = (B + B.T).astype(int) *# 转换成对称矩阵  
# 显示矩阵检验一下*fig3 = plt.figure(3)  
plt.imshow(B, cmap='hot', interpolation='nearest')  
plt.colorbar()  
plt.title("连接矩阵B")  
*# 检验是否是对称矩阵*print(np.allclose(B, B.T))  
  
''' 6.获得连接矩阵A' '''  
num\_point = 10  
*# 根据连接矩阵B'和欧式距离计算找到对应的区域中距离最近的两个点，并在零矩阵A中将该位置标记为1*for i in range(num\_centers):  
 for j in range(i+1, num\_centers):  
 if B[i][j] == 1:  
 *# 0~9 10~19 20~29 (i-1)\*10~i\*10-1  
 # X和Y代表那两个中心点以及其聚类中的相关点的集合* X = L2[i\*10:(i+1)\*10]  
 Y = L2[j\*10:(j+1)\*10]  
 *# 获取X中每个点到Y中每个点之间的距离* d = dist.cdist(X,Y)  
 *# 找到距离最近的点的坐标* index = np.where(d == np.min(d))  
 x,y = int(index[0]),int(index[1]) *# X中的第x+1个点和Y中的第y+1个点  
 # 找到连接矩阵对应位置* x = (i-1)\*10 + x  
 y = (j-1)\*10 + y  
 A[x,y],A[y,x] = 1,1  
*# 显示矩阵检验一下*fig4 = plt.figure(4)  
plt.imshow(A, cmap='hot', interpolation='nearest')  
plt.colorbar()  
plt.title("连接矩阵A")  
*# 检验是否是对称矩阵*print(np.allclose(A, A.T))  
  
''' 7.每个区域进行全连接生成结果矩阵E' '''  
*# 不太理解，个人理解将区域内的计算出来后放入结果矩阵E*E = np.zeros((1000,1000))  
*#print(E)*for i in range(0,len(L2),10):  
 for j in range(0,10):  
 for k in range(0,10):  
 dist\_q = math.sqrt((L2[i+j,0]-L2[i+k,0])\*\*2+(L2[i+j,1]-L2[i+k,1])\*\*2)  
 E[i+j][i+k]=dist\_q  
print(E)  
*# 显示矩阵检验一下*fig5 = plt.figure(5)  
plt.imshow(E, cmap='hot', interpolation='nearest')  
plt.colorbar()  
plt.title("结果矩阵E")  
  
  
plt.show() *# 显示图形***

**六、实验总结**

1.Numpy提供了一种高效的方法来进行数值计算和数组操作，它可以高效地处理大型多维数组和矩阵，提供了简单的方法来进行数学运算，如矩阵乘法、逆、转置等。在本实验中，我们学习了如何使用 Numpy 数组进行快速数组操作。

2.这次实验学习了如何创建一个 Numpy 数组，并使用数组的轴来获取和操作它的元素。我们还学习了如何更改数组的形状和大小，以及如何在数组中添加或删除元素。

3.使用 Numpy 数组进行算术计算非常容易，它支持常见的算术运算，如加、减、乘、除、平方等。还有这次使用了线性代数中的二次范式表示距离的计算，提高了代码书写效率

4.在本实验中还学习了如何使用 Numpy 的切片和索引来选择数组中的子集和元素，并进行不同的操作。

总结来说，Numpy的矩阵很厉害，不同于列表，其内置函数相信在接下来的实验中会使用越来越多，直到完全熟练。