实验二 《数据统计和可视化》

题目

基于实验一中清洗后的数据练习统计和视化操作，100个同学（样本），每个同学有11门课程的成绩（11维的向量）；那么构成了一个100x11的数据矩阵。以你擅长的语言C/C++/Java/Python/Matlab，编程计算：

库：

import pyecharts.options as opts # for 散点图  
import pylab as pl  
import heapq # for max值

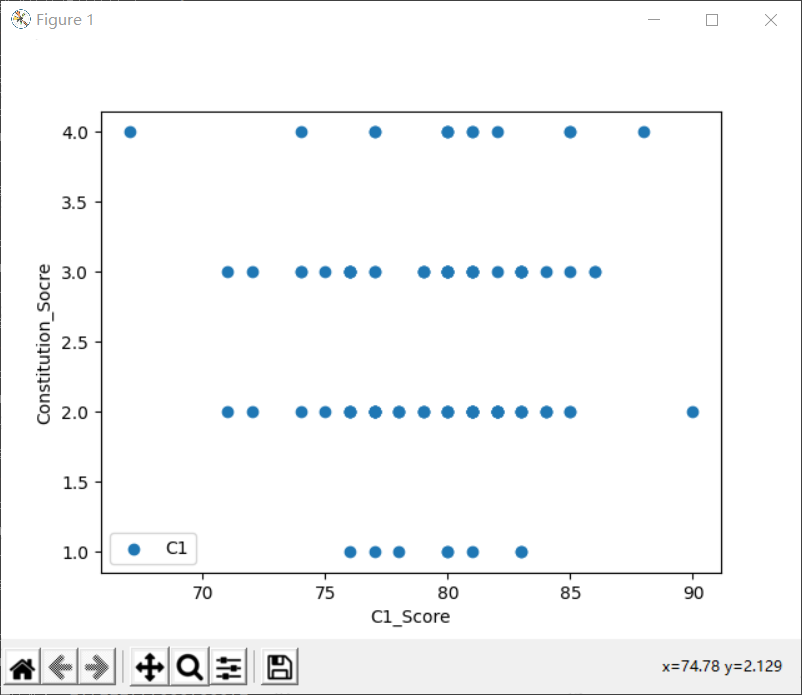
1. 请以课程1成绩为x轴，体能成绩为y轴，画出散点图。

代码如下：（数据使用了实验一中的数据）

from matplotlib import pyplot as plt  
# 加上横轴与纵轴标签  
plt.xlabel("C1\_Score")  
plt.ylabel("Constitution\_Socre")  
# 绘制散点图，并且加上标签  
plt.scatter(C1\_Score, Constitution\_Socre, label="C1")  
plt.legend()  
plt.show()

截图：bad/general/good/excellent分别为1分/2分/3分/4分

因为数据有一样的部分，所以显示出来的点比实际少。



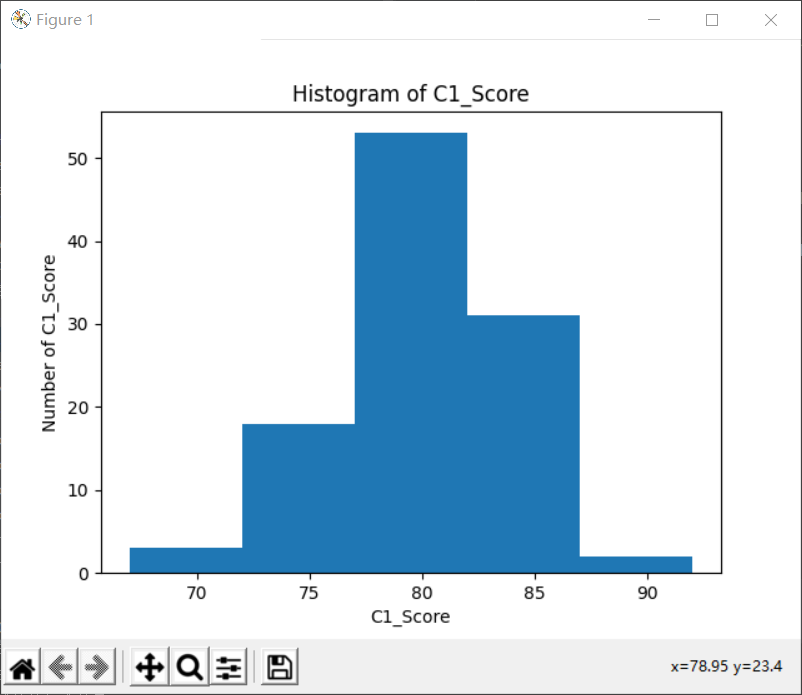
1. 以5分为间隔，画出课程1的成绩直方图。

代码如下：

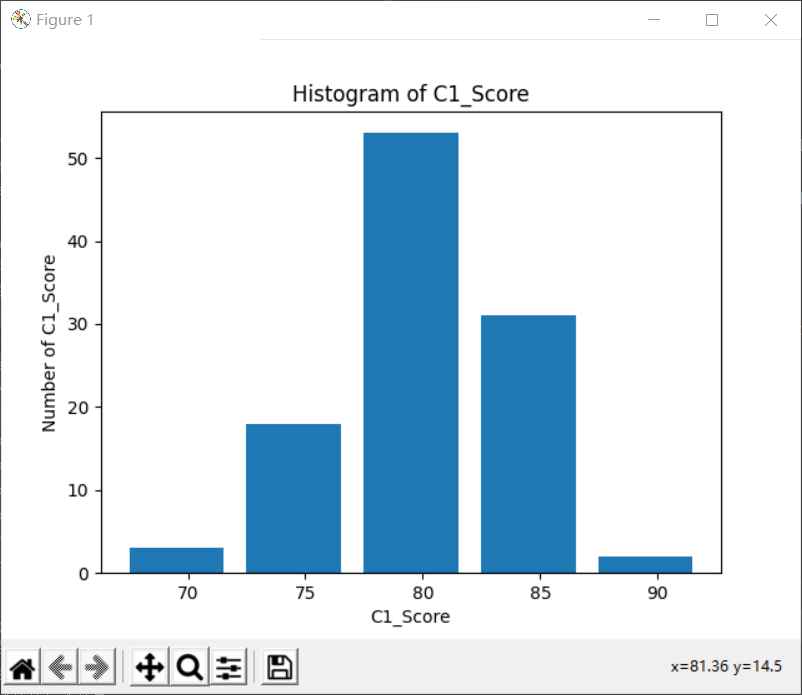
# 以5分为间隔，画出课程1的成绩直方图。  
Score\_OF\_C1 = []  
for m in range(len(d)): # 计算C1的总成绩  
 if d[m][5] != '':  
 Score\_OF\_C1.append(int(d[m][5]))  
Score\_OF\_C1.sort()  
print(Score\_OF\_C1)  
print(len(Score\_OF\_C1))  
bins = [67, 72, 77, 82, 87, 92] #设置间隔  
pl.hist(Score\_OF\_C1, bins, rwidth=0.8)  
pl.xlabel('C1\_Score')  
pl.ylabel('Number of C1\_Score')  
pl.title('Histogram of C1\_Score')  
pl.show()

截图：

不设置柱体的宽度的时候：



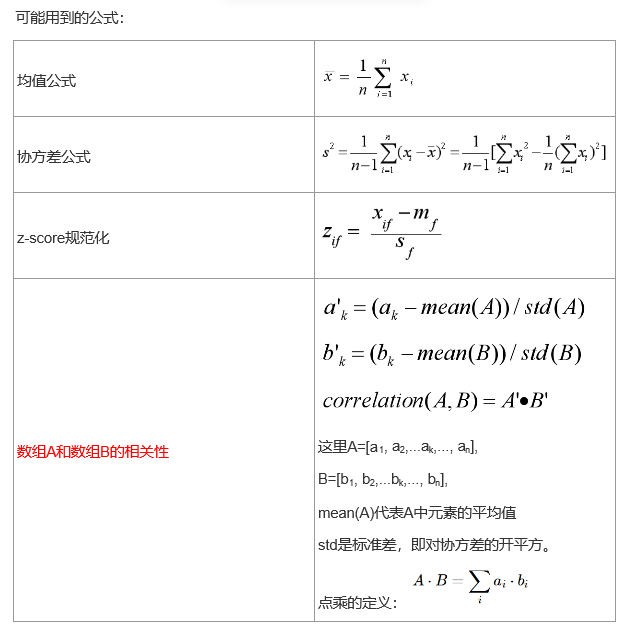
设置柱体的宽度的时候：



1. 对每门成绩进行z-score归一化，得到归一化的数据矩阵。

代码如下：

我的计算是基于老师给的文档。

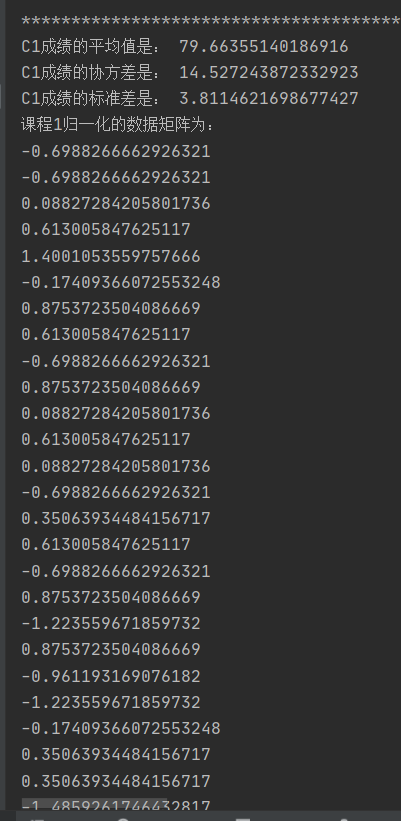


# 3.对每门成绩进行z-score归一化，得到归一化的数据矩阵。  
  
# 计算C1的z-score  
C1\_z\_score = []  
for i in range(len(d)):  
 if d[i][5] != '':  
 C1\_z\_score.append(float(d[i][5]))  
# 求协方差  
ss\_z\_C1 = 0.0  
mean\_z\_C1 = mean(C1\_z\_score) # 成绩的平均值  
for i in range(len(C1\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C1 += (C1\_z\_score[i] - mean\_z\_C1) \* (C1\_z\_score[i] - mean\_z\_C1)  
ss\_z\_C1 = ss\_z\_C1/(len(C1\_z\_score) - 1)  
std\_z\_C1 = ss\_z\_C1 \*\* 0.5  
print("C1成绩的平均值是：", mean\_z\_C1)  
print("C1成绩的协方差是：", ss\_z\_C1)  
print("C1成绩的标准差是：", std\_z\_C1)  
C1\_z\_scoreFinal = []  
for m in range(len(C1\_z\_score)):   
 C1\_z\_scoreFinal.append((C1\_z\_score[m] - mean\_z\_C1)/std\_z\_C1)  
print("课程1归一化的数据矩阵为：")  
for x in range(len(C1\_z\_scoreFinal)):  
 print(C1\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
  
# 计算C2的z-score  
C2\_z\_score = []  
for i in range(len(d)):  
 if d[i][6] != '':  
 C2\_z\_score.append(float(d[i][6]))  
# 求协方差  
ss\_z\_C2 = 0.0  
mean\_z\_C2 = mean(C2\_z\_score) # 成绩的平均值  
for i in range(len(C2\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C2 += (C2\_z\_score[i] - mean\_z\_C2) \* (C2\_z\_score[i] - mean\_z\_C2)  
ss\_z\_C2 = ss\_z\_C2/(len(C2\_z\_score) - 1)  
std\_z\_C2 = ss\_z\_C2 \*\* 0.5  
print("C2成绩的平均值是：", mean\_z\_C2)  
print("C2成绩的协方差是：", ss\_z\_C2)  
print("C2成绩的标准差是：", std\_z\_C2)  
C2\_z\_scoreFinal = []  
for m in range(len(C2\_z\_score)):   
 C2\_z\_scoreFinal.append((C2\_z\_score[m] - mean\_z\_C2)/std\_z\_C2)  
print("课程2归一化的数据矩阵为：")  
for x in range(len(C2\_z\_scoreFinal)):  
 print(C2\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
  
# 计算C3的z-score  
C3\_z\_score = []  
for i in range(len(d)):  
 if d[i][7] != '':  
 C3\_z\_score.append(float(d[i][7]))  
# 求协方差  
ss\_z\_C3 = 0.0  
mean\_z\_C3 = mean(C3\_z\_score) # 成绩的平均值  
for i in range(len(C3\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C3 += (C3\_z\_score[i] - mean\_z\_C3) \* (C3\_z\_score[i] - mean\_z\_C3)  
ss\_z\_C3 = ss\_z\_C3/(len(C3\_z\_score) - 1)  
std\_z\_C3 = ss\_z\_C3 \*\* 0.5  
print("C3成绩的平均值是：", mean\_z\_C3)  
print("C3成绩的协方差是：", ss\_z\_C3)  
print("C3成绩的标准差是：", std\_z\_C3)  
C3\_z\_scoreFinal = []  
for m in range(len(C3\_z\_score)):  
 C3\_z\_scoreFinal.append((C3\_z\_score[m] - mean\_z\_C3)/std\_z\_C3)  
print("课程3归一化的数据矩阵为：")  
for x in range(len(C3\_z\_scoreFinal)):  
 print(C3\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
  
# 计算C4的z-score  
C4\_z\_score = []  
for i in range(len(d)):  
 if d[i][8] != '':  
 C4\_z\_score.append(float(d[i][8]))  
# 求协方差  
ss\_z\_C4 = 0.0  
mean\_z\_C4 = mean(C4\_z\_score) # 成绩的平均值  
for i in range(len(C4\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C4 += (C4\_z\_score[i] - mean\_z\_C4) \* (C4\_z\_score[i] - mean\_z\_C4)  
ss\_z\_C4 = ss\_z\_C4/(len(C4\_z\_score) - 1)  
std\_z\_C4 = ss\_z\_C4 \*\* 0.5  
print("C4成绩的平均值是：", mean\_z\_C4)  
print("C4成绩的协方差是：", ss\_z\_C4)  
print("C4成绩的标准差是：", std\_z\_C4)  
C4\_z\_scoreFinal = []  
for m in range(len(C4\_z\_score)):  
 C4\_z\_scoreFinal.append((C4\_z\_score[m] - mean\_z\_C4)/std\_z\_C4)  
print("课程4归一化的数据矩阵为：")  
for x in range(len(C4\_z\_scoreFinal)):  
 print(C4\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
  
# 计算C5的z-score  
C5\_z\_score = []  
for i in range(len(d)):  
 if d[i][9] != '':  
 C5\_z\_score.append(float(d[i][9]))  
# 求协方差  
ss\_z\_C5 = 0.0  
mean\_z\_C5 = mean(C5\_z\_score) # 成绩的平均值  
for i in range(len(C5\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C5 += (C5\_z\_score[i] - mean\_z\_C5) \* (C5\_z\_score[i] - mean\_z\_C5)  
ss\_z\_C5 = ss\_z\_C5/(len(C5\_z\_score) - 1)  
std\_z\_C5 = ss\_z\_C5 \*\* 0.5  
print("C5成绩的平均值是：", mean\_z\_C5)  
print("C5成绩的协方差是：", ss\_z\_C5)  
print("C5成绩的标准差是：", std\_z\_C5)  
C5\_z\_scoreFinal = []  
for m in range(len(C5\_z\_score)):  
 C5\_z\_scoreFinal.append((C5\_z\_score[m] - mean\_z\_C5)/std\_z\_C5)  
print("课程5归一化的数据矩阵为：")  
for x in range(len(C5\_z\_scoreFinal)):  
 print(C5\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
  
# 计算C6的z-score  
C6\_z\_score = []  
for i in range(len(d)):  
 if d[i][10] != '':  
 C6\_z\_score.append(float(d[i][10]))  
# 求协方差  
ss\_z\_C6 = 0.0  
mean\_z\_C6 = mean(C6\_z\_score) # 成绩的平均值  
for i in range(len(C6\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C6 += (C6\_z\_score[i] - mean\_z\_C6) \* (C6\_z\_score[i] - mean\_z\_C6)  
ss\_z\_C6 = ss\_z\_C6/(len(C6\_z\_score) - 1)  
std\_z\_C6 = ss\_z\_C6 \*\* 0.5  
print("C6成绩的平均值是：", mean\_z\_C6)  
print("C6成绩的协方差是：", ss\_z\_C6)  
print("C6成绩的标准差是：", std\_z\_C6)  
C6\_z\_scoreFinal = []  
for m in range(len(C6\_z\_score)):  
 C6\_z\_scoreFinal.append((C6\_z\_score[m] - mean\_z\_C6)/std\_z\_C6)  
print("课程6归一化的数据矩阵为：")  
for x in range(len(C6\_z\_scoreFinal)):  
 print(C6\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C7的z-score  
C7\_z\_score = []  
for i in range(len(d)):  
 if d[i][11] != '':  
 C7\_z\_score.append(float(d[i][11]))  
# 求协方差  
ss\_z\_C7 = 0.0  
mean\_z\_C7 = mean(C7\_z\_score) # 成绩的平均值  
for i in range(len(C7\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C7 += (C7\_z\_score[i] - mean\_z\_C7) \* (C7\_z\_score[i] - mean\_z\_C7)  
ss\_z\_C7 = ss\_z\_C7/(len(C7\_z\_score) - 1)  
std\_z\_C7 = ss\_z\_C7 \*\* 0.5  
print("C7成绩的平均值是：", mean\_z\_C7)  
print("C7成绩的协方差是：", ss\_z\_C7)  
print("C7成绩的标准差是：", std\_z\_C7)  
C7\_z\_scoreFinal = []  
for m in range(len(C7\_z\_score)):  
 C7\_z\_scoreFinal.append((C7\_z\_score[m] - mean\_z\_C7)/std\_z\_C7)  
print("课程7归一化的数据矩阵为：")  
for x in range(len(C7\_z\_scoreFinal)):  
 print(C7\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C8的z-score  
C8\_z\_score = []  
for i in range(len(d)):  
 if d[i][12] != '':  
 C8\_z\_score.append(float(d[i][12]))  
# 求协方差  
ss\_z\_C8 = 0.0  
mean\_z\_C8 = mean(C8\_z\_score) # 成绩的平均值  
for i in range(len(C8\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C8 += (C8\_z\_score[i] - mean\_z\_C8) \* (C8\_z\_score[i] - mean\_z\_C8)  
ss\_z\_C8 = ss\_z\_C8/(len(C8\_z\_score) - 1)  
std\_z\_C8 = ss\_z\_C8 \*\* 0.5  
print("C8成绩的平均值是：", mean\_z\_C8)  
print("C8成绩的协方差是：", ss\_z\_C8)  
print("C8成绩的标准差是：", std\_z\_C8)  
C8\_z\_scoreFinal = []  
for m in range(len(C8\_z\_score)):  
 C8\_z\_scoreFinal.append((C8\_z\_score[m] - mean\_z\_C8)/std\_z\_C8)  
print("课程8归一化的数据矩阵为：")  
for x in range(len(C8\_z\_scoreFinal)):  
 print(C8\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C9的z-score  
C9\_z\_score = []  
for i in range(len(d)):  
 if d[i][13] != '':  
 C9\_z\_score.append(float(d[i][13]))  
# 求协方差  
ss\_z\_C9 = 0.0  
mean\_z\_C9 = mean(C9\_z\_score) # 成绩的平均值  
for i in range(len(C9\_z\_score)): # 协方差公式，先计算和  
 ss\_z\_C9 += (C9\_z\_score[i] - mean\_z\_C9) \* (C9\_z\_score[i] - mean\_z\_C9)  
ss\_z\_C9 = ss\_z\_C9/(len(C9\_z\_score) - 1)  
std\_z\_C9 = ss\_z\_C9 \*\* 0.5  
print("C9成绩的平均值是：", mean\_z\_C9)  
print("C9成绩的协方差是：", ss\_z\_C9)  
print("C9成绩的标准差是：", std\_z\_C9)  
C9\_z\_scoreFinal = []  
for m in range(len(C9\_z\_score)):  
 C9\_z\_scoreFinal.append((C9\_z\_score[m] - mean\_z\_C9)/std\_z\_C9)  
print("课程9归一化的数据矩阵为：")  
for x in range(len(C9\_z\_scoreFinal)):  
 print(C9\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

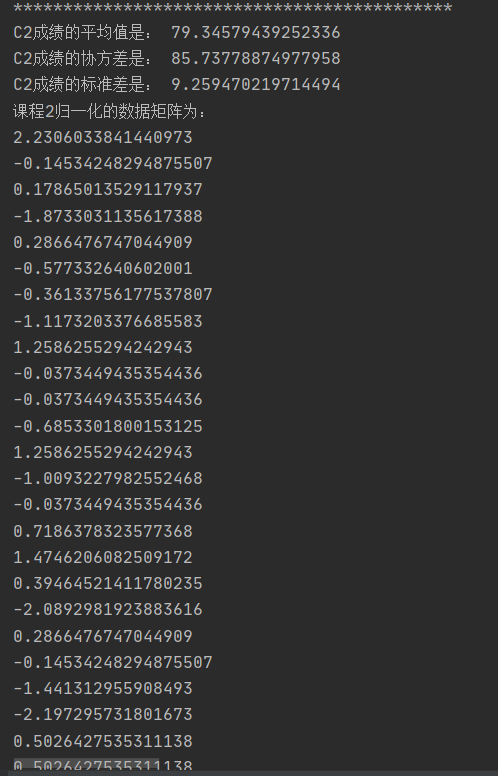
# 体能成绩的协方差，标准差已在实验一计算过，不再重复计算  
# 体能成绩的归一化矩阵  
print("体能成绩的协方差是：", ss\_Constitution)  
print("体能成绩的标准差是：", std\_Constitution)  
Constitution\_z\_scoreFinal = [] # 保存归一化矩阵  
# Constitution\_Socre保存着体能成绩（以1分，2分，3分，4分表示）  
for m in range(len(Constitution\_Socre)):  
 Constitution\_z\_scoreFinal.append((Constitution\_Socre[m] - mean\_Constitution)/std\_Constitution)  
print("体能成绩归一化的数据矩阵为：")  
for x in range(len(Constitution\_z\_scoreFinal)):  
 print(Constitution\_z\_scoreFinal[x])  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

截图：

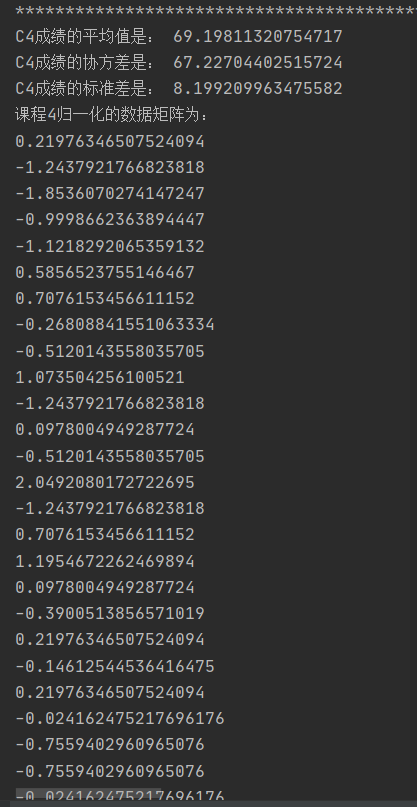
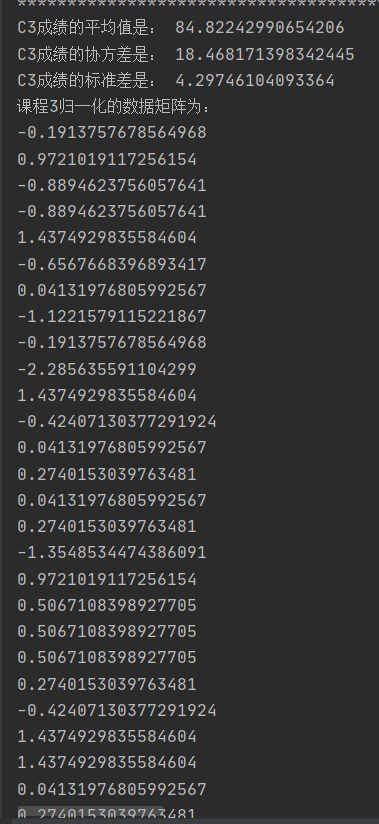
课程1的归一化数据矩阵：（部分截图）



课程2的归一化数据矩阵：



其他：



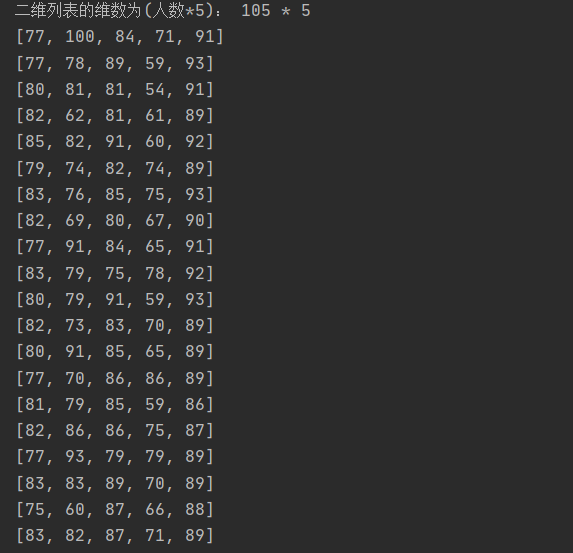
1. 计算出100x100的相关矩阵，并可视化出混淆矩阵。（为避免歧义，这里“协相关矩阵”进一步细化更正为100x100的相关矩阵，100为学生样本数目，视实际情况而定）

代码如下：

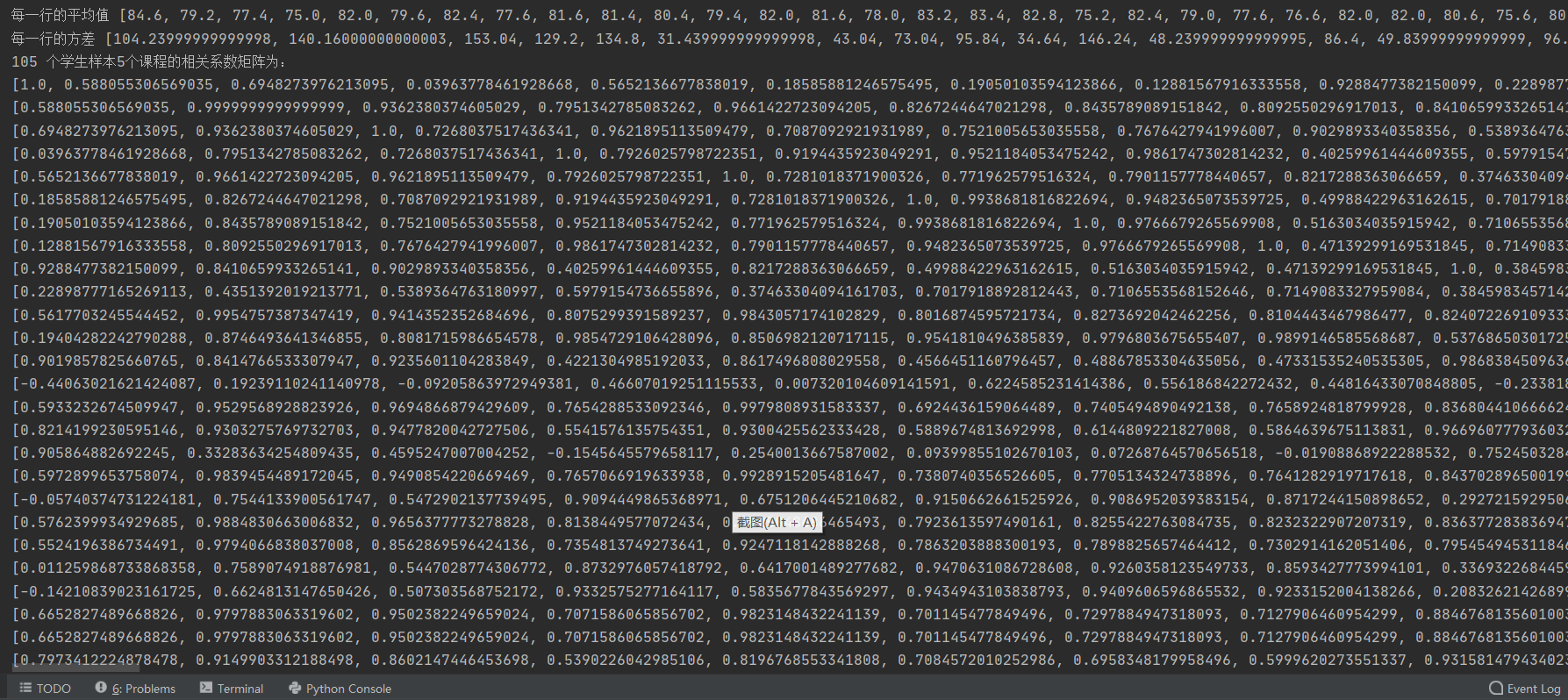
# 4.计算出100x100的相关矩阵，并可视化出混淆矩阵。（为避免歧义，这里“协相关矩阵”进一步细化更正为100x100的相关矩阵，100为学生样本数目，视实际情况而定）  
# 把学生课程1成绩至课程5成绩保存在一个新的列表,因为课程1-5是百分制的，6-9是十分制的  
row = []  
column = []  
data = []  
row\_mean = [] # 存放行的平均值  
s\_ID = [] # 存放data中学生的ID  
# 每一行的平均值  
for x in range(len(d)):  
 temp\_sum = 0.0  
 flag = 0 # 标志某一行是否有空值，1则有；0则没有  
 for y in range(5, 10): # 某一行有空值，则不加入列表data；相当于舍去有某一项成绩为空的学生  
 if d[x][y] != '':  
 row.append(int(d[x][y]))  
 temp\_sum += int(d[x][y]) # 存放第x行的和  
 if d[x][y] == '':  
 flag = 1  
 if flag == 0:  
 row\_mean.append(temp\_sum/5.0)  
 data.append(row)  
 s\_ID.append(d[x][0])  
 row = []  
  
# 计算每一行的方差  
row\_s = [] # 存放每一行的方差  
for x in range(len(data)):  
 temp\_sum = 0.0  
 for y in range(5):  
 temp\_sum += (data[x][y] - row\_mean[x]) \*\* 2 # 先求差的平方  
 row\_s.append(temp\_sum/5) # 第x行的方差  
  
print("二维列表的维数为(人数\*5)：", len(data), "\*", len(data[0]))  
for line in data: # 输出最后结果  
 print(line)  
print("每一行的平均值", row\_mean)  
print("每一行的方差", row\_s)  
  
# 求协方差  
p\_x\_y = [] # 相关系数矩阵  
row\_p = [] # 行的相关系数  
for x in range(len(data)): # data第x行和第y行的相关系数  
 for y in range(len(data)):  
 temp\_sum = 0.0  
 for m in range(5):  
 temp\_sum += (data[x][m] - row\_mean[x]) \* (data[y][m] - row\_mean[y])  
 row\_p.append(temp\_sum/(5\*((row\_s[x] \* row\_s[y]) \*\* 0.5))) # 相关系数公式  
 p\_x\_y.append(row\_p)  
 row\_p = []  
print(len(data), "个学生样本5个课程的相关系数矩阵为：")  
for line in p\_x\_y:  
 print(line)

截图：

这个矩阵保存了105个学生课程1-5的成绩



部分结果截图



1. 根据相关矩阵，找到距离每个样本最近的三个样本，得到100x3的矩阵（每一行为对应三个样本的ID）输出到txt文件中，以\t,\n间隔。

代码如下：

# 5. 根据相关矩阵，找到距离每个样本最近的三个样本，得到100x3的矩阵（每一行为对应三个样本的ID）输出到txt文件中，以\t,\n间隔。  
min\_distance = [] # 保存100x3的矩阵（每一行为对应三个样本的ID）  
  
for i in range(len(data)):  
 nums = p\_x\_y[i][:]  
 max\_num\_index\_list = list(map(nums.index, heapq.nlargest(4, nums))) # 找到每一行的4个最大值的下标,因为每一行最大值为1，也计算进去了  
 # 所以找到最大的4个值，舍去为1的值的下标  
 temp\_id = []  
 temp\_id.append(s\_ID[max\_num\_index\_list[1]]) # 把对应下标的ID放进min\_distance矩阵，但并未排序  
 temp\_id.append(s\_ID[max\_num\_index\_list[2]])  
 temp\_id.append(s\_ID[max\_num\_index\_list[3]])  
 min\_distance.append(temp\_id)  
print("5. 根据相关矩阵，找到距离每个样本最近的三个样本，得到100x3的矩阵（每一行为对应三个样本的ID）")  
for line in min\_distance:  
 print(line)  
# 输出到txt文件中，以\t,\n间隔。  
with open(r'F:\大三上\机器学习\100-3.txt', 'w') as f:  
 for i in min\_distance:  
 for j in i:  
 f.write(j)  
 f.write('\t')  
 f.write('\n')  
 f.close()

截图：

