1. 用至少两种方法分别创建一维数组、二维数组、三维数组；生成一个三维随机数组，并显示他的所有属性；生成一个随机数组，找出符合某个条件（自给）的所有元素；生成多个随机数组，实现数组的不同轴上的合并；生成一个随机数组，实现数组的不同轴上的分割；生成一个随机数组，实现数组的不同轴上的排序；生成一个1000\*4的随机数组，描述其一些统计属性。
2. import numpy as np
3. *# 1. 创建一维数组、二维数组、三维数组*
4. *# 方法1：使用 np.array*
5. array\_1d = np.array([1, 2, 3, 4, 5])
6. array\_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
7. array\_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
8. *# 方法2：使用 np.arange 和 np.reshape*
9. array\_1d\_method2 = np.arange(5)
10. array\_2d\_method2 = np.arange(9).reshape(3, 3)
11. array\_3d\_method2 = np.arange(12).reshape(2, 2, 3)
12. *# 2. 生成一个三维随机数组，并显示其所有属性*
13. random\_3d\_array = np.random.rand(2, 3, 4)
14. print("三维随机数组:\n", random\_3d\_array)
15. print("数组形状:", random\_3d\_array.shape)
16. print("数组维度:", random\_3d\_array.ndim)
17. print("数组元素总数:", random\_3d\_array.size)
18. print("数组数据类型:", random\_3d\_array.dtype)
19. *# 3. 生成一个随机数组，找出符合某个条件（自给）的所有元素*
20. random\_array = np.random.randint(1, 10, size=(5, 5))
21. print("随机数组:\n", random\_array)
22. condition\_elements = random\_array[random\_array > 5]
23. print("符合条件的元素:", condition\_elements)
24. *# 4. 生成多个随机数组，实现数组的不同轴上的合并*
25. random\_array1 = np.random.randint(1, 10, size=(3, 4))
26. random\_array2 = np.random.randint(1, 10, size=(3, 4))
27. concatenated\_array = np.concatenate((random\_array1, random\_array2), axis=0) *# 沿着第一维拼接*
28. print("合并后的数组:\n", concatenated\_array)
29. *# 5. 生成一个随机数组，实现数组的不同轴上的分割*
30. split\_array = np.random.randint(1, 10, size=(4, 4))
31. split\_result = np.split(split\_array, 2, axis=1) *# 沿着第二维分割*
32. print("分割后的数组:")
33. for i, sub\_array in enumerate(split\_result):
34. print(f"子数组{i}:\n", sub\_array)
35. *# 6. 生成一个随机数组，实现数组的不同轴上的排序*
36. sort\_array = np.random.randint(1, 100, size=(3, 4))
37. sorted\_array = np.sort(sort\_array, axis=1) *# 沿着第二维排序*
38. print("排序后的数组:\n", sorted\_array)
39. *# 7. 生成一个1000\*4的随机数组，描述其一些统计属性*
40. large\_random\_array = np.random.rand(1000, 4)
41. print("数组的统计属性：")
42. print("平均值:", np.mean(large\_random\_array, axis=0))
43. print("标准差:", np.std(large\_random\_array, axis=0))
44. print("最小值:", np.min(large\_random\_array, axis=0))
45. print("最大值:", np.max(large\_random\_array, axis=0))
46. print("中位数:", np.median(large\_random\_array, axis=0))



2. 读取mtcars.csv文件；查看数据的维度、大小、占用内存空间、类型等数据信息； 获得各属性的若干统计信息；按cyl属性进行分组，并计算每组disp、hp、drat、wt、qsec等属性的统计信息；通过柱状图、饼图等工具可视化上述统计信息。

import pandas as pd

import matplotlib.pyplot as plt

plt.rcParams["font.sans-serif"] = ["SimHei"]

plt.rcParams['axes.unicode\_minus'] = False

*# 1. 读取CSV文件*

data = pd.read\_csv('mtcars.csv')

*# 2. 查看数据的维度、大小、占用内存空间、类型等信息*

print("数据维度:", data.shape) *# 行数和列数*

print("数据大小:", data.size) *# 总元素个数*

print("数据占用内存空间:", data.memory\_usage(deep=True).sum(), "bytes") *# 内存占用*

print("数据类型:\n", data.dtypes) *# 每列的数据类型*

*# 3. 获得各属性的若干统计信息*

print("\n各属性的统计信息：")

print(data.describe()) *# 数值列的统计信息*

*# 4. 按cyl属性进行分组，并计算每组disp、hp、drat、wt、qsec等属性的统计信息*

grouped\_data = data.groupby('cyl')[['disp', 'hp', 'drat', 'wt', 'qsec']].describe()

print("\n按cyl分组后的统计信息：")

print(grouped\_data)

*# 5. 可视化统计信息：柱状图和饼图*

*# a. 各车型的平均马力（hp）柱状图*

mean\_hp = data.groupby('cyl')['hp'].mean()

mean\_hp.plot(kind='bar', title='平均马力（hp）按cyl分组', color='skyblue')

plt.ylabel('平均马力')

plt.xlabel('cyl')

plt.show()

*# b. 按cyl属性显示各类别的数量（饼图）*

cyl\_counts = data['cyl'].value\_counts()

cyl\_counts.plot(kind='pie', autopct='%1.1f%%', title='cyl分布饼图')

plt.ylabel('') *# 不显示y标签*

plt.show()

*# c. 柱状图：各车型的平均重量（wt）*

mean\_wt = data.groupby('cyl')['wt'].mean()

mean\_wt.plot(kind='bar', title='平均重量（wt）按cyl分组', color='lightgreen')

plt.ylabel('平均重量')

plt.xlabel('cyl')

plt.show()

*# d. 计算各属性的相关性，生成相关系数矩阵*

correlation\_matrix = data.corr()

print("\n相关系数矩阵：")

print(correlation\_matrix)

*# e. 可视化相关系数矩阵（热图）*

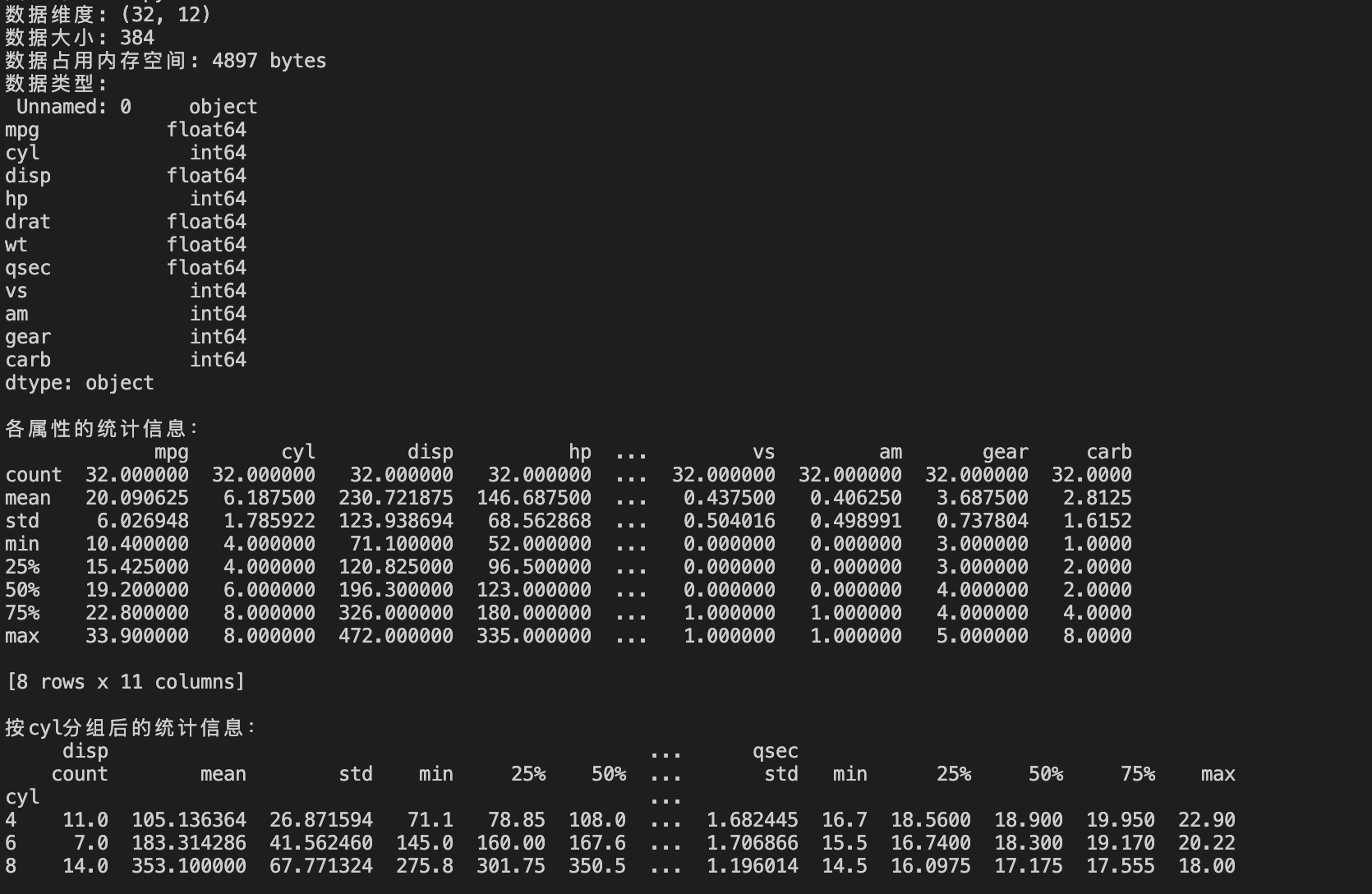
import seaborn as sns

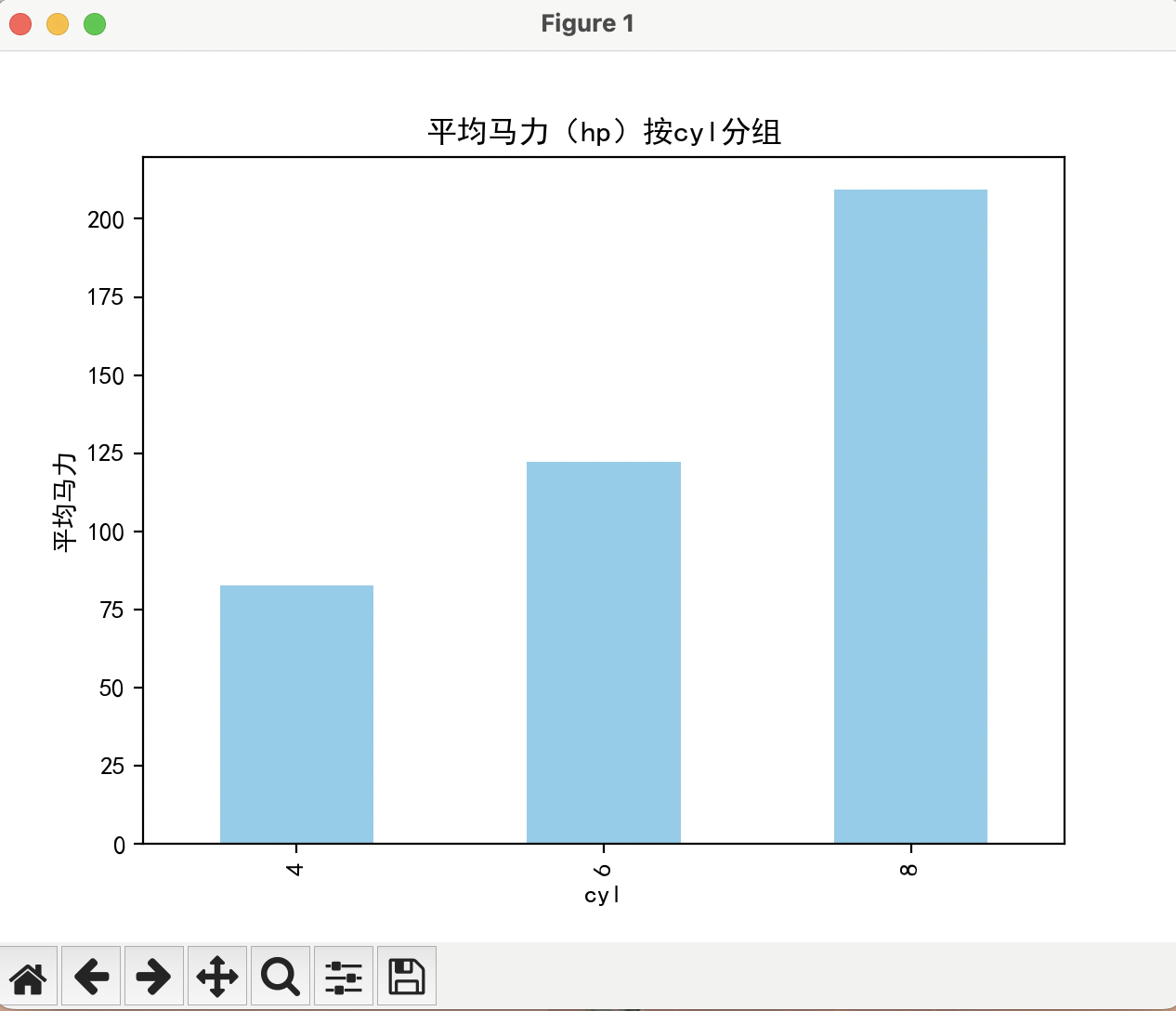
plt.figure(figsize=(10, 8))

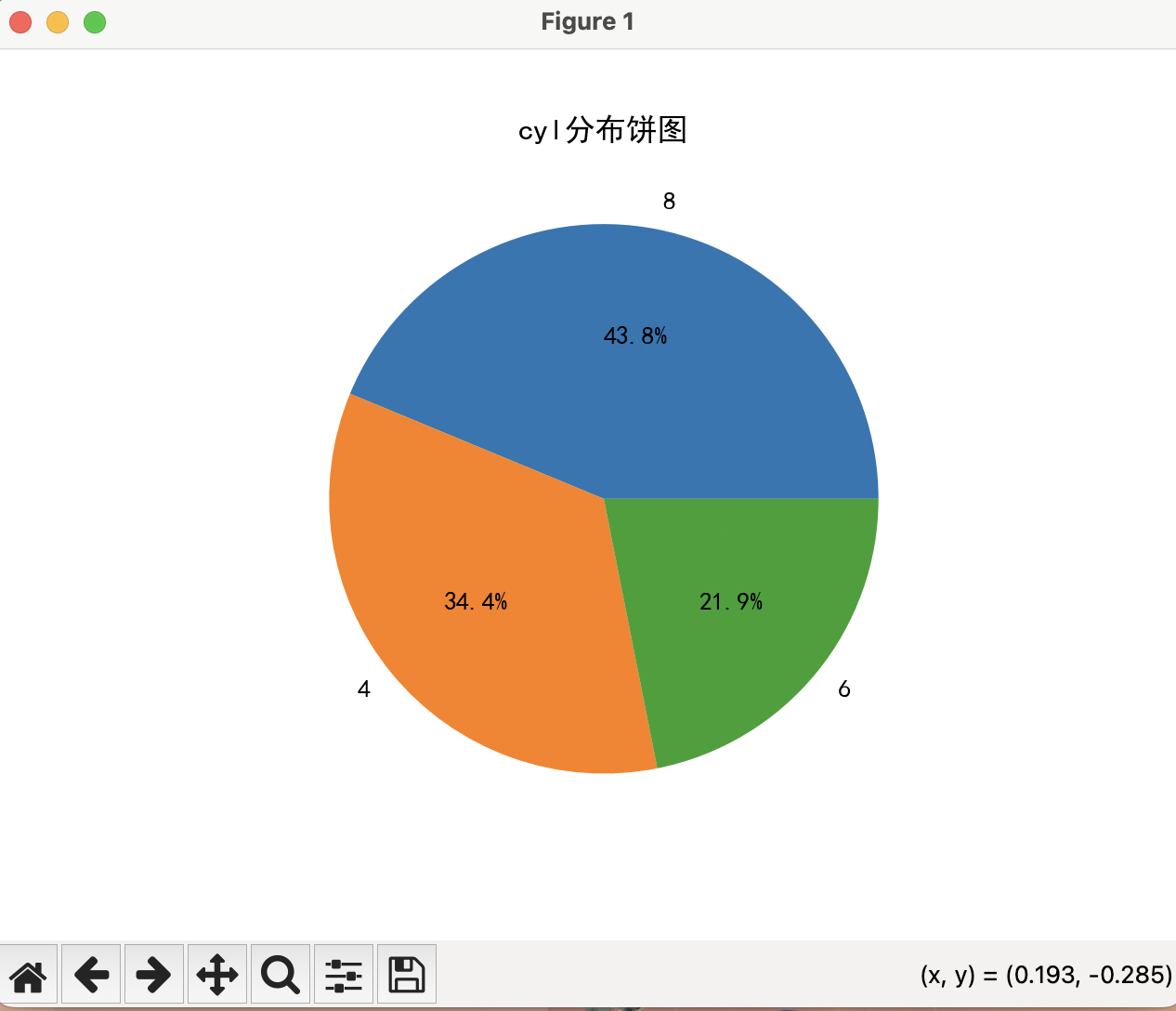
sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

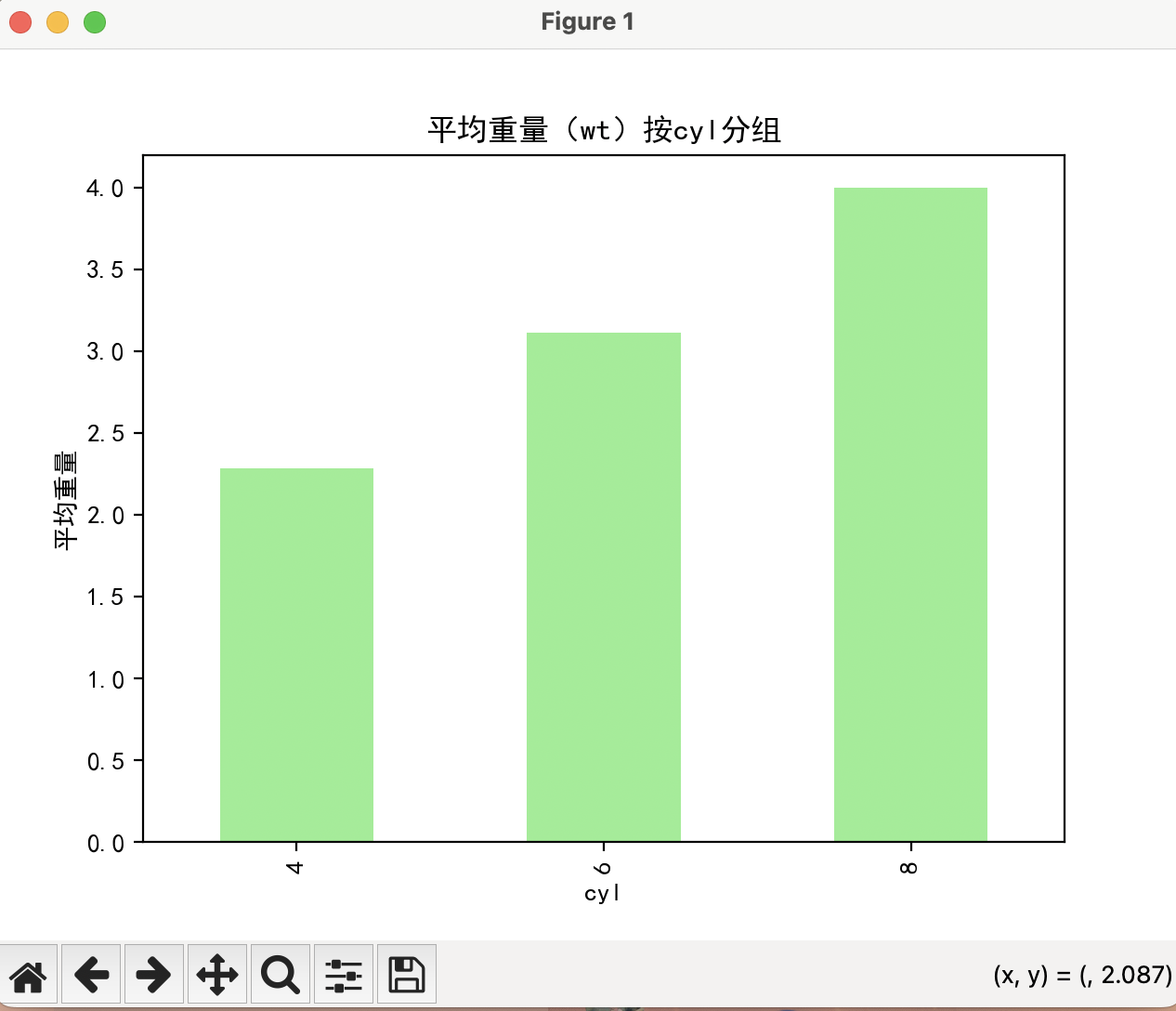
plt.title("属性相关系数矩阵")

plt.show()









3. 读取missing\_data.csv文件；填充其中缺失值，并有理有据说明选择所采用的填充方法的理由。

import pandas as pd

*# 读取数据*

data = pd.read\_csv('missing\_data.csv', header=None)

*# 查看原数据*

print("原数据：")

print(data)

*# 填充缺失值*

*# 1. 对于数值列，我们使用均值填充缺失值*

data\_filled\_mean = data.fillna(data.mean())

*# 2. 如果我们认为数据可能有极端值，可以选择使用中位数填充*

*# data\_filled\_median = data.fillna(data.median())*

*# 3. 如果数据的缺失是随机且前后数据相关，可以使用前向填充（ffill）或后向填充（bfill）*

*# data\_filled\_ffill = data.fillna(method='ffill')*

*# data\_filled\_bfill = data.fillna(method='bfill')*

*# 4. 如果有更复杂的插值需求，可以使用插值法*

*# data\_filled\_interpolate = data.interpolate(method='linear')*

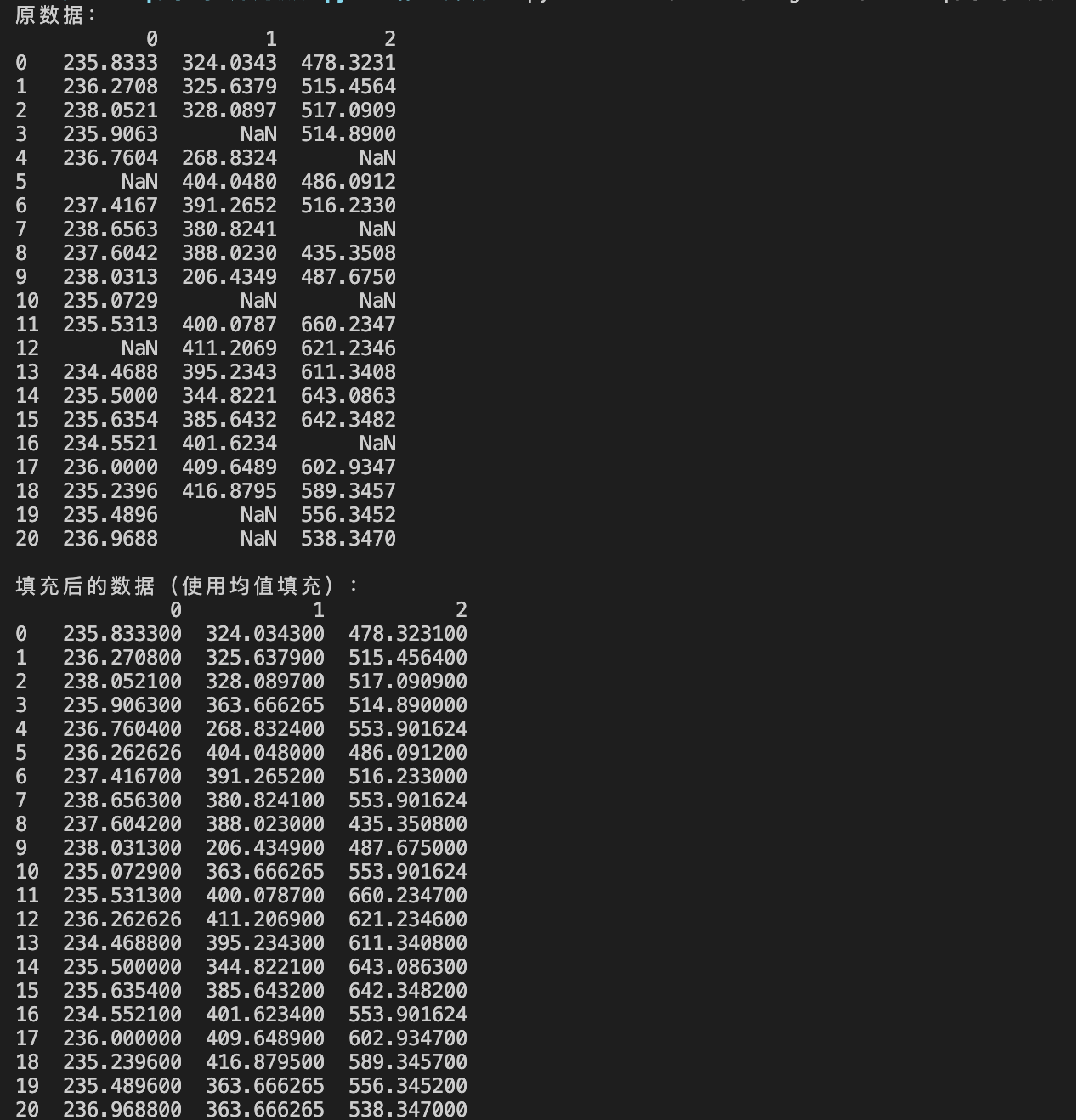
*# 输出填充后的数据*

print("\n填充后的数据（使用均值填充）：")

print(data\_filled\_mean)

*# 保存填充后的数据（如果需要）*

*# data\_filled\_mean.to\_csv('filled\_data.csv', index=False)*



因为数据是数值型的，并且看上去是测量数据，使用均值填充是一种合理的选择。这样填充后的数据保持了整体的均衡性，同时不至于造成信息丢失。