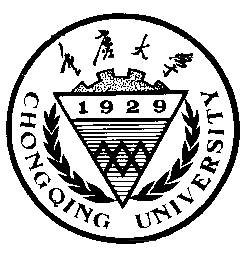
数据科学导论实验报告

实验一 python数据基础实践



学 生：严奕凡\_20220815

年 级：2022级

专 业：数据科学与大数据技术

重庆大学大数据与软件学院

2024年 5月 7日

## 实验目的：

为后续上机实验做准备，掌握python 基础操作：包括熟悉python 程序运行环

境及基本语法，理解python 基本数据类型，掌握python 的算术运算规则及表达

式的书写方法；掌握常用的python 基本数据函数分析库：pandas、numpy.

## 实验要求

1. Python基础函数分析库
   1. 熟悉常用的Scipy科学计算操作
   2. 熟悉常用的matplotlib绘图操作
2. Python基础数据分析

2.1、探索数据描述

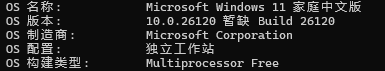
2.2、数据清理

2.3、数据变换操作

2.4、数据相关性分析

2.5、扩展实验

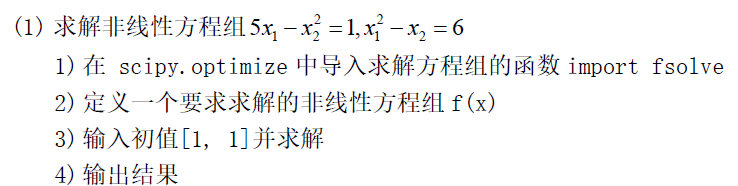
## 开发环境：

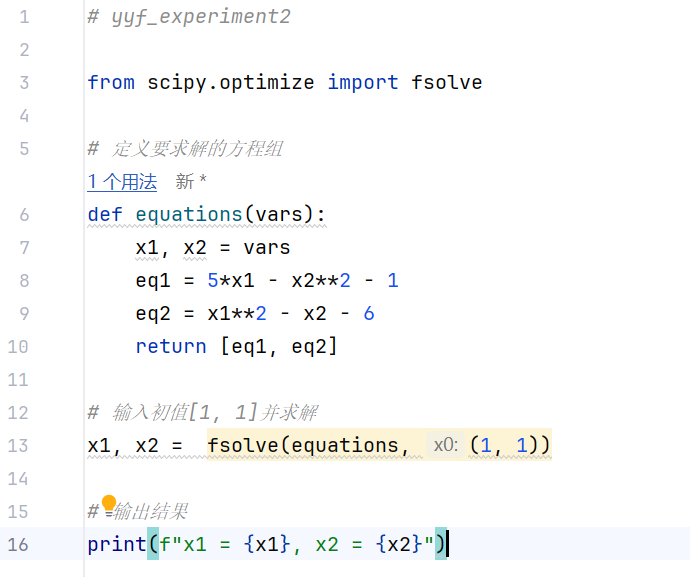


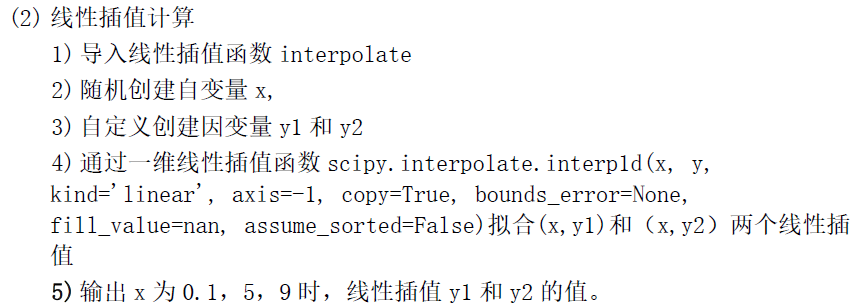
## 实验内容：

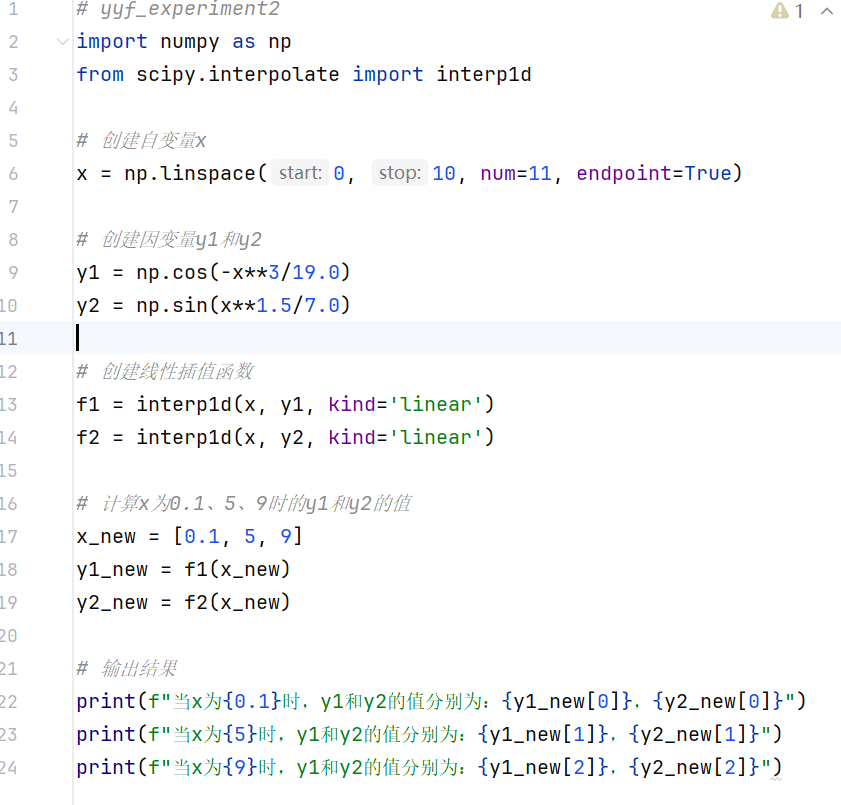
Python 基本函数分析库

（一）熟悉常用的Scipy 科学计算操作

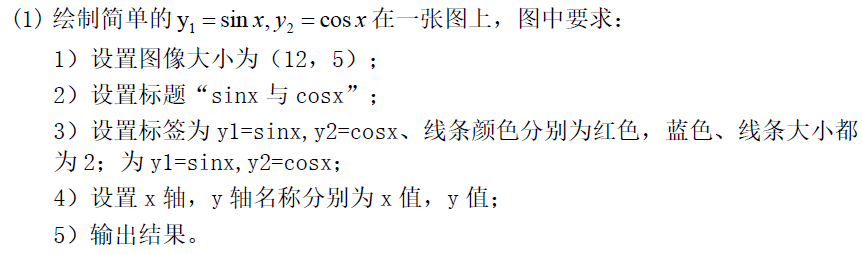


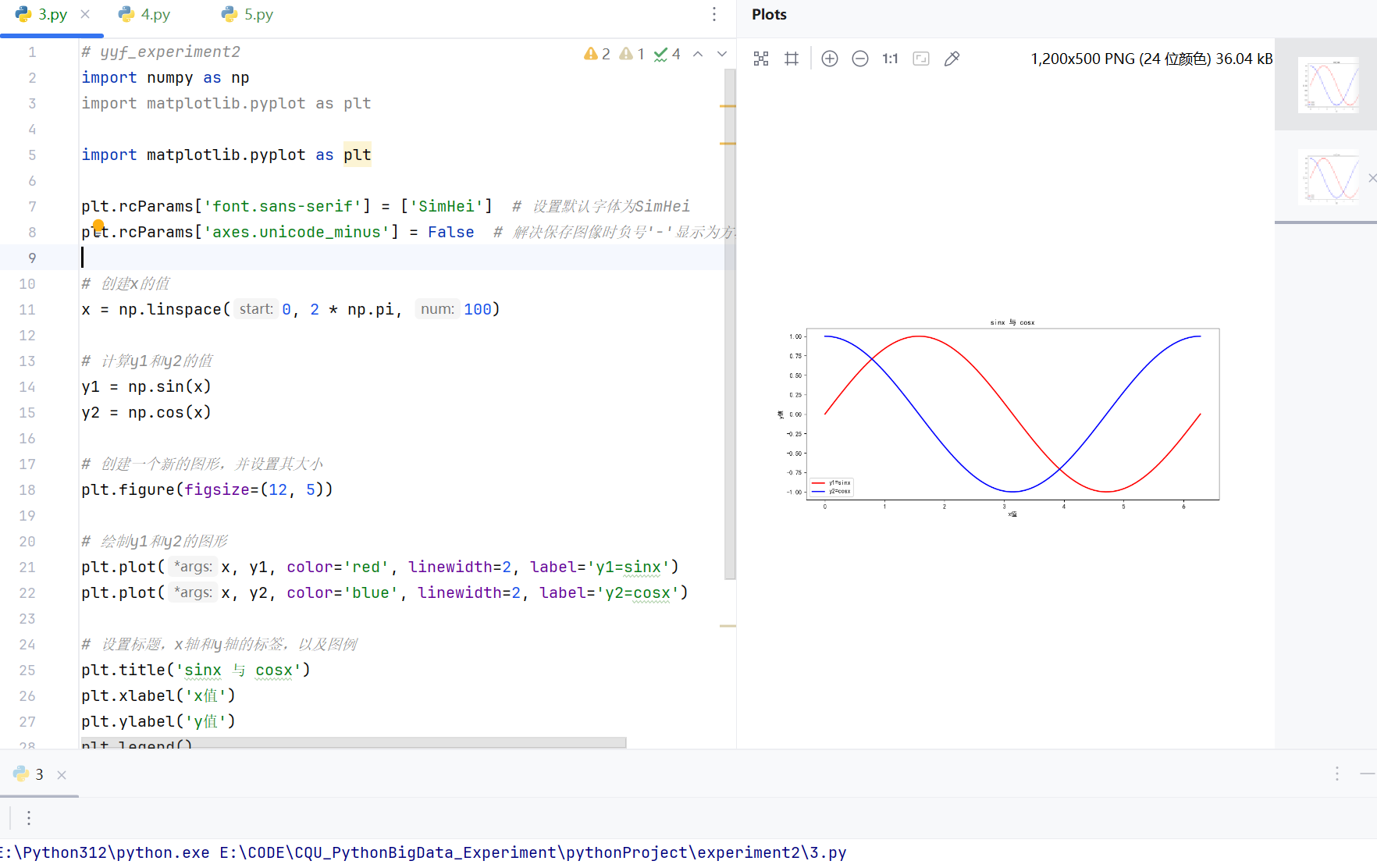






（二）熟悉常用的matplotlib 绘图操作





Python 基础数据分析

1. 探索数据描述

（在执行以下每个操作之后，使用head（）、tail（）来显示数据。）

(1) 从kaggle 下载fifa19 完整球员数据集(https://www.kaggle.com/karangadiya/f

ifa19)在数据.csv 文件包含许多列，包括每个玩家的行号、ID、name、ag

e....。读取fifa19 数据集。

(2) 将数据读入Pandas 数据框。

(3) 筛选年龄在25 岁以下的年轻球员。

(4) 根据Jumping 分数对数据进行排序。

(5) 使用describe（）方法显示列Volleys 和Dribbling 的count、mean、std、

min 和分位数数据。



1. 数据清理

（在执行以下每个操作之后，使用head（）、tail（）来显示数据。）

(1) 从数据中通过drop 删除以下三个属性Photo, Flag, Club Logo，显示结果；

(2) 用isnull（）找出所有缺少值的条目；

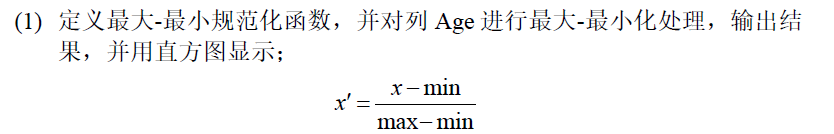
(3) 用0 填补空值；

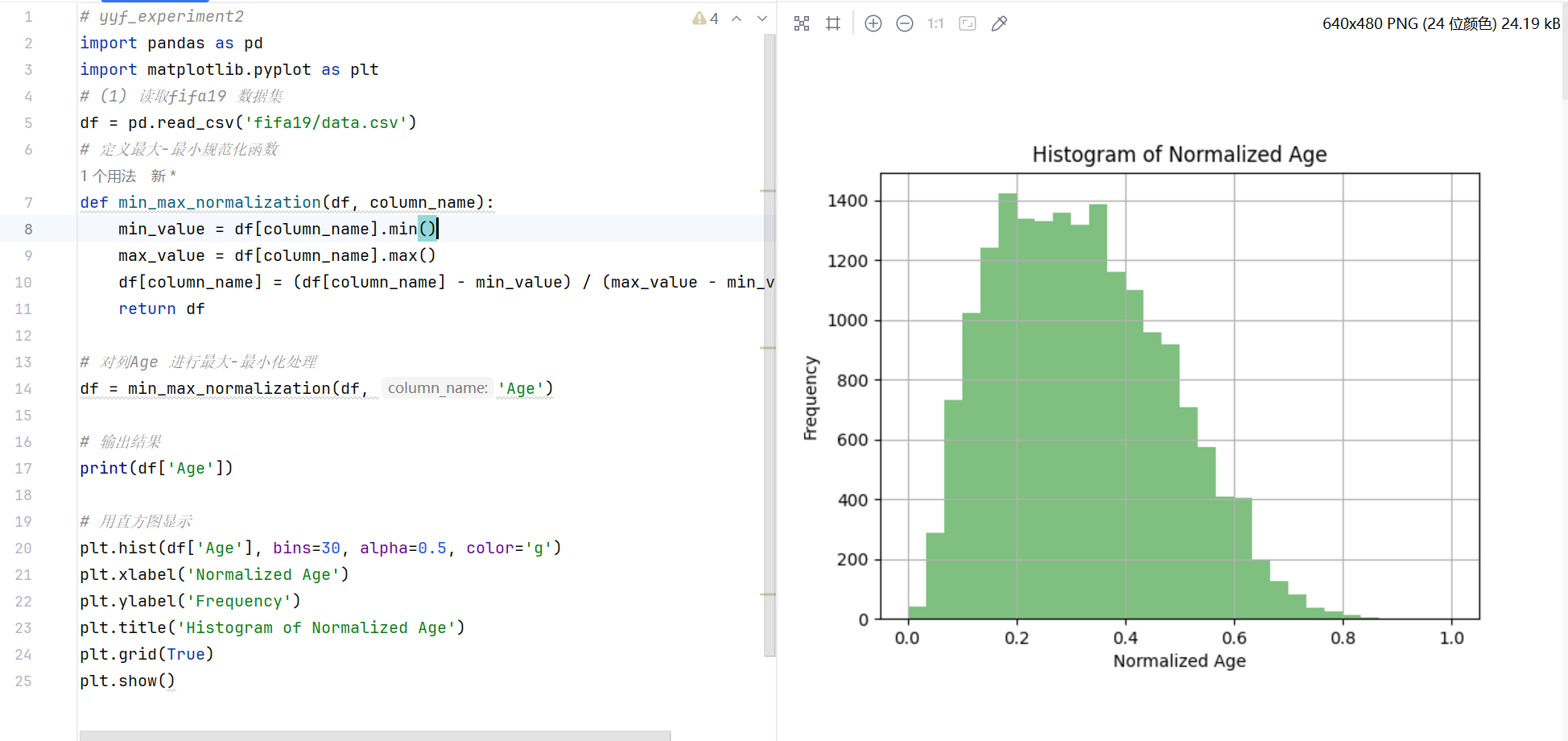
(4) 将“Real Face”属性的特征值‘Yes’替换为1，‘No’变为0；“Preferred

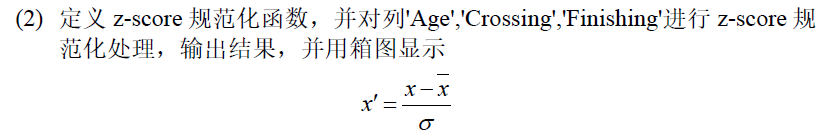
Foot”属性的特征值‘Right’变为1，‘Left’变为0, 并输出结果。

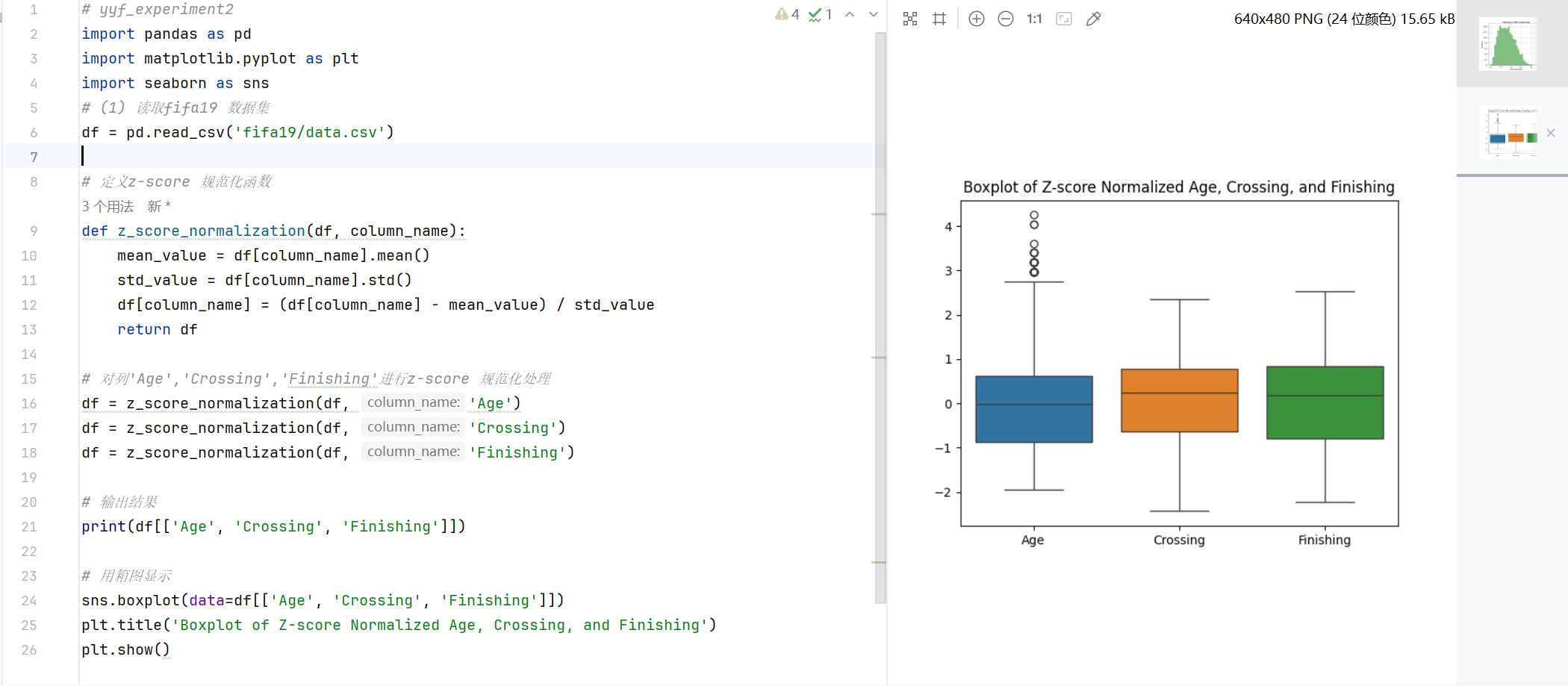


1. 数据变换操作

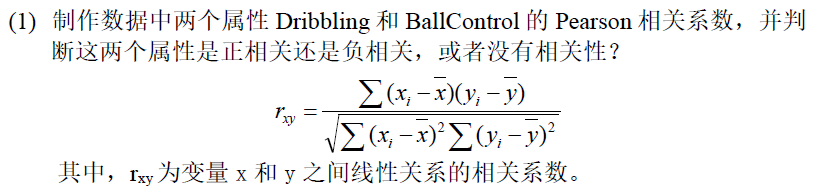


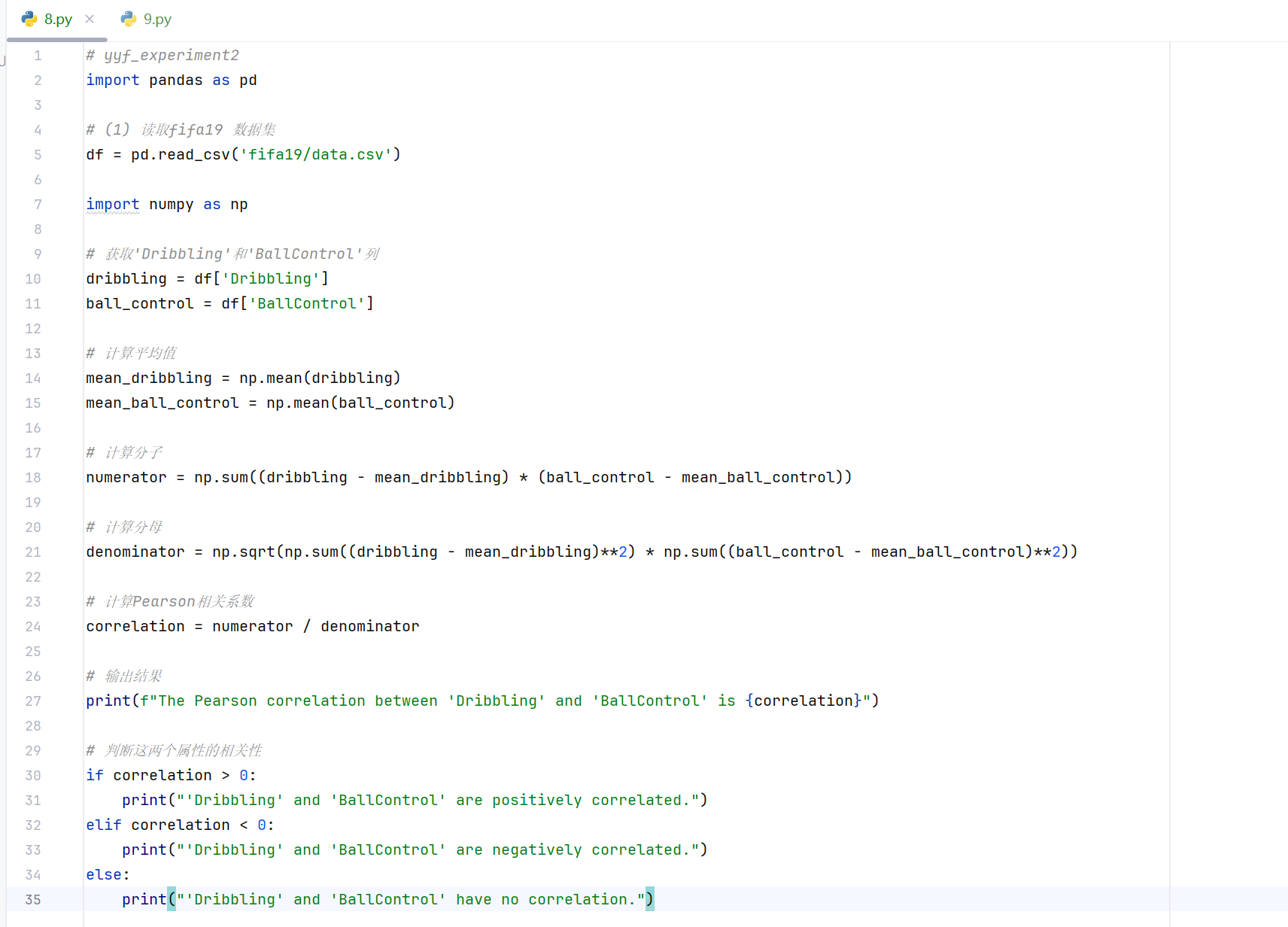






1. 数据相关性分析





(2) 使用pandas 中包含内置的求解pearson 系数方法函数,求两个属性Dribblin

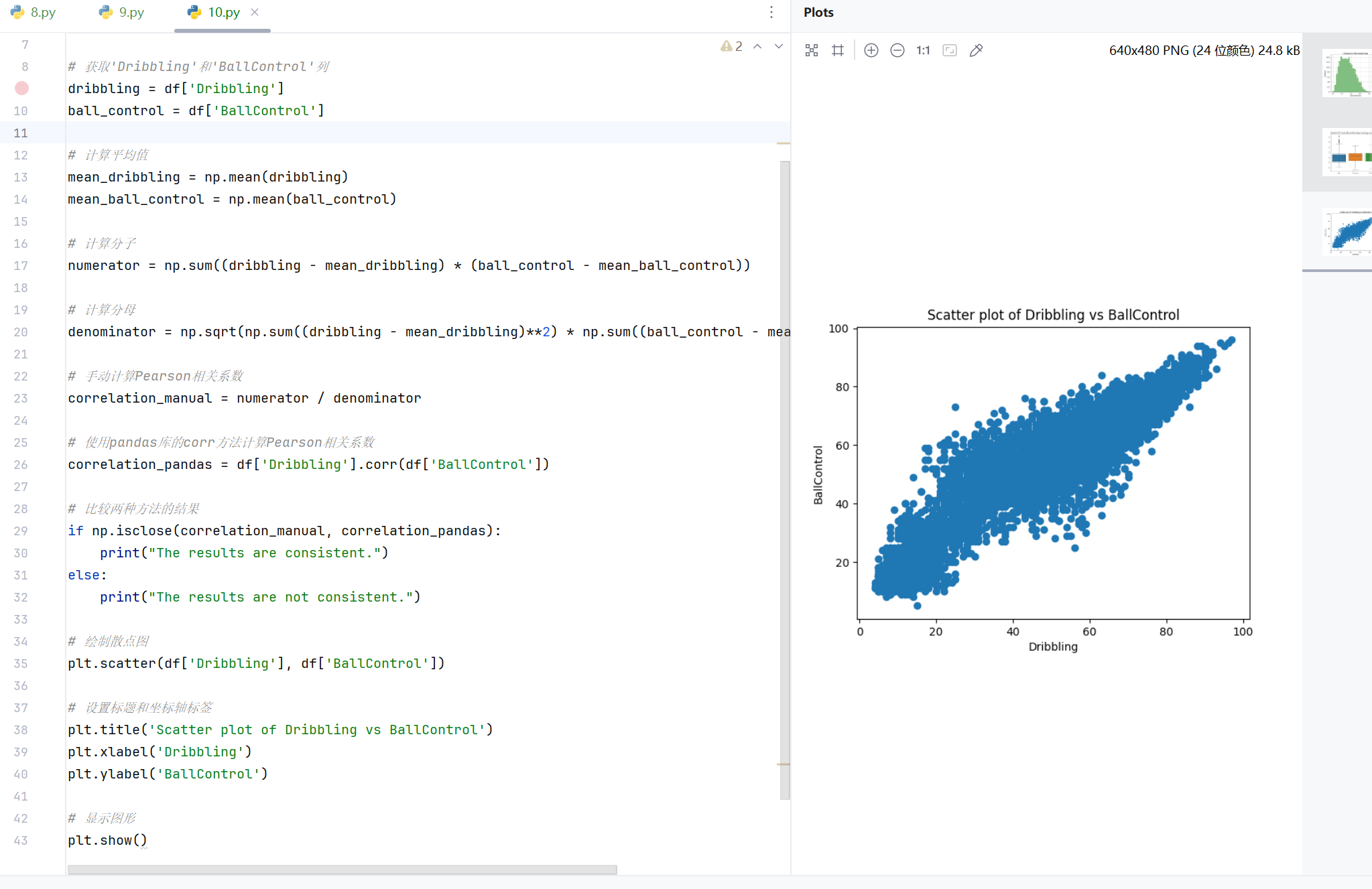
g 和BallControl 的Pearson 相关系数，判断这两个属性是正相关还是负相

关，或者没有相关性？



(3) 判断前两种方法算出结果是否一致？并用散点图绘制上述两个属性的相关

性分析图



1. 扩展实验

(4) 假设你负责一个数据分析项目，研究FIFA 19 的数据集，你认为你可以从

给定的数据集得出什么样的结论。写出至少3 个问题，找出可以用来回答

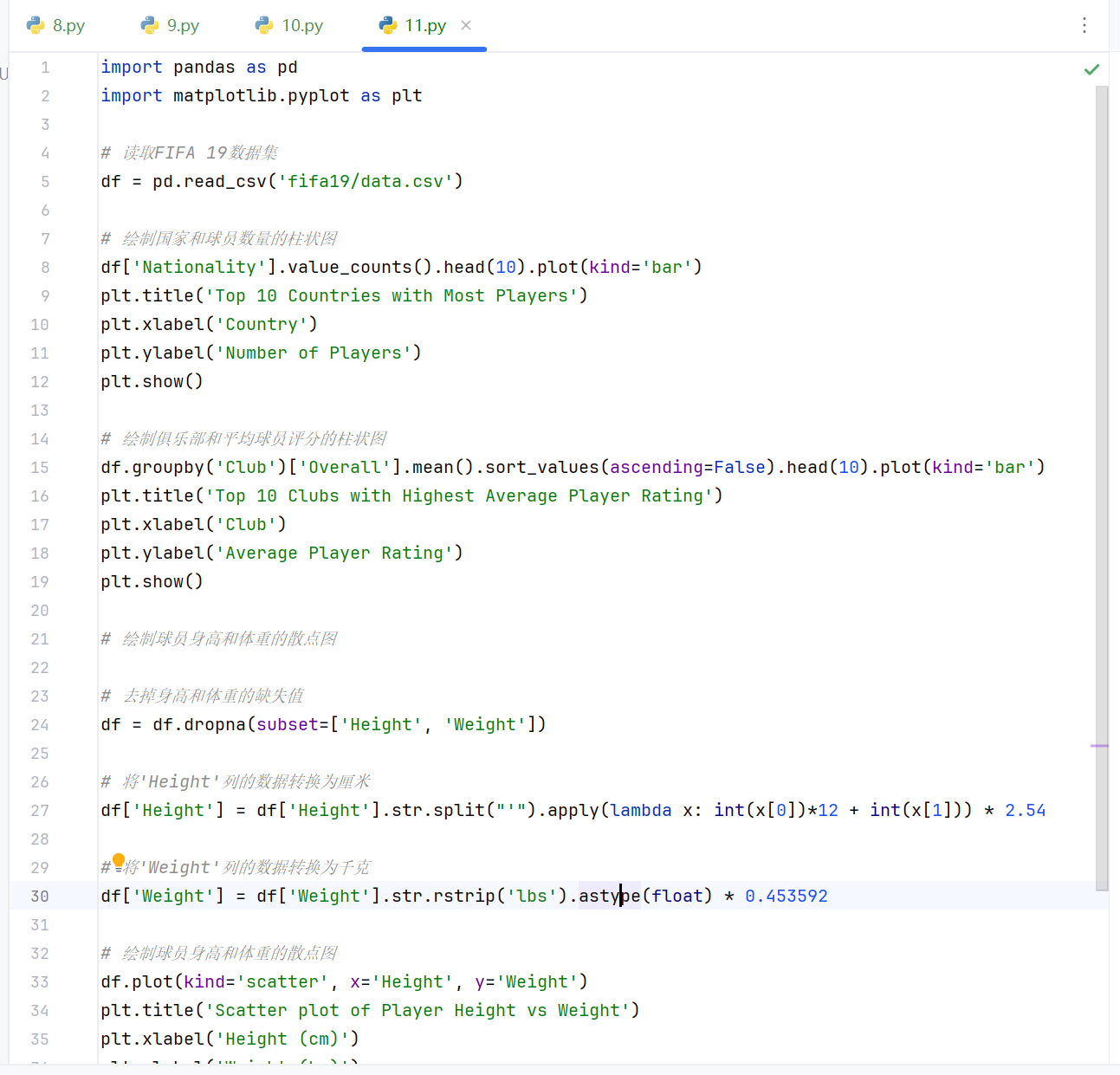
你问题的数据。例如，一个问题是每个年龄组有多少球员（假设你每5 年

指定一个年龄组，比如16-20，21-25，26-30…？读取FIFA 19 数据集并

自主绘制该数据集至少三个不同的图表？等）

(5) 参考绘制图官网：<https://matplotlib.org/stable/index.html>

探索问题：  
哪个国家球员最多？  
哪个俱乐部平均球员评分最高？  
球员的身高和体重有无相关性？



## 实验结果分析(重点)

1、熟悉常用的Scipy 科学计算操作

****

这段Python代码使用了scipy.optimize库中的fsolve函数来求解一个非线性方程组。这个方程组由两个方程组成：5\*x1 - x2\*\*2 = 1和x1\*\*2 - x2 = 6。

首先，代码定义了一个函数equations，这个函数接收一个包含两个变量x1和x2的列表作为参数，然后返回一个包含两个方程的列表。

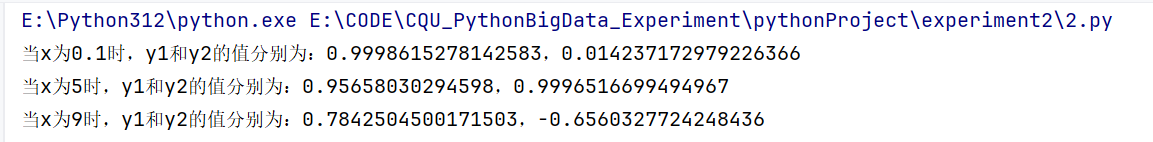
然后，代码调用fsolve函数来求解这个方程组。fsolve函数的第一个参数是要求解的方程组，第二个参数是求解的初始值。在这个例子中，初始值被设置为(1, 1)。

fsolve函数会返回一个包含求解结果的列表。在这个例子中，求解结果被赋值给了变量x1和x2。

最后，代码打印出求解结果。

这段代码的目的是求解给定的非线性方程组，并打印出求解结果。

2、线性插值计算

****

这段Python代码使用了scipy.interpolate库中的interp1d函数来进行一维线性插值。

首先，代码创建了一个从0到10的等差数列作为自变量x，然后计算了-x\*\*3/19.0的余弦值和x\*\*1.5/7.0的正弦值作为因变量y1和y2。

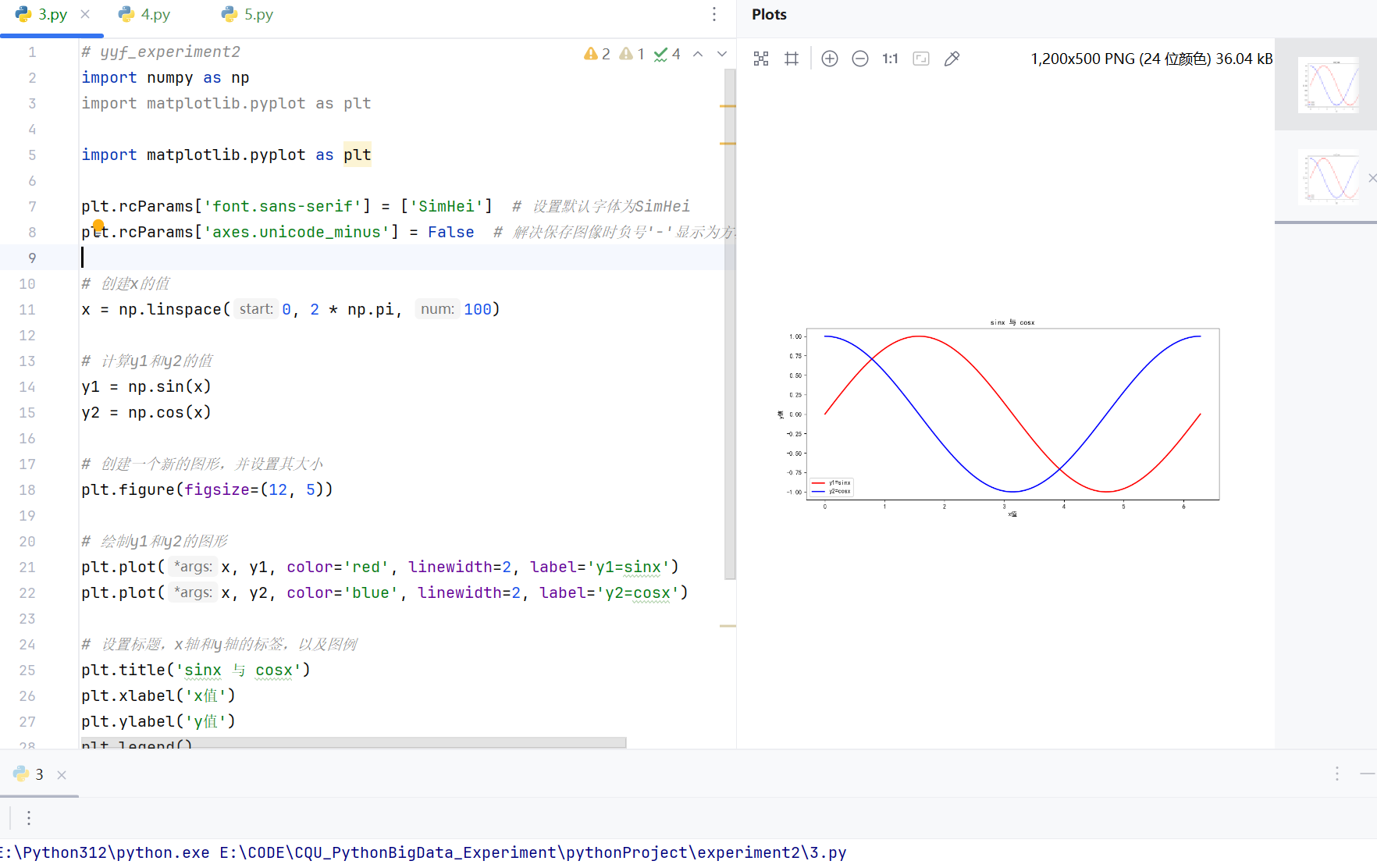
然后，代码使用interp1d函数创建了两个线性插值函数f1和f2。这两个函数分别对应于y1和y2关于x的线性插值。

接着，代码定义了一个新的x值列表x\_new，包含了0.1、5和9这三个值，并使用f1和f2计算了这三个x值对应的y1和y2的值。

最后，代码打印出了当x为0.1、5、9时，y1和y2的值。

这段代码的目的是对给定的数据点进行线性插值，并打印出特定x值对应的插值结果。

3、绘制正余弦函数

****

这段Python代码使用了numpy和matplotlib.pyplot库来绘制sinx和cosx的图形。

首先，代码设置了matplotlib的默认字体为'SimHei'，并解决了保存图像时负号'-'显示为方块的问题。

然后，代码创建了一个从0到2π的等差数列作为x的值，然后计算了x的正弦和余弦值作为y1和y2的值。

接着，代码创建了一个新的图形，并设置了其大小为(12, 5)。然后，代码绘制了y1和y2的图形，并设置了线条的颜色和大小，以及图例的标签。

最后，代码设置了图形的标题，x轴和y轴的标签，然后显示了图形。

这段代码的目的是绘制sinx和cosx的图形，并设置了图形的各种属性，如大小、标题、标签、线条颜色和大小等。

1. 探索数据描述

**Unnamed: 0 ID ... GKReflexes Release Clause**

**0 0 158023 ... 8.0 €226.5M**

**1 1 20801 ... 11.0 €127.1M**

**2 2 190871 ... 11.0 €228.1M**

**3 3 193080 ... 94.0 €138.6M**

**4 4 192985 ... 13.0 €196.4M**

**[5 rows x 89 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**18202 18202 238813 ... 9.0 €143K**

**18203 18203 243165 ... 12.0 €113K**

**18204 18204 241638 ... 13.0 €165K**

**18205 18205 246268 ... 9.0 €143K**

**18206 18206 246269 ... 9.0 €165K**

**[5 rows x 89 columns]**

**Unnamed: 0 ID Name ... GKPositioning GKReflexes Release Clause**

**15 15 211110 P. Dybala ... 5.0 8.0 €153.5M**

**16 16 202126 H. Kane ... 14.0 11.0 €160.7M**

**25 25 231747 K. Mbappé ... 11.0 6.0 €166.1M**

**42 42 205600 S. Umtiti ... 12.0 15.0 €121.1M**

**55 55 222492 L. Sané ... 9.0 14.0 €125.1M**

**[5 rows x 89 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**18202 18202 238813 ... 9.0 €143K**

**18203 18203 243165 ... 12.0 €113K**

**18204 18204 241638 ... 13.0 €165K**

**18205 18205 246268 ... 9.0 €143K**

**18206 18206 246269 ... 9.0 €165K**

**[5 rows x 89 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**17909 17909 244777 ... 58.0 €62K**

**16944 16944 242382 ... 60.0 €435K**

**17249 17249 241882 ... 60.0 €131K**

**13520 13520 239746 ... 66.0 €495K**

**17603 17603 240489 ... 52.0 €87K**

**[5 rows x 89 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**13272 13272 240701 ... NaN NaN**

**13275 13275 244541 ... NaN NaN**

**13278 13278 246078 ... NaN NaN**

**13279 13279 239679 ... NaN NaN**

**13280 13280 244543 ... NaN NaN**

**[5 rows x 89 columns]**

**Volleys Dribbling**

**count 18159.000000 18159.000000**

**mean 42.909026 55.371001**

**std 17.694408 18.910371**

**min 4.000000 4.000000**

**25% 30.000000 49.000000**

**50% 44.000000 61.000000**

**75% 57.000000 68.000000**

**max 90.000000 97.000000**

这段代码主要用于处理和分析FIFA 19的球员数据集。以下是每一步的详细解释：

导入pandas库：pandas是一个强大的数据处理库，可以用于数据清洗和分析。

读取数据集：使用pandas的read\_csv函数读取CSV文件中的数据，并将其存储在DataFrame对象df中。

显示数据的头部和尾部：使用head()和tail()函数分别显示数据集的前5行和后5行。

筛选年轻球员：通过在df中应用布尔索引，筛选出年龄小于25岁的球员，并将结果存储在young\_players中。

显示年轻球员的头部和尾部：使用head()和tail()函数分别显示年轻球员数据的前5行和后5行。

根据Jumping分数排序：使用sort\_values函数，按照'Jumping'列的值对年轻球员数据进行排序，并将结果存储在sorted\_players中。

显示排序后的球员数据的头部和尾部：使用head()和tail()函数分别显示排序后的数据的前5行和后5行。

描述统计：使用describe()函数对'Volleys'和'Dribbling'两列进行描述性统计，包括计数、均值、标准差、最小值、四分位数和最大值，并将结果存储在volleys\_dribbling中。

打印描述统计结果：打印volleys\_dribbling，显示'Volleys'和'Dribbling'两列的描述性统计结果。

5、数据清理这段代码首先删除了不需要的列，然后找出并填充了所有的缺失值，最后将'Real Face'和'Preferred Foot'两个属性的特征值进行了转换。

**Unnamed: 0 ID ... GKReflexes Release Clause**

**0 0 158023 ... 8.0 €226.5M**

**1 1 20801 ... 11.0 €127.1M**

**2 2 190871 ... 11.0 €228.1M**

**3 3 193080 ... 94.0 €138.6M**

**4 4 192985 ... 13.0 €196.4M**

**[5 rows x 86 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**18202 18202 238813 ... 9.0 €143K**

**18203 18203 243165 ... 12.0 €113K**

**18204 18204 241638 ... 13.0 €165K**

**18205 18205 246268 ... 9.0 €143K**

**18206 18206 246269 ... 9.0 €165K**

**[5 rows x 86 columns]**

**Unnamed: 0 0**

**ID 0**

**Name 0**

**Age 0**

**Nationality 0**

**...**

**GKHandling 48**

**GKKicking 48**

**GKPositioning 48**

**GKReflexes 48**

**Release Clause 1564**

**Length: 86, dtype: int64**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**0 0 158023 ... 8.0 €226.5M**

**1 1 20801 ... 11.0 €127.1M**

**2 2 190871 ... 11.0 €228.1M**

**3 3 193080 ... 94.0 €138.6M**

**4 4 192985 ... 13.0 €196.4M**

**[5 rows x 86 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**18202 18202 238813 ... 9.0 €143K**

**18203 18203 243165 ... 12.0 €113K**

**18204 18204 241638 ... 13.0 €165K**

**18205 18205 246268 ... 9.0 €143K**

**18206 18206 246269 ... 9.0 €165K**

**[5 rows x 86 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**0 0 158023 ... 8.0 €226.5M**

**1 1 20801 ... 11.0 €127.1M**

**2 2 190871 ... 11.0 €228.1M**

**3 3 193080 ... 94.0 €138.6M**

**4 4 192985 ... 13.0 €196.4M**

**[5 rows x 86 columns]**

**Unnamed: 0 ID ... GKReflexes Release Clause**

**18202 18202 238813 ... 9.0 €143K**

**18203 18203 243165 ... 12.0 €113K**

**18204 18204 241638 ... 13.0 €165K**

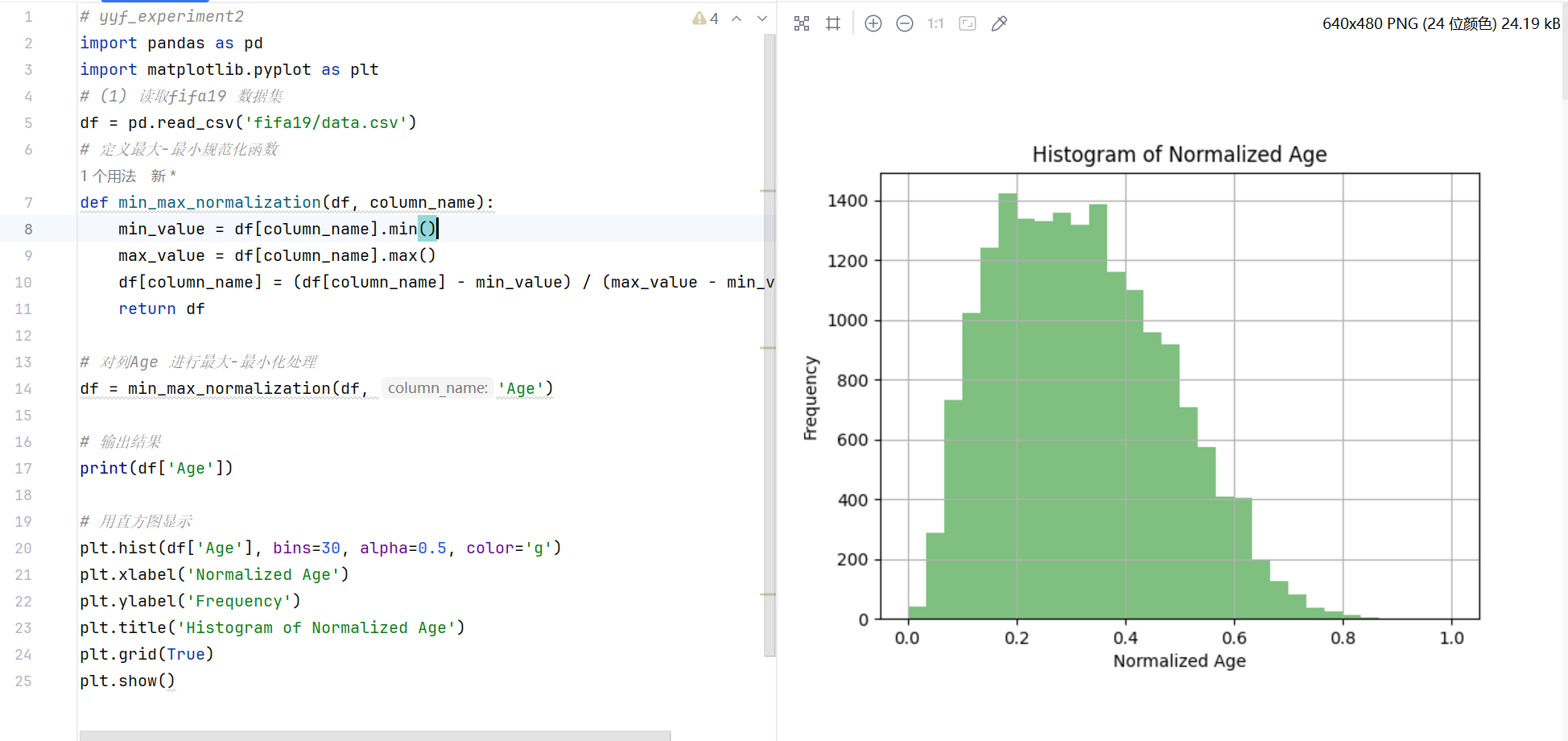
**18205 18205 246268 ... 9.0 €143K**

**18206 18206 246269 ... 9.0 €165K**

**[5 rows x 86 columns]**

1. 数据变换操作

（1）



这段代码首先读取了一个名为"fifa19/data.csv"的数据集，并将其存储在名为df的Pandas DataFrame中。

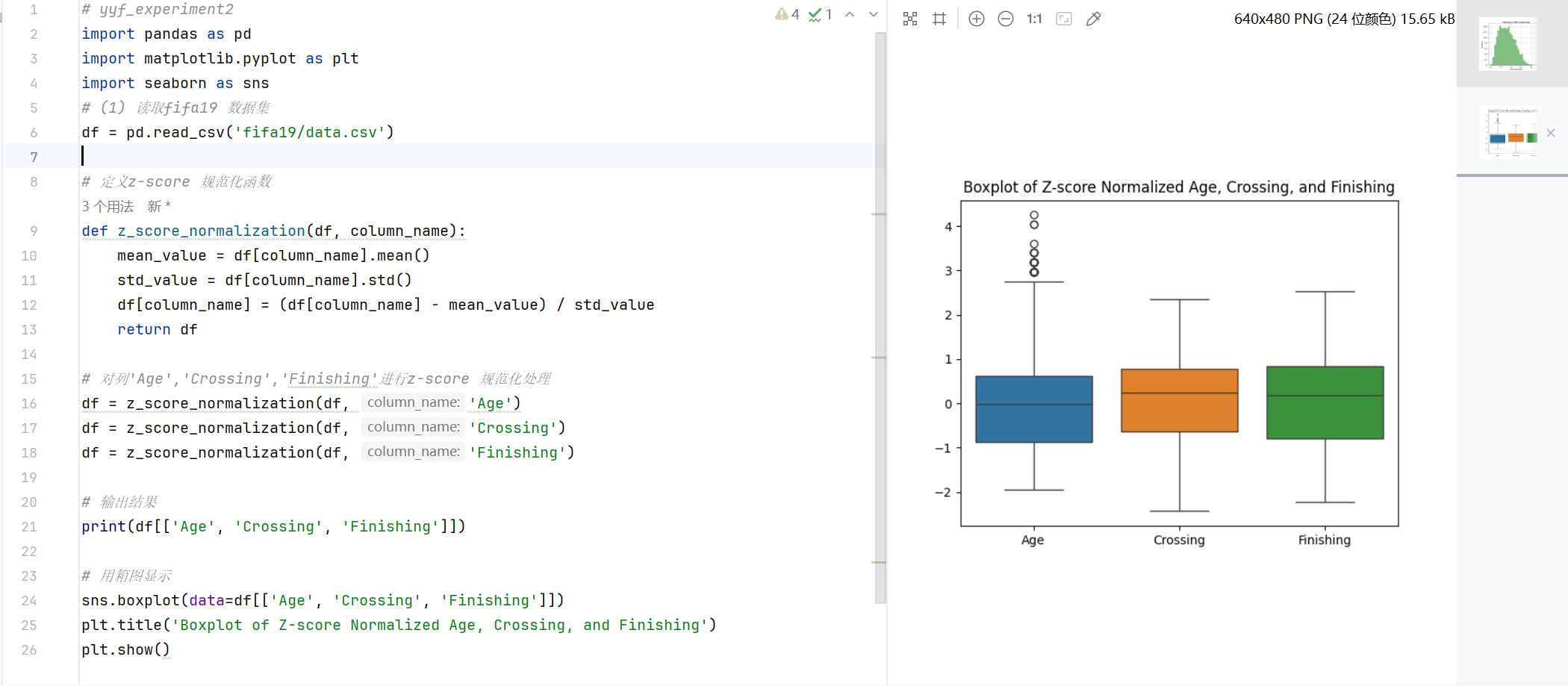
然后，它定义了一个名为min\_max\_normalization的函数，该函数接受一个DataFrame和一个列名作为输入，然后将该列的每个值减去该列的最小值，然后除以该列的最大值和最小值的差。这是一种常见的数据规范化技术，称为最大-最小规范化，它可以将数据转换到0和1之间。

接下来，这段代码对'Age'列进行了最大-最小规范化处理。

然后，它打印出了规范化后的'Age'列的值。

最后，它使用matplotlib库的hist函数绘制了规范化后的'Age'列的直方图。直方图是一种用于显示数据分布的图形，它将数据分成几个等宽的区间，然后显示每个区间的数据点数量。这可以帮助我们理解数据的分布。

**（2）**



这段代码首先读取了一个名为"fifa19/data.csv"的数据集，并将其存储在名为df的Pandas DataFrame中。

然后，它定义了一个名为z\_score\_normalization的函数，该函数接受一个DataFrame和一个列名作为输入，然后将该列的每个值减去该列的平均值，然后除以该列的标准差。这是一种常见的数据规范化技术，称为Z-score规范化或标准化，它可以将数据转换为均值为0，标准差为1的分布。

接下来，这段代码对'Age'，'Crossing'和'Finishing'这三列进行了Z-score规范化处理。

然后，它打印出了这三列规范化后的值。

最后，它使用seaborn库的boxplot函数绘制了这三列的箱线图。箱线图是一种用于显示数据分布的图形，包括最小值、第一四分位数（Q1）、中位数、第三四分位数（Q3）和最大值。这可以帮助我们理解数据的分布和离群值。

7、数据相关性分析

（1）The Pearson correlation between 'Dribbling' and 'BallControl' is 0.9389422570105279

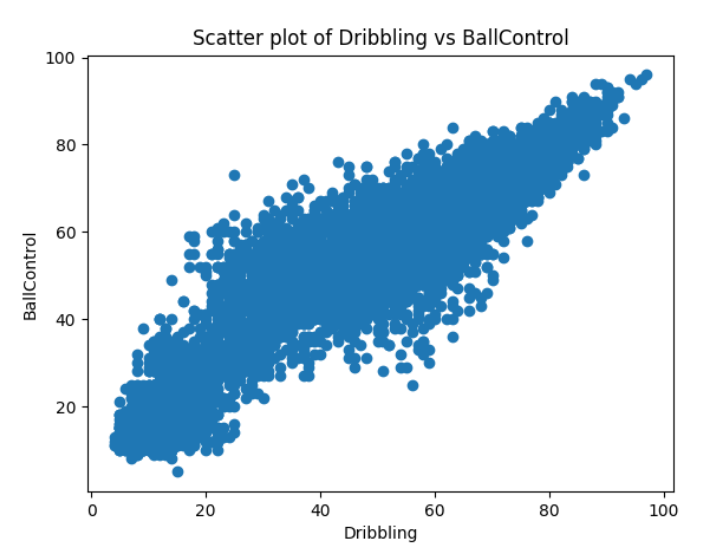
'Dribbling' and 'BallControl' are positively correlated.

这是手动给出rxy的情况 相关系数接近1 高度相关

（2）The Pearson correlation between 'Dribbling' and 'BallControl' is 0.9389422570105282

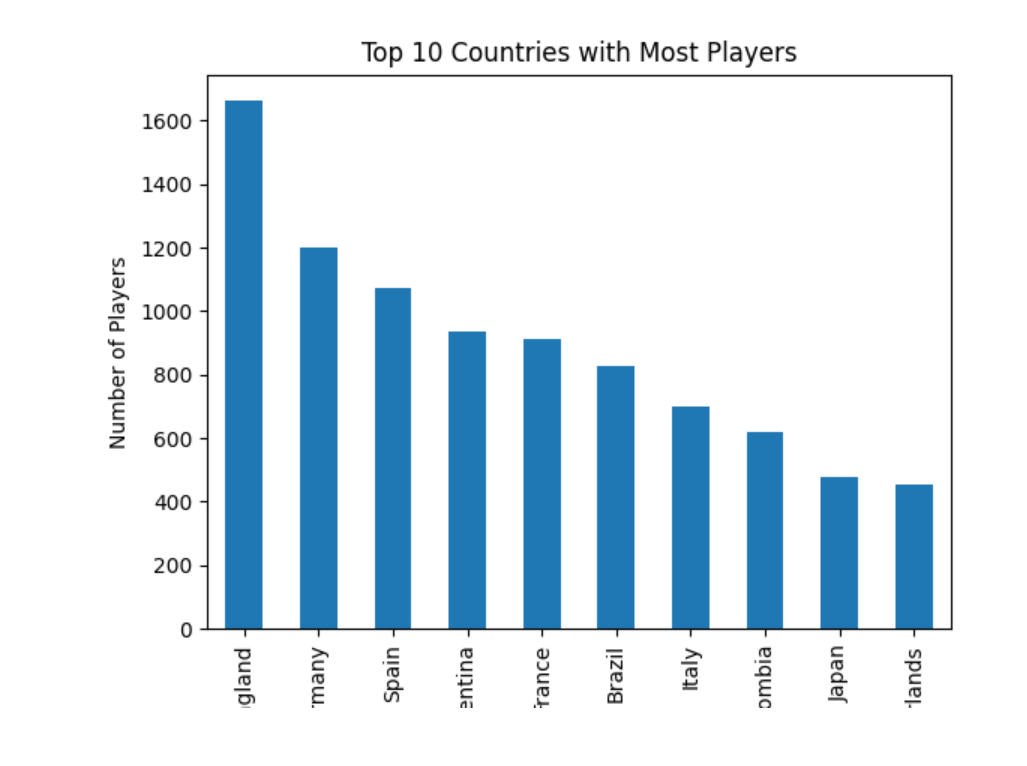
'Dribbling' and 'BallControl' are positively correlated.

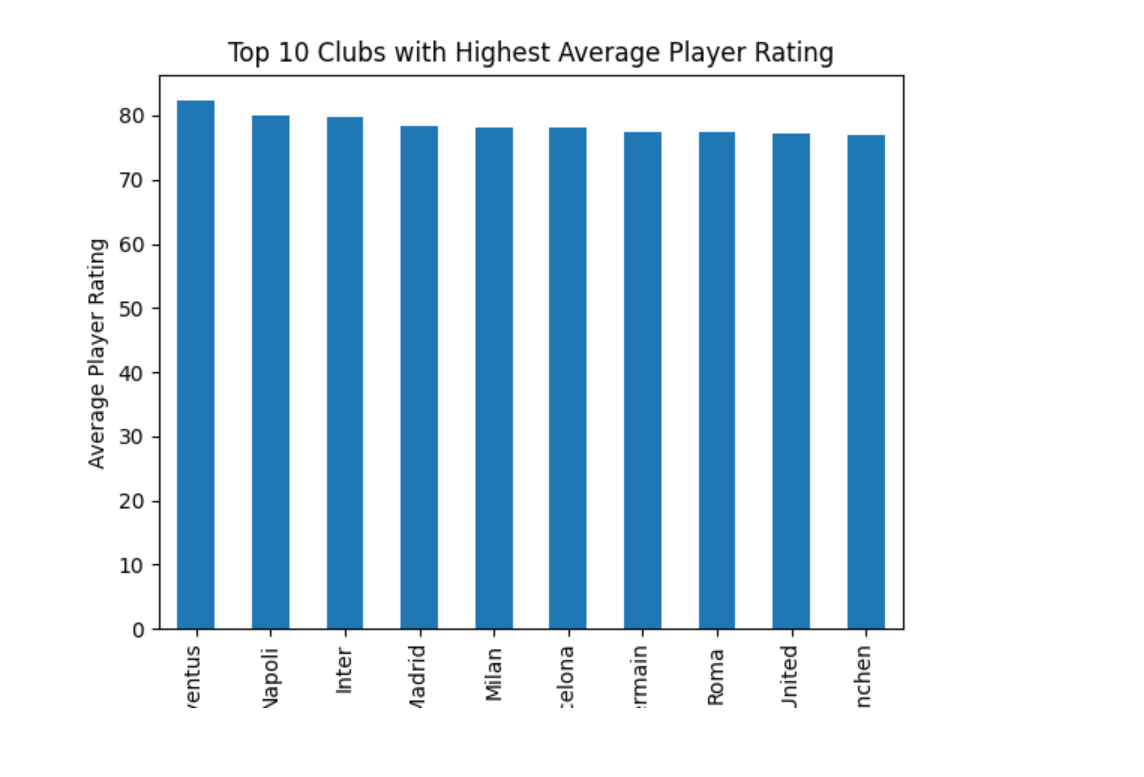
这是调用库中corr计算的情况 相关系数接近1 高度相关

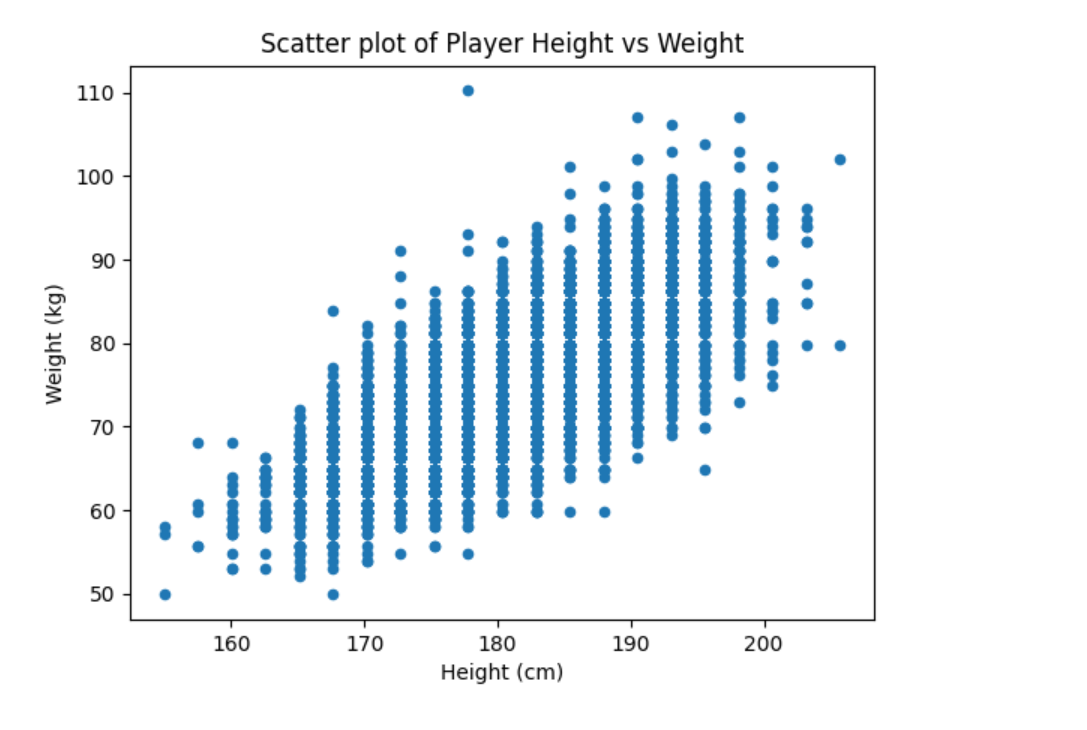
（3）The results are consistent. 

手动计算和调用库是一致的

8、扩展实验

探索问题：  
1、哪个国家球员最多？  
英国 德国 西班牙 阿根廷 法国  
  
2、哪个俱乐部平均球员评分最高？

  
3、球员的身高和体重有无相关性？

****

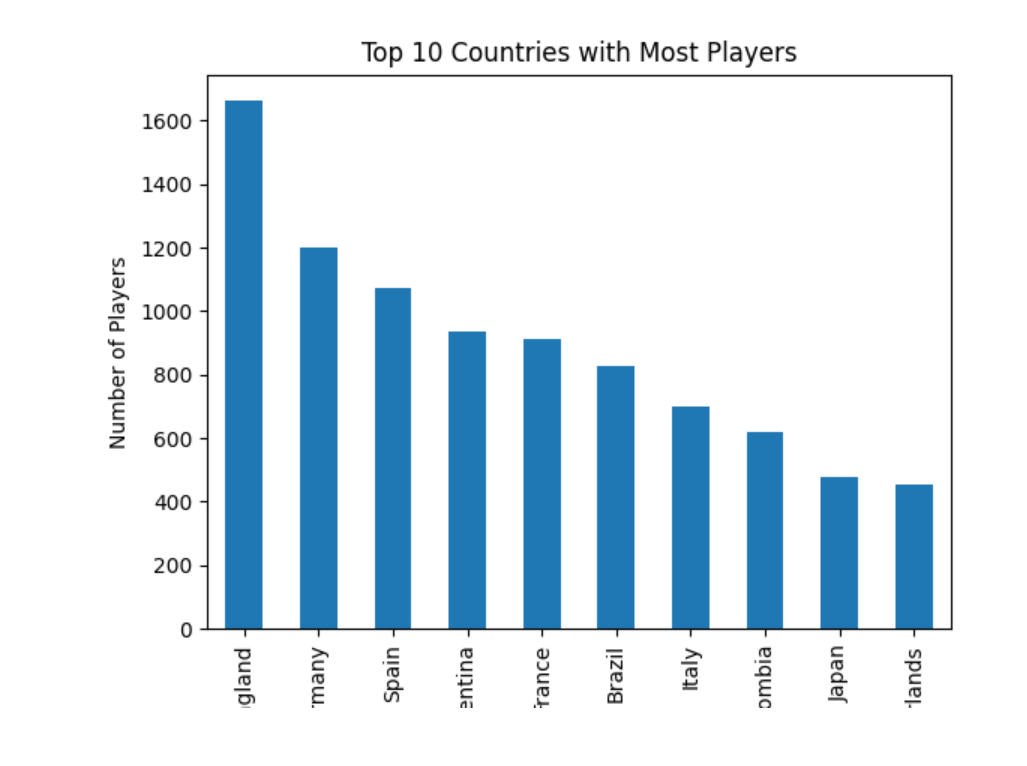
## 问题和解决

在cmd 使用pip安装python的一些库时，遇到错误 Unable to **create process using ……**

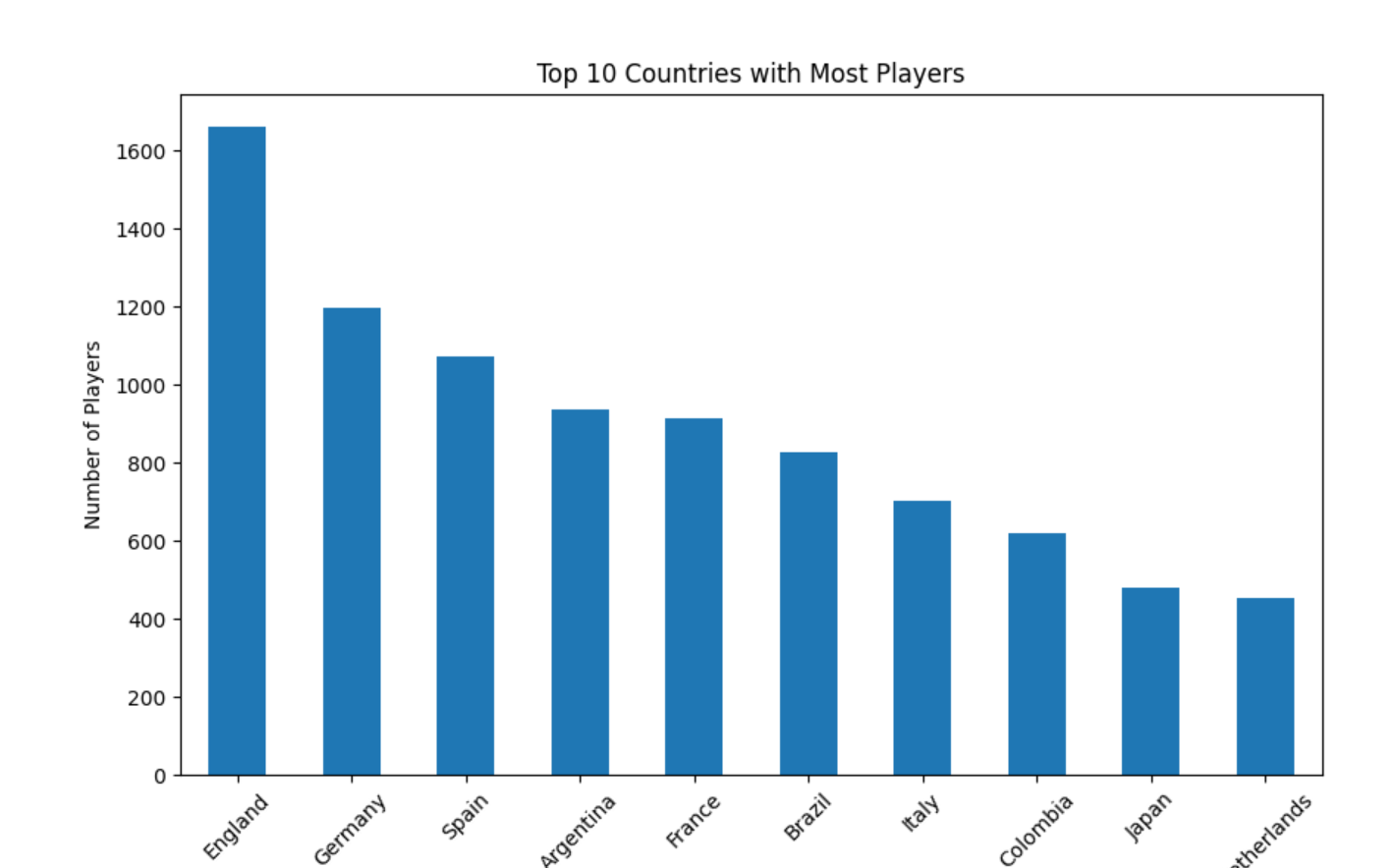
经过长时间错误排查，发现之前原因是改变了python目录后pip没有自动更改python依赖目录，之前python安装在c盘，我将其手动更改到e盘后，pip无法正常使用，误以为是环境变量出现了问题，但是经验证 （用where python和where pip）发现问题不是这里。后面查阅资料发现重装pip后问题得到解决，可能原因是重新让pip设置了依赖python的目录

还有个问题

有些时候得到的图片底部或者左侧有些文字看不清楚 可以通过旋转图片和改变文字大小进行修正  
比如

这是调整前的  


这是调整后的



我的代码仓库地址：[mozhongzhou/CQU\_Python\_experiment (github.com)](https://github.com/mozhongzhou/CQU_Python_experiment)