PYTHON数据分析（一）

数据可视化初探索

数据可视化

数据可视化指的是通过可视化表示来探索数据，它与数据挖掘紧密相关，而数据挖掘指的是使用代码来探索数据集的规律和关联。数据集可以是用一行代码就能表示的小型数字列表，也可以是数以及字节的数据。

——摘自《Python编程：从入门到实践》

数据可视化

我们可以这样理解，数据可视化和数据挖掘都是探索数据和分析数据的一种手段，只不过数据挖掘是以代码为探索途径，而数据可视化是将数据转换为图形、图表这样可视的形式来进行分析。

MATPLOTLIB

matplotlib是一个Python的2D绘图库，我们可以通过这个库将数据绘制成各种2D图形（直方图、散点图、条形图等）。

MATPLOTLIB

matplotlib是一个功能很强大的绘图库，其提供的一系列功能完善的api可以帮助我们快速地建立起我们所需的图形，接下来是几个简单的例子：

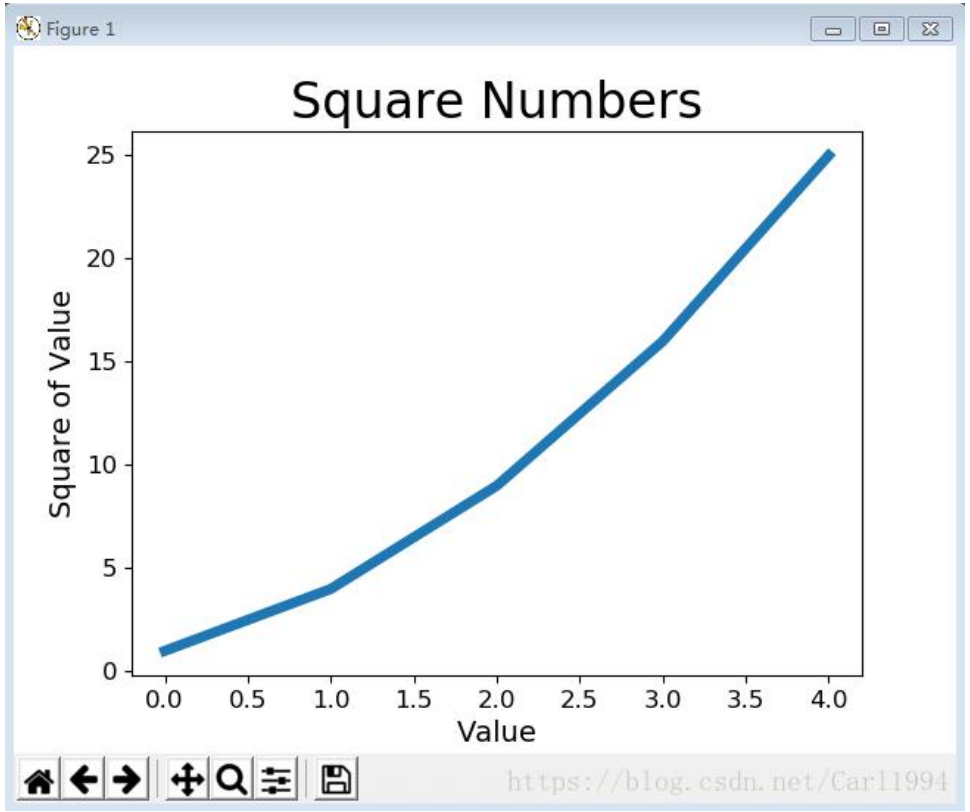
实例1-折线图

```
import matplotlib.pyplot as plt

squares = [1, 4, 9, 16, 25]
plt.plot(squares, linewidth=5)

plt.title("Square Numbers", fontsize=24)
plt.xlabel("Value", fontsize=14)
plt.ylabel("Square of Value", fontsize=14)

plt.tick_params(axis='both', labelsize=12)
plt.show()
```



- 上面的图中，左边的是绘制简单折线图的程序，右图是运行的结果

实例1-折线图

下面对代码的关键行进行了编号：

```
①import matplotlib.pyplot as plt

squares = [1, 4, 9, 16, 25]
②plt.plot(squares, linewidth=5)

③plt.title("Square Numbers", fontsize=24)
④plt.xlabel("Value", fontsize=14)
plt.ylabel("Square of Value", fontsize=14)

⑤plt.tick_params(axis='both', labelsize=12)
plt.show()
```

接下来我们会逐步对每一个编号的代码行进行分析。

实例1解析

①导入matplotlib.pyplot模块并赋予其别名plt，这是Python中常用的小技巧，目的主要是简化一些名称，当然我们也可以这样写：

```
from matplotlib import pyplot
```

不过这样一来，在后面调用到pyplot的方法时需要把之前的plt改为全称pyplot，运行结果不会有任何差别。

实例1解析

②plot()方法：这里将存放了一组平方数的列表传入plot()，它将会尝试根据这些数据绘制出有意义的图形。再调用show()即可将图形显示出来。实参linewidth=5指定了折线的宽度。

实例1解析

③title()方法：使用此方法为图标添加标题，实参fontsize=24指定了文字尺寸，后面的方法中该参数含义相同。

实例1解析

④xlabel()和ylabel()方法：为x轴和y轴命名。

实例1解析

⑤tick_params()方法：设置坐标轴刻度的样式，实参axis='both'表示同时设置两条轴，也可以指定为x或y单独设置。

实例1解析

相信大家发现了一个问题，我们传入的数据是一组平方值，而我们起始的数据是1的平方，但是x轴的起点却是0——事实上当你向plot()提供一系列数字时，它会默认x轴的0作为数据的起点，要改变这种默认配置，只需要对代码稍作修改，再提供一组x轴的值与平方数一一对应：

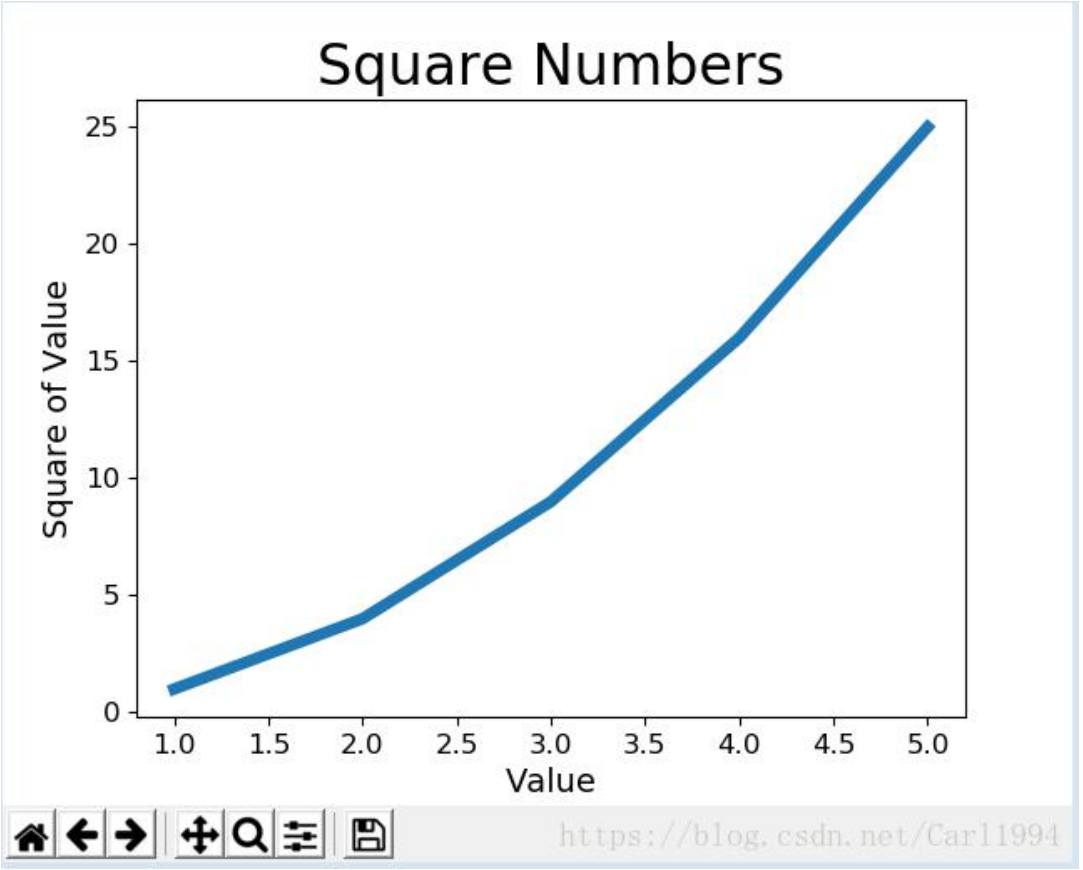
实例1解析

```
import matplotlib.pyplot as plt

input_values = [1, 2, 3, 4, 5]
squares = [1, 4, 9, 16, 25]
plt.plot(input_values, squares, linewidth=5)

plt.title("Square Numbers", fontsize=24)
plt.xlabel("Value", fontsize=14)
plt.ylabel("Square of Value", fontsize=14)

plt.tick_params(axis='both', labelsize=12)
plt.show()
```



实例2-散点图

在绘制折线图的时候，我们使用了plot()方法来接收数据，而对于散点图，则需要使用scatter()方法。我们直接来看看代码：

```
import matplotlib.pyplot as plt

①x_values = list(range(1, 101))  # 区分list()和range()
y_values = [x ** 2 for x in x_values]

②plt.scatter(x_values, y_values, c=y_values, cmap=plt.cm.Blues, edgecolors='none', s=40)
plt.title("Square Numbers", fontsize=24)
plt.xlabel("Value", fontsize=14)
plt.ylabel("Square of Value", fontsize=14)
plt.tick_params(axis="both", labelsize=14)
③plt.axis([0, 110, 0, 11000])
# plt.show()
④plt.savefig('squares_plot.png', bbox_inches='tight')
```

实例2解析

①x_values和y_values，定义了数据源，分别对应x轴的输入值和y轴的输出值，相互间的关系为y轴的输出值为x轴值的平方。在Python中，range()方法的含义是产生一个可迭代的对象，它的很多行为都与list相似，比如在遍历range(1, 101)并打印将会得到1至100的值，但是它与list是有本质区别的，即它在迭代的情况下返回的是一个索引值而非在内存中真正生成一个列表对象，所以在Python3中执行print(range(1, 101))，你将会得到的打印结果是range(1, 101)而非一个从1到100的列表。所以想要得到一个真正的列表则需要跟list()方法结合使用。对于这一句[x ** 2 for x in x_values]，可能看着有些奇怪，但并不难理解，我们可以将它看成遍历列表x_values，每次遍历时将取出一个x值将它做平方放入y_values中生成一个列表。

实例2解析

②scatter()方法：和上一个例子中的plot()方法类似，都是负责接收数据绘制图形，我们可以通过传入实参c、edgecolors、s来分别指定散点颜色、散点边缘颜色和散点大小，对于颜色可以直接传“red”、“blue”这种简单的颜色，也可以传入rgb色值。在本段代码中，则是利用颜色映射（colormap）来设置颜色，即代码中的实参cmap，结合c=y_values，绘制出的散点将根据y轴值由小到大颜色逐渐加深，基本颜色为蓝色。

实例2解析

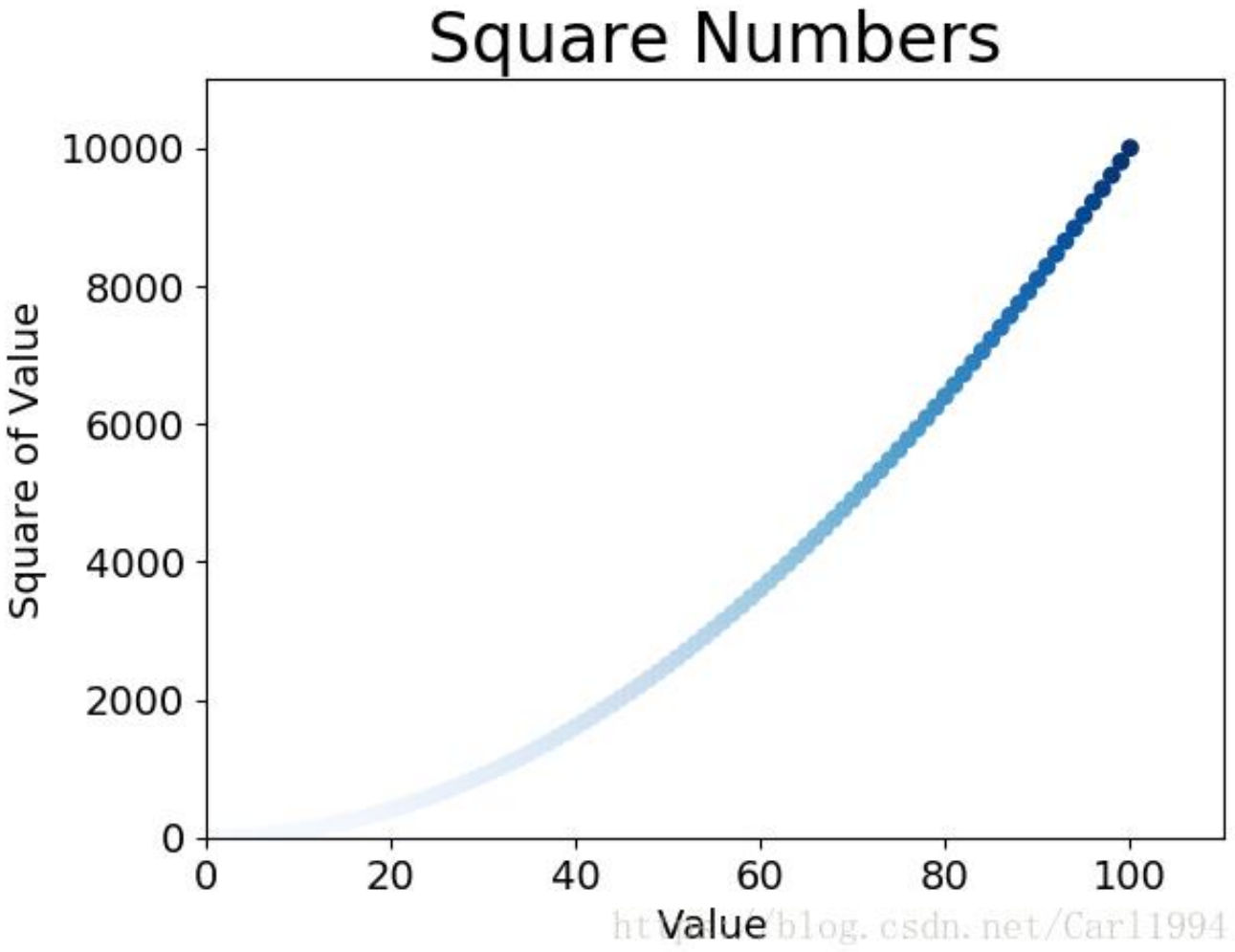
③axis()方法：指定每个坐标轴的取值范围，[x_min, x_max, y_min, y_max]。

实例2解析

④savefig()方法：可以注意到，在这个例子中，我将show()方法注释掉了，改为savefig()方法，这个方法将把绘制结果储存在项目目录下，第一个参数指定了图形文件名称，第二个参数表示将图表多余的空白区域裁掉。

实例2解析

运行结果是在项目目录下生成了一个图表，如右图：



作业

1.数字的三次方被称为其立方。请绘制一个图形，显示前5 个整数的立方值，再绘制一个图形，显示前5000 个整数的立方值。

2.给你前面绘制的立方图指定颜色映射。

鸣谢

本课程内容来源CSDN博客，作者arlTortoise 。