Python 绘图库 Matplotlib 入门教程

Python开发者 2018-04-17

> 来源：强波的技术博客
> qiangbo.space/2018-04-06/matplotlib_l1/

Matplotlib是一个Python语言的2D绘图库，它支持各种平台，并且功能强大，能够轻易绘制出各种专业的图像。本文是对它的一个入门教程。

运行环境

由于这是一个Python语言的软件包，因此需要你的机器上首先安装好Python语言的环境。关于这一点，请自行在网络上搜索获取方法。

关于如何安装Matplotlib请参见这里：Matplotlib Installing。

笔者推荐大家通过pip的方式进行安装，具体方法如下：

```
sudo pip3 install matplotlib
```

本文的代码在如下环境中测试：

- Apple OS X 10.13
- Python 3.6.3
- matplotlib 2.1.1
- numpy 1.13.3

介绍

Matplotlib适用于各种环境，包括：

- Python脚本
- IPython shell
- Jupyter notebook
- Web应用服务器
- 用户图形界面工具包

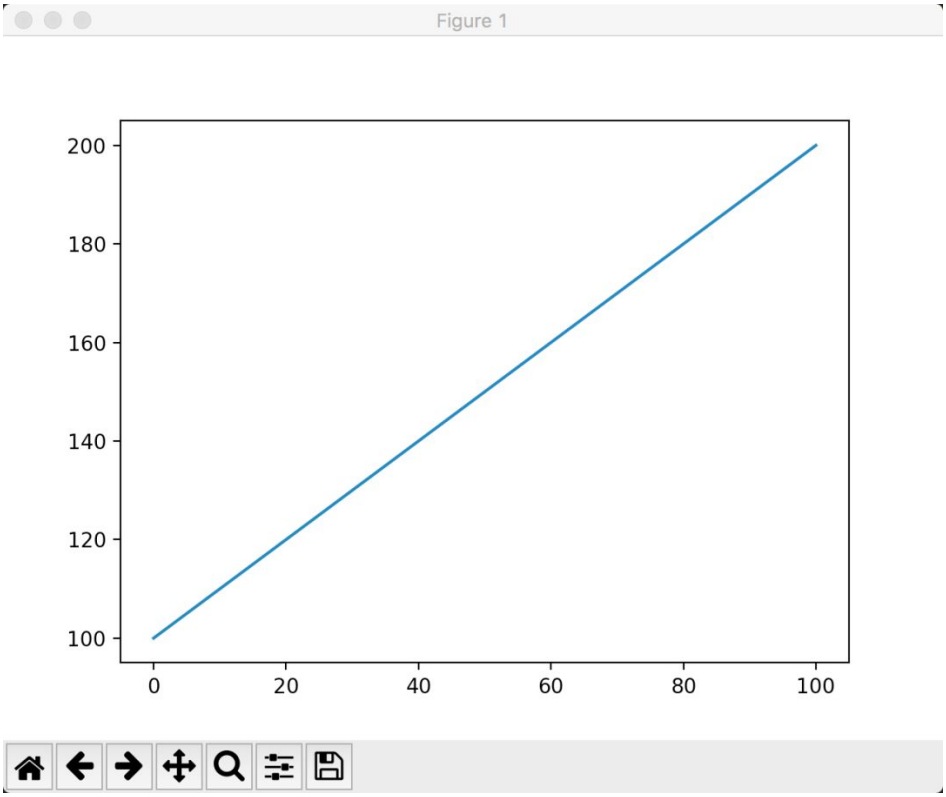
使用Matplotlib，能够的轻易生成各种类型的图像，例如：直方图，波谱图，条形图，散点图等。并且，可以非常轻松的实现定制。

入门代码示例

下面我们先看一个最简单的代码示例，让我们感受一下Matplotlib是什么样的：

```
# test.py

import matplotlib.pyplot as plt
import numpy as np

data = np.arange(100, 201)
plt.plot(data)
plt.show()
```

这段代码的主体逻辑只有三行，但是它却绘制出了一个非常直观的线性图，如下所示：



对照着这个线形图，我们来讲解一下三行代码的逻辑：

1. 通过np.arange(100, 201)生成一个[100, 200]之间的整数数组，它的值是：[100, 101, 102, ... , 200]
2. 通过matplotlib.pyplot将其绘制出来。很显然，绘制出来的值对应了图中的纵坐标（y轴）。而matplotlib本身为我们设置了图形的横坐标（x轴）：[0, 100]，因为我们刚好有100个数值
3. 通过plt.show()将这个图形显示出来

这段代码非常的简单，运行起来也是一样。如果你已经有了本文的运行环境，将上面的代码保存到一个文本文件中（或者通过Github获取本文的源码），然后通过下面的命令就可以在你自己的电脑上看到上面的图形了：

```
python3 test.py
```

> 注1：后面的教程中，我们会逐步讲解如何定制图中的每一个细节。例如：坐标轴，图形，着色，线条样式，等等。

> 注2：如果没有必要，下文的截图会去掉图形外侧的边框，只保留图形主体。

一次绘制多个图形

有些时候，我们可能希望一次绘制多个图形，例如：两组数据的对比，或者一组数据的不同展示方式等。

可以通过下面的方法创建多个图形：

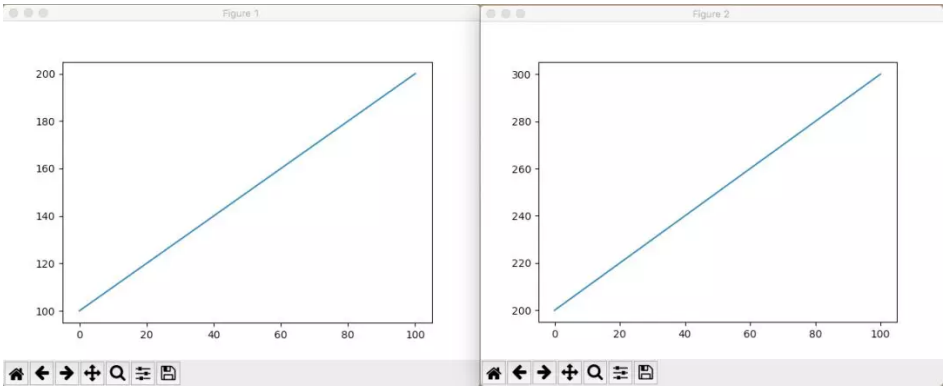
多个figure

可以简单的理解为一个figure就是一个图形窗口。matplotlib.pyplot会有一个默认的figure，我们也可以通过plt.figure()创建更多个。如下面的代码所示：

```
# figure.py

import matplotlib.pyplot as plt
```

```
import numpy as np

data = np.arange(100, 201)
plt.plot(data)

data2 = np.arange(200, 301)
plt.figure()
plt.plot(data2)

plt.show()
```

这段代码绘制了两个窗口的图形，它们各自是一个不同区间的线形图，如下所示：



> 注：初始状态这两个窗口是完全重合的。

多个subplot

有些情况下，我们是希望在同一个窗口显示多个图形。此时就这可以用多个subplot。下面是一段代码示例：

```
# subplot.py

import matplotlib.pyplot as plt
import numpy as np

data = np.arange(100, 201)
plt.subplot(2, 1, 1)
plt.plot(data)

data2 = np.arange(200, 301)
plt.subplot(2, 1, 2)
plt.plot(data2)

plt.show()
```

这段代码中，除了subplot函数之外都是我们熟悉的内容。subplot函数的前两个参数指定了subplot数量，即：它们是以矩阵的形式来分割当前图形，两个整数分别指定了矩阵的行数和列数。而第三个参数是指矩阵中的索引。

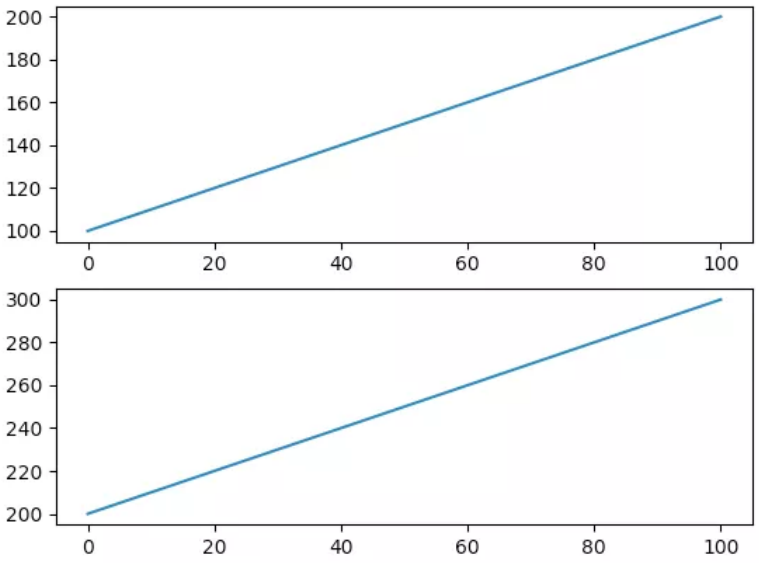
因此，下面这行代码指的是：2行1列subplot中的第1个subplot。

```
plt.subplot(2, 1, 1)
```

下面这行代码指的是：2行1列subplot中的第2个subplot。

```
plt.subplot(2, 1, 2)
```

所以这段代码的结果是这个样子：



subplot函数的参数不仅仅支持上面这种形式，还可以将三个整数（10之内的）合并一个整数。例如：2, 1, 1可以写成211，2, 1, 2可以写成212。

因此，下面这段代码的结果是一样的：

```
import matplotlib.pyplot as plt
import numpy as np

data = np.arange(100, 201)
plt.subplot(211)
plt.plot(data)

data2 = np.arange(200, 301)
plt.subplot(212)
plt.plot(data2)

plt.show()
```

> subplot函数的详细说明参见这里：matplotlib.pyplot.subplot

常用图形示例

Matplotlib可以生成非常多的图形式样，多到令人惊叹的地步。大家可以在这里：Matplotlib Gallery 感受一下。

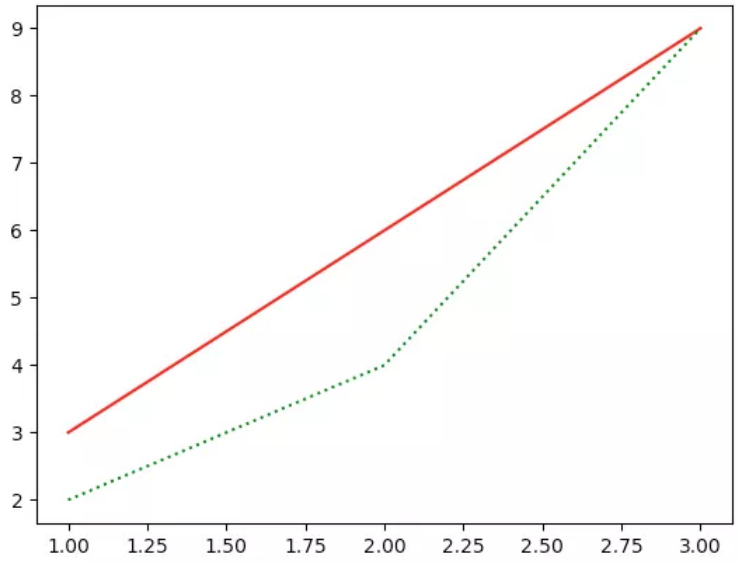
本文作为第一次的入门教程，我们先来看看最常用的一些图形的绘制。

线性图

前面的例子中，线性图的横轴的点都是自动生成的，而我们很可能希望主动设置它。另外，线条我们可能也希望对其进行定制。看一下下面这个例子：

```
# plot.py

import matplotlib.pyplot as plt

plt.plot([1, 2, 3], [3, 6, 9], '-r')
plt.plot([1, 2, 3], [2, 4, 9], ':g')

plt.show()
```

这段代码可以让我们得到这样的图形：



这段代码说明如下：

1. plot函数的第一个数组是横轴的值，第二个数组是纵轴的值，所以它们一个是直线，一个是折线；
2. 最后一个参数是由两个字符构成的，分别是线条的样式和颜色。前者是红色的直线，后者是绿色的点线。关于样式和颜色的说明请参见plot函数的API Doc：matplotlib.pyplot.plot

散点图

scatter函数用来绘制散点图。同样，这个函数也需要两组配对的数据指定x和y轴的坐标。下面是一段代码示例：

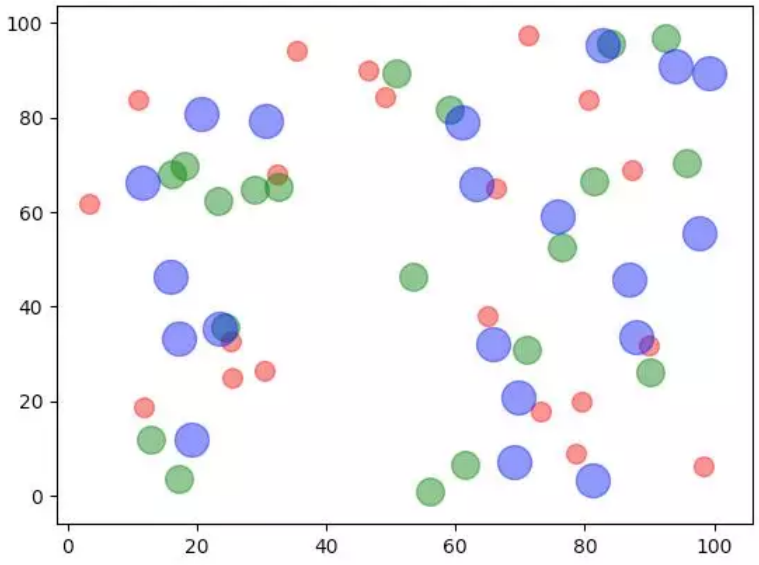
```
# scatter.py

import matplotlib.pyplot as plt
import numpy as np

N = 20

plt.scatter(np.random.rand(N) * 100,
        np.random.rand(N) * 100,
        c='r', s=100, alpha=0.5)

plt.scatter(np.random.rand(N) * 100,
        np.random.rand(N) * 100,
        c='g', s=200, alpha=0.5)

plt.scatter(np.random.rand(N) * 100,
        np.random.rand(N) * 100,
        c='b', s=300, alpha=0.5)
```

```
plt.show()
```

这段代码说明如下：

1. 这幅图包含了三组数据，每组数据都包含了20个随机坐标的位置
2. 参数c表示点的颜色，s是点的大小，alpha是透明度

这段代码绘制的图形如下所示：



> scatter函数的详细说明参见这里：matplotlib.pyplot.scatter

饼状图

pie函数用来绘制饼状图。饼状图通常用来表达集合中各个部分的百分比。

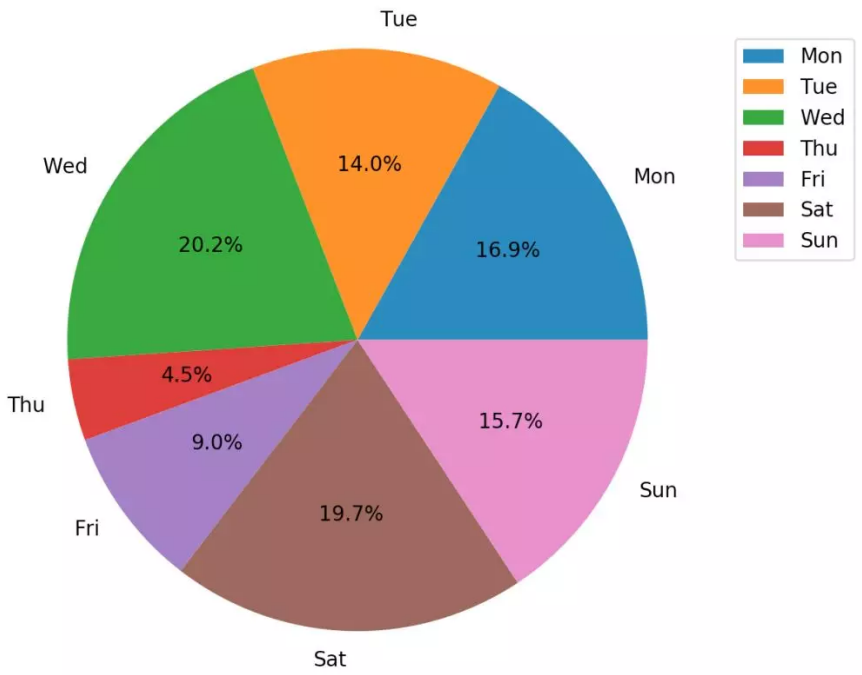
```
# pie.py

import matplotlib.pyplot as plt
import numpy as np

labels = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

data = np.random.rand(7) * 100

plt.pie(data, labels=labels, autopct='%1.1f%%')
plt.axis('equal')
plt.legend()

plt.show()
```

这段代码说明如下：

1. data是一组包含7个数据的随机数值
2. 图中的标签通过labels来指定
3. autopct指定了数值的精度格式
4. plt.axis('equal')设置了坐标轴大小一致

5. plt.legend()指明要绘制图例（见下图的右上角）

这段代码输出的图形如下所示：



> pie函数的详细说明参见这里：matplotlib.pyplot.pie

条形图

bar函数用来绘制条形图。条形图常常用来描述一组数据的对比情况，例如：一周七天，每天的城市车流量。

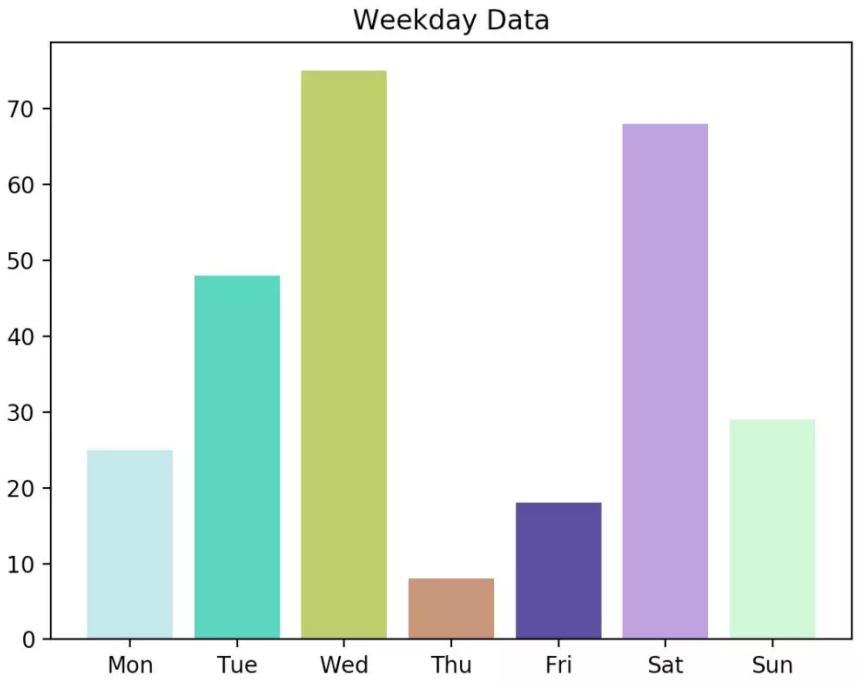
下面是一个代码示例：

```
# bar.py

import matplotlib.pyplot as plt
import numpy as np

N = 7

x = np.arange(N)
data = np.random.randint(low=0, high=100, size=N)
colors = np.random.rand(N * 3).reshape(N, -1)
labels = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

plt.title("Weekday Data")
plt.bar(x, data, alpha=0.8, color=colors, tick_label=labels)
plt.show()
```

这段代码说明如下：

1. 这幅图展示了一组包含7个随机数值的结果，每个数值是[0, 100]的随机数
2. 它们的颜色也是通过随机数生成的。np.random.rand(N * 3).reshape(N, -1)表示先生成21（N x 3）个随机数，然后将它们组装成7行，那么每行就是三个数，这对应了颜色的三个组成部分。如果不理解这行代码，请先学习一下Python 机器学习库 NumPy 教程
3. title指定了图形的标题，labels指定了标签，alpha是透明度

这段代码输出的图形如下所示：



> bar函数的详细说明参见这里：matplotlib.pyplot.bar

直方图

hist函数用来绘制直方图。直方图看起来是条形图有些类似。但它们的含义是不一样的，直方图描述了数据中某个范围内数据出现的频度。这么说有些抽象，我们通过一个代码示例来描述就好理解了：

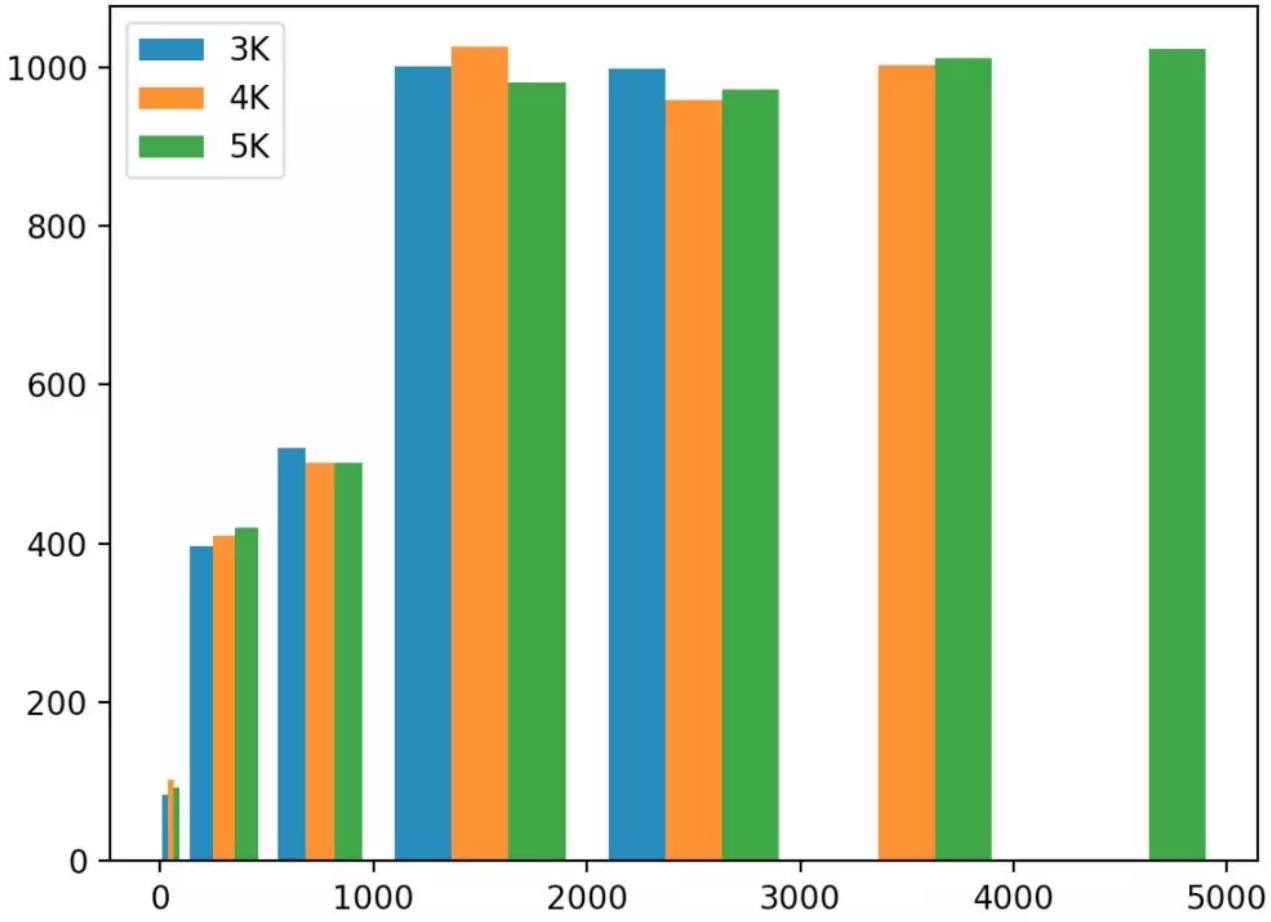
```
# hist.py

import matplotlib.pyplot as plt
import numpy as np

data = [np.random.randint(0, n, n) for n in [3000, 4000, 5000]]
labels = ['3K', '4K', '5K']
bins = [0, 100, 500, 1000, 2000, 3000, 4000, 5000]

plt.hist(data, bins=bins, label=labels)
plt.legend()

plt.show()
```

上面这段代码中，[np.random.randint(0, n, n) for n in [3000, 4000, 5000]]生成了包含了三个数组的数组，这其中：

- 第一个数组包含了3000个随机数，这些随机数的范围是 [0, 3000)
- 第二个数组包含了4000个随机数，这些随机数的范围是 [0, 4000)
- 第三个数组包含了5000个随机数，这些随机数的范围是 [0, 5000)

bins数组用来指定我们显示的直方图的边界，即：[0, 100) 会有一个数据点，[100, 500)会有一个数据点，以此类推。所以最终结果一共会显示7个数据点。同样的，我们指定了标签和图例。

这段代码的输出如下图所示：



在这幅图中，我们看到，三组数据在3000以下都有数据，并且频度是差不多的。但蓝色条只有3000以下的数据，橙色条只有4000以下的数据。这与我们的随机数组数据刚好吻合。

> hist函数的详细说明参见这里：matplotlib.pyplot.hist

结束语

通过本文，我们已经知道了Matplotlib的大致使用方法和几种最基本的图形的绘制方式。

需要说明的是，由于是入门教程，因此本文中我们只给出了这些函数和图形最基本的使用方法。但实际上，它们的功能远不止这么简单。因此本文中我们贴出了这些函数的API地址以便读者进一步的研究。

阅读原文