

第一回：Matplotlib初相识

一、认识matplotlib

Matplotlib是一个Python 2D绘图库，能够以多种硬拷贝格式和跨平台的交互式环境生成出版物质量的图形，用来绘制各种静态，动态，交互式的图表。

Matplotlib可用于Python脚本，Python和IPython Shell、Jupyter notebook，Web应用程序服务器和各种图形用户界面工具包等。

Matplotlib是Python数据可视化库中的泰斗，它已经成为python中公认的数据可视化工具，我们所熟知的pandas和seaborn的绘图接口其实也是基于matplotlib所作的高级封装。

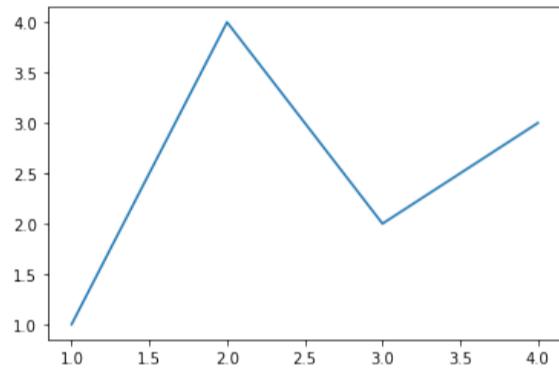
为了对matplotlib有更好的理解，让我们从一些最基本的概念开始认识它，再逐渐过渡到一些高级技巧中。

二、一个最简单的绘图例子

Matplotlib的图像是画在figure（如windows, jupyter窗体）上的，每一个figure又包含了一个或多个axes（一个可以指定坐标系的子区域）。最简单的创建figure以及axes的方式是通过`pyplot.subplots`命令，创建axes以后，可以使用`Axes.plot`绘制最简易的折线图。

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

fig, ax = plt.subplots() # 创建一个包含一个axes的figure
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]); # 绘制图像
```

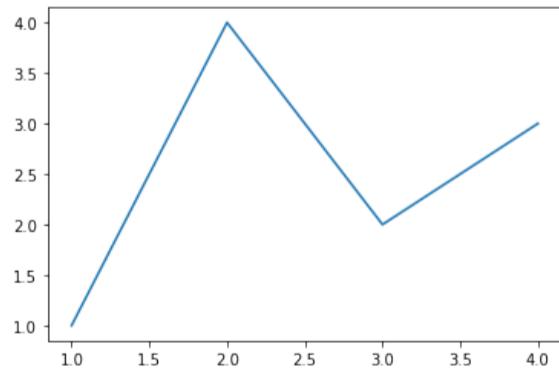


Trick：在jupyter notebook中使用matplotlib时会发现，代码运行后自动打印出类似`<matplotlib.lines.Line2D at 0x23155916dc0>`这样一段话，这是因为matplotlib的绘图代码默认打印出最后一个对象。如果不想显示这句话，有以下三种方法，在本章节的代码示例中你能找到这三种方法的使用。

1. 在代码块最后加一个分号；
2. 在代码块最后加一句`plt.show()`；
3. 在绘图时将绘图对象显式赋值给一个变量，如将`plt.plot([1, 2, 3, 4])`改成`line = plt.plot([1, 2, 3, 4])`

和MATLAB命令类似，你还可以通过一种更简单的方式绘制图像，`matplotlib.pyplot`方法能够直接在当前axes上绘制图像，如果用户未指定axes，matplotlib会帮你自动创建一个。所以上面的例子也可以简化为以下这一行代码。

```
line = plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
```



三、Figure的组成

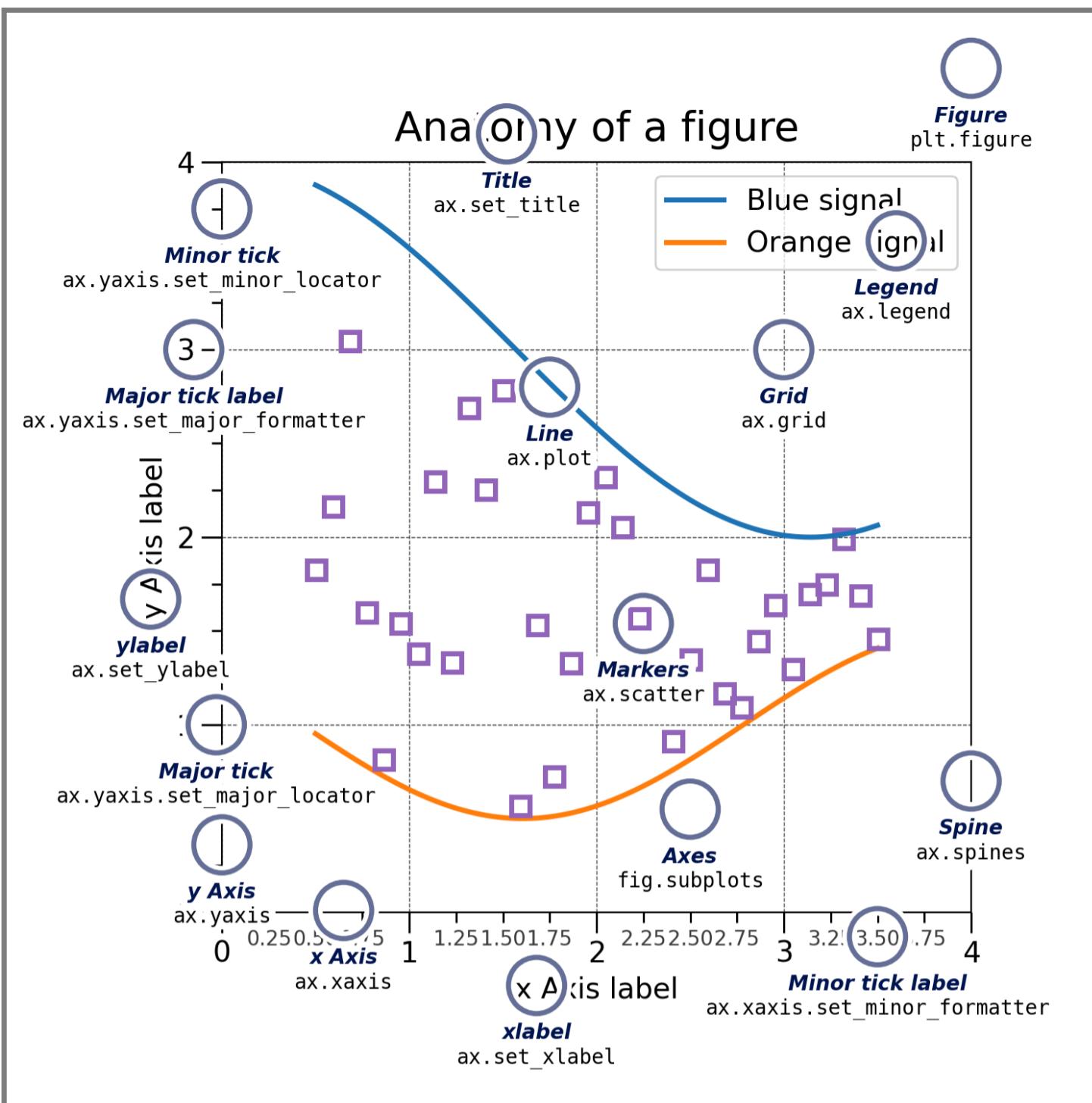
现在我们来深入看一下figure的组成。通过一张figure解剖图，我们可以看到一个完整的matplotlib图像通常会包括以下四个层级，这些层级也被称为容器（container），下一节会详细介绍。在matplotlib的世界中，我们将通过各种命令方法来操纵图像中的每一个部分，从而达到数据可视化的最终效果，一副完整的图像实际上是各类子元素的集合。

- **Figure**: 顶层级，用来容纳所有绘图元素
- **Axes**: matplotlib宇宙的核心，容纳了大量元素用来构造一幅幅子图，一个figure可以由一个或多个子图组成
- **Axis**: axes的下属层级，用于处理所有和坐标轴，网格有关的元素
- **Tick**: axis的下属层级，用来处理所有和刻度有关的元素

Contents

- [二、认识matplotlib](#)
- [三、Figure的组成](#)
- [四、两种绘图接口](#)
- [五、通用绘图模板](#)

Print to PDF ▶



四、两种绘图接口

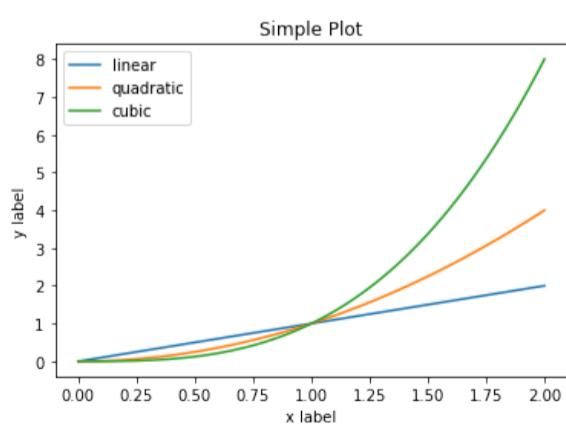
matplotlib提供了两种最常用的绘图接口

1. 显式创建figure和axes，在上面调用绘图方法，也被称为OO模式（object-oriented style）
2. 依赖pyplot自动创建figure和axes，并绘图

使用第一种绘图接口，是这样的：

```
x = np.linspace(0, 2, 100)

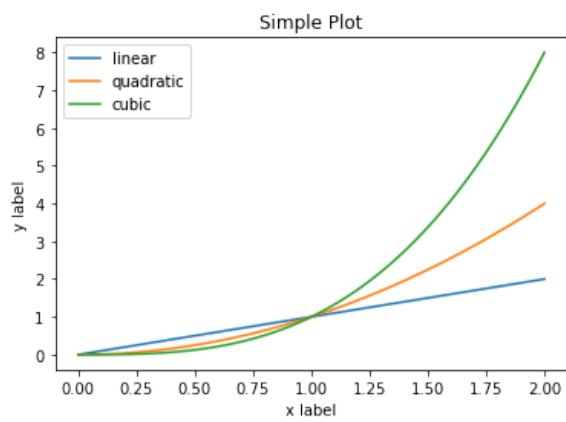
fig, ax = plt.subplots()
ax.plot(x, x, label='linear')
ax.plot(x, x**2, label='quadratic')
ax.plot(x, x**3, label='cubic')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title("Simple Plot")
ax.legend()
plt.show()
```



而如果采用第二种绘图接口，绘制同样的图，代码是这样的：

```
x = np.linspace(0, 2, 100)

plt.plot(x, x, label='linear')
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
plt.show()
```



五、通用绘图模板

由于matplotlib的知识点非常繁杂，在实际使用过程中也不可能将全部API都记住，很多时候都是边用边查。因此这里提供一个通用的绘图基础模板，任何复杂的图表几乎都可以基于这个模板骨架填充内容而成。初学者刚开始学习时只需要牢记这一模板就足以应对大部分简单图表的绘制，在学习过程中可以将这个模板模块化，了解每个模块在做什么，在绘制复杂图表时如何修改，填充对应的模块。

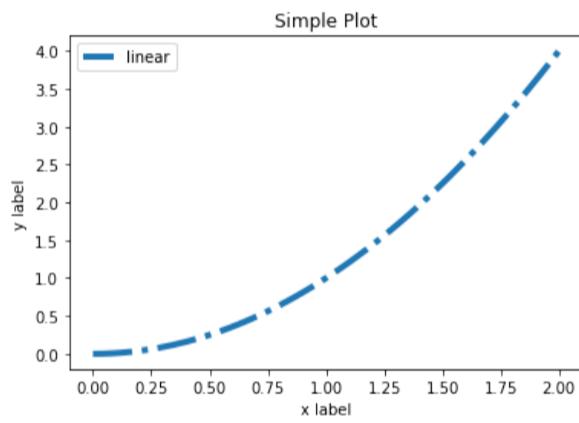
```
# step1 准备数据
x = np.linspace(0, 2, 100)
y = x**2

# step2 设置绘图样式，这一模块的扩展参考第五章进一步学习，这一步不是必须的，样式也可以在绘制图像
# 是进行设置
mpl.rcParams['lines', linewidth=4, linestyle='-.']

# step3 定义布局，这一模块的扩展参考第三章进一步学习
fig, ax = plt.subplots()

# step4 绘制图像，这一模块的扩展参考第二章进一步学习
ax.plot(x, y, label='linear')

# step5 添加标签，文字和图例，这一模块的扩展参考第四章进一步学习
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title("Simple Plot")
ax.legend() ;
```



思考题

- 请思考两种绘图模式的优缺点和各自适合的使用场景
- 在第五节绘图模板中我们是以OO模式作为例子展示的，请思考并写一个pyplot绘图模式的简单模板