基本用法

本章知识点归纳如下,我们可以通过这三种方法中的任一种实现多图：

1.plt.subplot()

2.plt.subplot2grid()

3.gridspec.GridSpec()

4.plt.subplots()

5.图中图：fig.add_axes()

6.次坐标轴：ax.twinx()

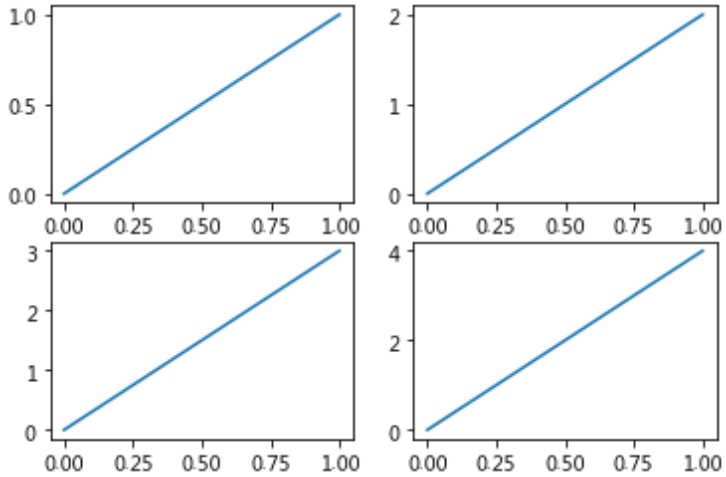
Subplot 多合一显示

1.均匀图中图

首先使用import导入matplotlib.pyplot模块, 并简写成plt。 使用plt.figure创建一个图像窗口.使用plt.subplot来创建小图。plt.subplot(2,2,1)表示将整个图像窗口分为2行2列, 当前位置为1. 使用plt.plot([0,1],[0,1])在第1个位置创建一个小图。plt.subplot(2,2,2)表示将整个图像窗口分为2行2列, 当前位置为2。使用plt.plot([0,1],[0,2])在第2个位置创建一个小图。plt.subplot(2,2,3)表示将整个图像窗口分为2行2列,当前位置为3。 plt.subplot(2,2,3)可以简写成plt.subplot(223), matplotlib同样可以识别。使用plt.plot([0,1],[0,3])在第3个位置创建一个小图。plt.subplot(224)表示将整个图像窗口分为2行2列, 当前位置为4。使用plt.plot([0,1],[0,4])在第4个位置创建一个小图。

In [5]:

```
import matplotlib.pyplot as plt
plt.figure()

plt.subplot(2,2,1)
plt.plot([0,1],[0,1])

plt.subplot(2,2,2)
plt.plot([0,1],[0,2])

plt.subplot(223)
plt.plot([0,1],[0,3])

plt.subplot(224)
plt.plot([0,1],[0,4])

plt.show()  # 展示
```



2.不均匀图中图

如果希望展示的小图的大小不相同, 应该怎么做呢? 以上面的4个小图为例, 如果把第1个小图放到第一行, 而剩下的3个小图都放到第二行。使用plt.subplot(2,1,1)将整个图像窗口分为2行1列, 当前位置为1。使用plt.plot([0,1],[0,1])在第1个位置创建一个小图。使用plt.subplot(2,3,4)将整个图像窗口分为2行3列, 当前位置为4。使用plt.plot([0,1],[0,2])在第4个位置创建一个小图。

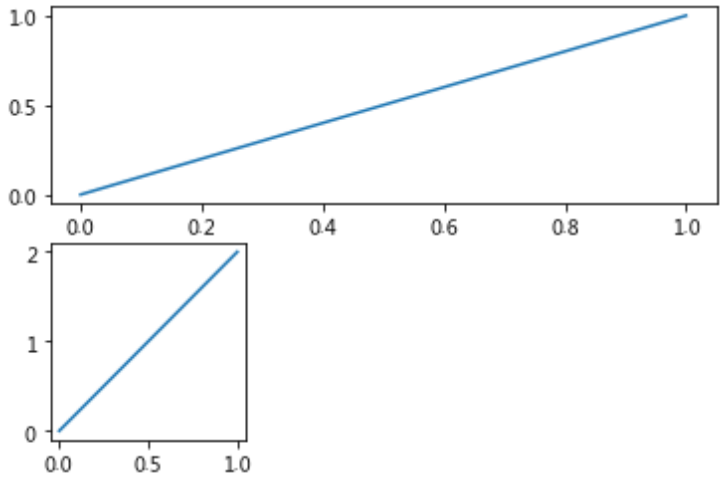
这里需要解释一下为什么第4个位置放第2个小图。上一步中使用plt.subplot(2,1,1)将整个图像窗口分为2行1列, 第1个小图占用了第1个位置, 也就是整个第1行。这一步中使用plt.subplot(2,3,4)将整个图像窗口分为2行3列, 于是整个图像窗口的第1行就变成了3列, 也就是成了3个位置, 于是第2行的第1个位置是整个图像窗口的第4个位置。

In [6]:

```
plt.subplot(2,1,1)
plt.plot([0,1],[0,1])

plt.subplot(2,3,4)
plt.plot([0,1],[0,2])
```

Out[6]:

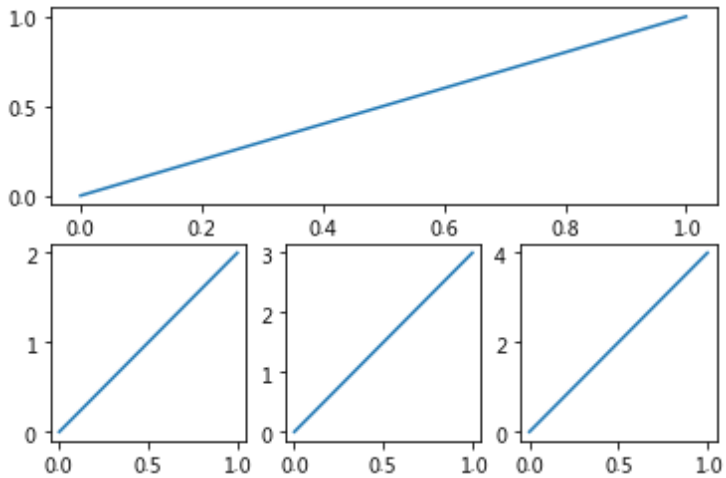
```
[<matplotlib.lines.Line2D at 0x7f06584b3f60>]
```



接着，使用plt.subplot(235)将整个图像窗口分为2行3列,当前位置为5。使用plt.plot([0,1],[0,3])在第5个位置创建一个小图。同上, 再创建plt.subplot(236).

In [7]:

```
plt.subplot(2,1,1)
plt.plot([0,1],[0,1])

plt.subplot(2,3,4)
plt.plot([0,1],[0,2])

plt.subplot(235)
plt.plot([0,1],[0,3])

plt.subplot(236)
plt.plot([0,1],[0,4])

plt.show()  # 展示
```



matplotlib 的 subplot 还可以是分格的,这里介绍三种方法同样也能达到 subplot() 函数的效果：subplot2grid、gridspec 和 subplots 。他们相比起普通的 subplot 会更加方便，在判断图的编号时不需要进行很复杂的考虑。

subplot2grid

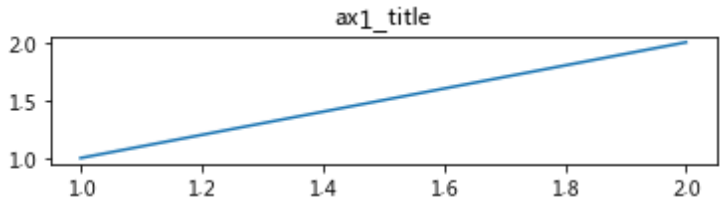
使用plt.subplot2grid来创建第1个小图, (3,3)表示将整个图像窗口分成3行3列, (0,0)表示从第0行第0列开始作图，colspan=3表示列的跨度为3, rowspan=1表示行的跨度为1。colspan和rowspan缺省, 表示默认跨度为1。

In [8]:

```
ax1 = plt.subplot2grid((3, 3), (0, 0), colspan=3)
ax1.plot([1, 2], [1, 2])     # 画小图
ax1.set_title('ax1_title')  # 设置小图的标题
```

Out[8]:

```
Text(0.5,1,'ax1_title')
```



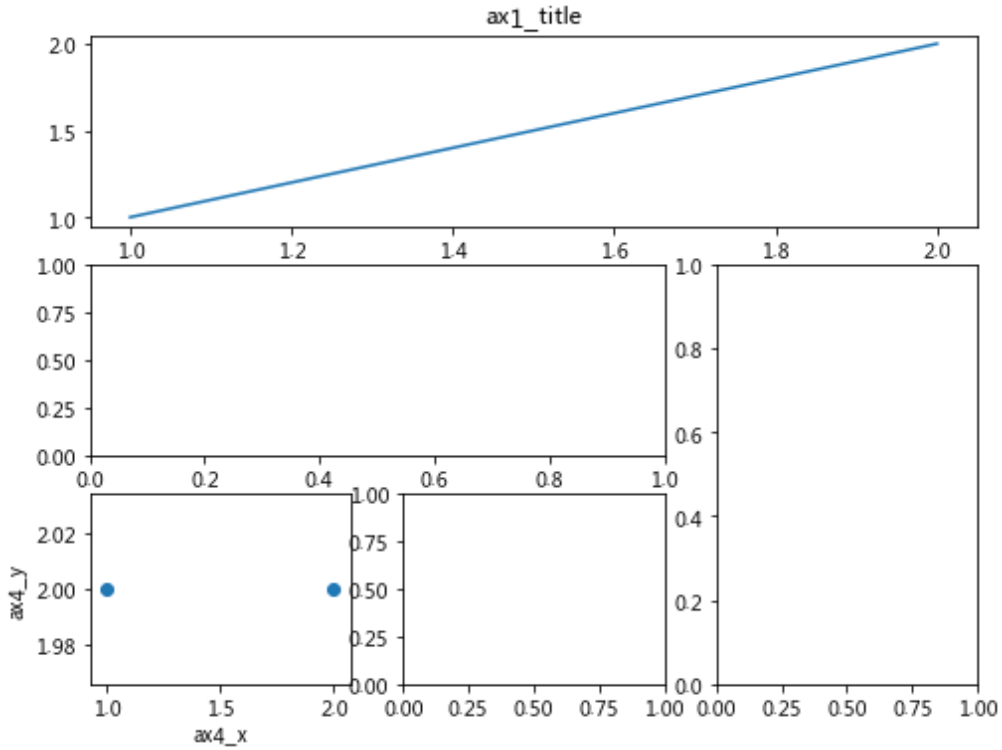
使用plt.subplot2grid来创建第2个小图, (3,3)表示将整个图像窗口分成3行3列, (1,0)表示从第1行第0列开始作图，colspan=2表示列的跨度为2. 同上画出 ax3, (1,2)表示从第1行第2列开始作图，rowspan=2表示行的跨度为2. 再画一个 ax4 和 ax5, 使用默认 colspan, rowspan。使用ax4.scatter创建一个散点图, 使用 ax4.setxlabel 和 ax4.setylabel 来对x轴和y轴命名。这样，我们就通过 subplot2grid() 完成了一张不均匀图中图。 *细心的小伙伴可能可以注意到，在第一章我们设置标题与坐标轴时，使用的是plt.title()这样的语句，针对小图 ax ，我们的设置前都要加上 'set'。*

In [15]:

```
plt.figure(figsize=(8, 6))
ax1 = plt.subplot2grid((3, 3), (0, 0), colspan=3)
ax1.plot([1, 2], [1, 2])    # 画小图
ax1.set_title('ax1_title')  # 设置小图的标题
ax2 = plt.subplot2grid((3, 3), (1, 0), colspan=2)
ax3 = plt.subplot2grid((3, 3), (1, 2), rowspan=2)
ax4 = plt.subplot2grid((3, 3), (2, 0))
ax5 = plt.subplot2grid((3, 3), (2, 1))
ax4.scatter([1, 2], [2, 2])
ax4.set_xlabel('ax4_x')
ax4.set_ylabel('ax4_y')
```

Out[15]:

```
Text(0,0.5,'ax4_y')
```



gridspec

gridspec 同样能帮助我们画出前文的图，个人觉得 gridspec 的使用是最为方便的，因为它允许我们使用索引的方式指定小图的大小和位置。首先，利用import导入matplotlib.pyplot模块, 并简写成plt. 利用import导入matplotlib.gridspec, 并简写成gridspec.

In [1]:

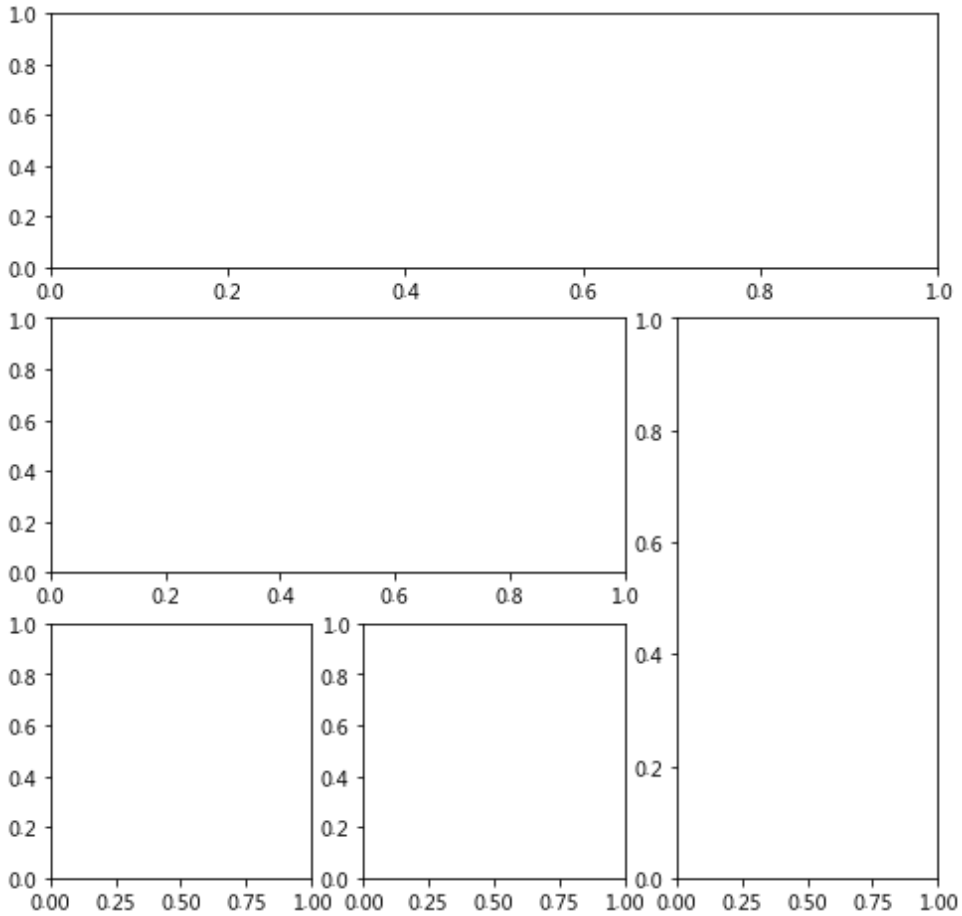
```
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
```

```
/opt/conda/lib/python3.5/site-packages/matplotlib/font_manager.py:278: UserWarning: Matplotlib is building the font cache using fc-list. This may ta
  'Matplotlib is building the font cache using fc-list.')
```

使用plt.figure()创建一个图像窗口, 使用gridspec.GridSpec将整个图像窗口分成3行3列.使用plt.subplot来作图, gs[0, :]表示这个图占第0行和所有列, gs[1, :2]表示这个图占第1行和第2列前的所有列, gs[1:, 2]表示这个图占第1行后的所有行和第2列, gs[-1, 0]表示这个图占倒数第1行和第0列, gs[-1, -2]表示这个图占倒数第1行和倒数第2列.

In [8]:

```
plt.figure(figsize=(8, 8))
gs = gridspec.GridSpec(3, 3)

ax6 = plt.subplot(gs[0, :])
ax7 = plt.subplot(gs[1, :2])
ax8 = plt.subplot(gs[1:, 2])
ax9 = plt.subplot(gs[-1, 0])
ax10 = plt.subplot(gs[-1, -2])
```



subplots

subplots 不同于 subplot, 它允许我们将图像窗口集合在一起表示。请看下面的代码：

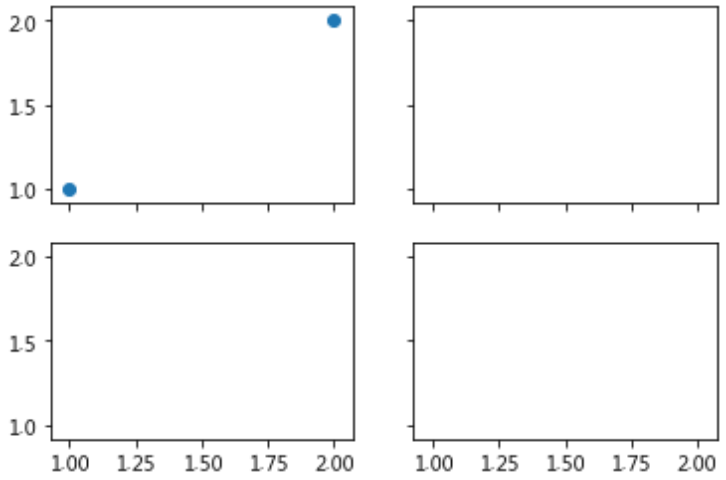
首先，使用plt.subplots建立一个2行2列的图像窗口，sharex=True表示共享x轴坐标, sharey=True表示共享y轴坐标. ((ax11, ax12), (ax13, ax14))表示第1行从左至右依次放ax11和ax12, 第2行从左至右依次放ax13和ax14.接着使用ax11.scatter创建一个散点图.

In [10]:

```
f, ((ax11, ax12), (ax13, ax14)) = plt.subplots(2, 2, sharex=True, sharey=True)
ax11.scatter([1,2], [1,2])
```

Out[10]:

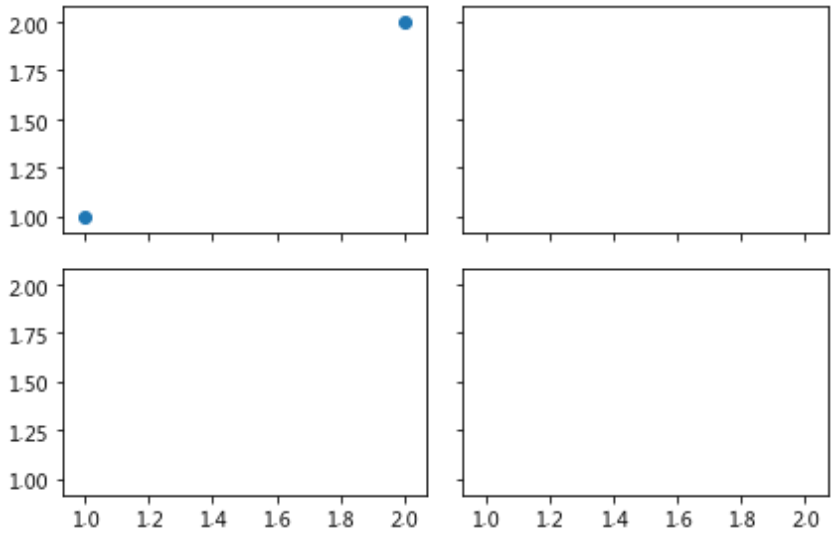
```
<matplotlib.collections.PathCollection at 0x7f6180777ef0>
```



plt.tight_layout()表示紧凑显示图像, plt.show()表示显示图像.

In [12]:

```
f, ((ax11, ax12), (ax13, ax14)) = plt.subplots(2, 2, sharex=True, sharey=True)
ax11.scatter([1,2], [1,2])
plt.tight_layout()
plt.show()
```



图中图

接下来我们来介绍 matplotlib 里一个很有意思的功能，叫做图中图(plot in plot)。通过它，我们可以在大图中嵌入小图。

1.生成数据

In [20]:

```
# 导入pyplot模块
import matplotlib.pyplot as plt

# 创建数据
x = [1, 2, 3, 4, 5, 6, 7]
y = [1, 3, 4, 2, 5, 8, 6]
```

```
<Figure size 432x288 with 0 Axes>
```

2.绘制大图

首先确定大图左下角的位置以及宽高：

In [14]:

```
left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
```

注意，4个值都是占整个figure窗口的百分比。在这里，假设figure的大小是10x10，那么大图就被包含在由(1, 1)开始，宽8，高8的坐标系内。在klab上大家可能并不能理解figure的窗口大小，因为我们的图片会直接在代码下方显示，但对于其他平台，会出现弹出图片窗口的情况，这里的 figure 大小指的就是弹出窗口的大小。

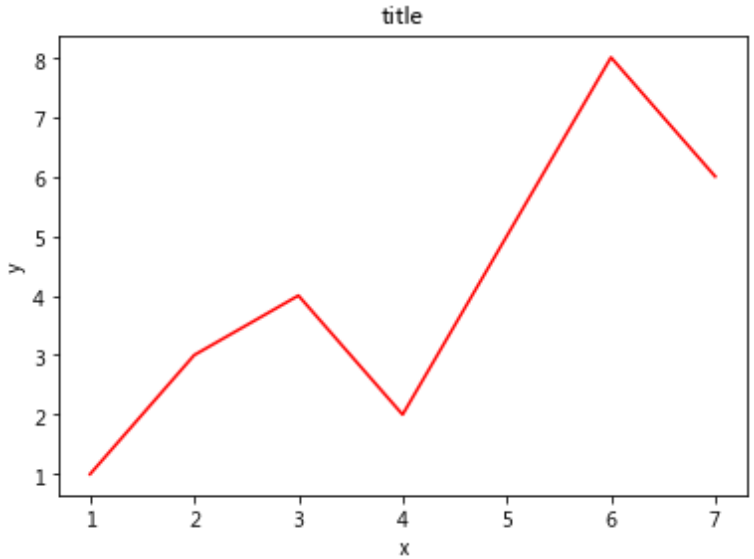
接着，我们将大图坐标系添加到 figure 中，颜色为 r(red)，取名为 title：

In [28]:

```
fig = plt.figure()

ax1 = fig.add_axes([left, bottom, width, height])
ax1.plot(x, y, 'r')
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title('title')
```

Out[28]:

```
Text(0.5,1,'title')
```



3.绘制小图

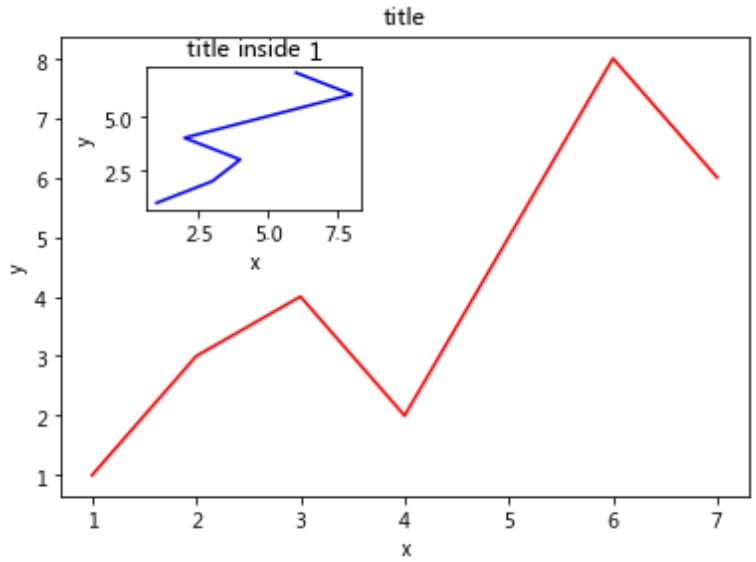
接着，我们来绘制左上角的小图，步骤和绘制大图一样，注意坐标系位置和大小的改变：

In [29]:

```
fig = plt.figure()

left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
ax1 = fig.add_axes([left, bottom, width, height])
ax1.plot(x, y, 'r')
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title('title')

left, bottom, width, height = 0.2, 0.6, 0.25, 0.25
ax2 = fig.add_axes([left, bottom, width, height])
ax2.plot(y, x, 'b')
ax2.set_xlabel('x')
ax2.set_ylabel('y')
ax2.set_title('title inside 1')
```

Out[29]:

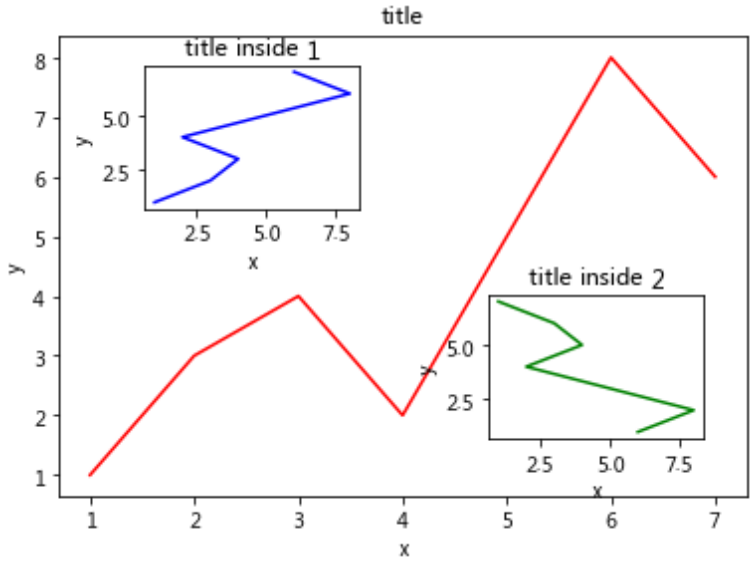
```
Text(0.5,1,'title inside 1')
```



最后，我们再来绘制右下角的小图。这里我们采用一种更简单方法，即直接往plt里添加新的坐标系：

In [32]:

```
fig = plt.figure()

left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
ax1 = fig.add_axes([left, bottom, width, height])
ax1.plot(x, y, 'r')
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title('title')

left, bottom, width, height = 0.2, 0.6, 0.25, 0.25
ax2 = fig.add_axes([left, bottom, width, height])
ax2.plot(y, x, 'b')
ax2.set_xlabel('x')
ax2.set_ylabel('y')
ax2.set_title('title inside 1')

plt.axes([0.6, 0.2, 0.25, 0.25])
plt.plot(y[::-1], x, 'g') # 注意对y进行了逆序处理
plt.xlabel('x')
plt.ylabel('y')
plt.title('title inside 2')

plt.show()
```



次坐标轴

有时候我们会用到次坐标轴，即在同个图上有第2个y轴存在。这件事同样可以用matplotlib做到，而且很简单。首先，我们做一些准备工作：

1.数据生成：

In [36]:

```
x = np.arange(0, 10, 0.1)
y1 = 0.05 * x**2
y2 = -1 * y1
```

2.设置两个坐标系并画图：

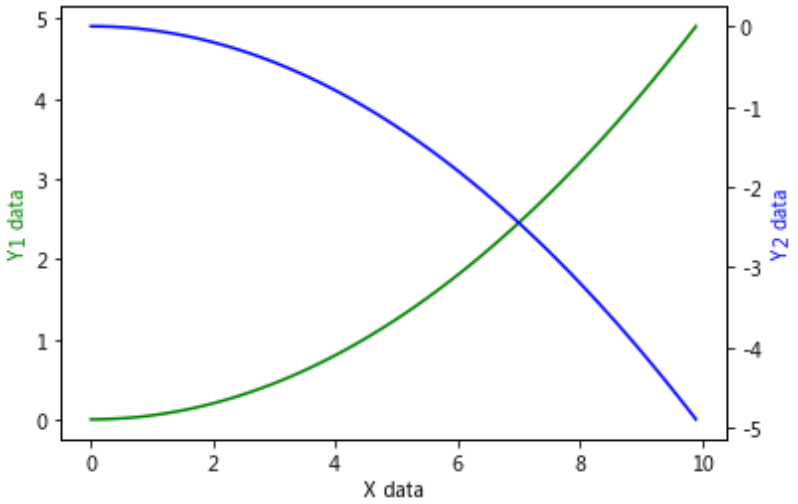
可以看到，y2和y1是互相倒置的。所以，我们先获取figure默认的坐标系 ax1；然后对ax1调用twinx()方法，生成如同镜面效果后的ax2；最后接着进行绘图, 将 y1, y2 分别画在 ax1, ax2 上：

In [39]:

```
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()

ax1.plot(x, y1, 'g-')    # green, solid line
ax1.set_xlabel('X data')
ax1.set_ylabel('Y1 data', color='g')
ax2.plot(x, y2, 'b-') # blue
ax2.set_ylabel('Y2 data', color='b')
```

Out[39]:

```
Text(0,0.5,'Y2 data')
```



练一练

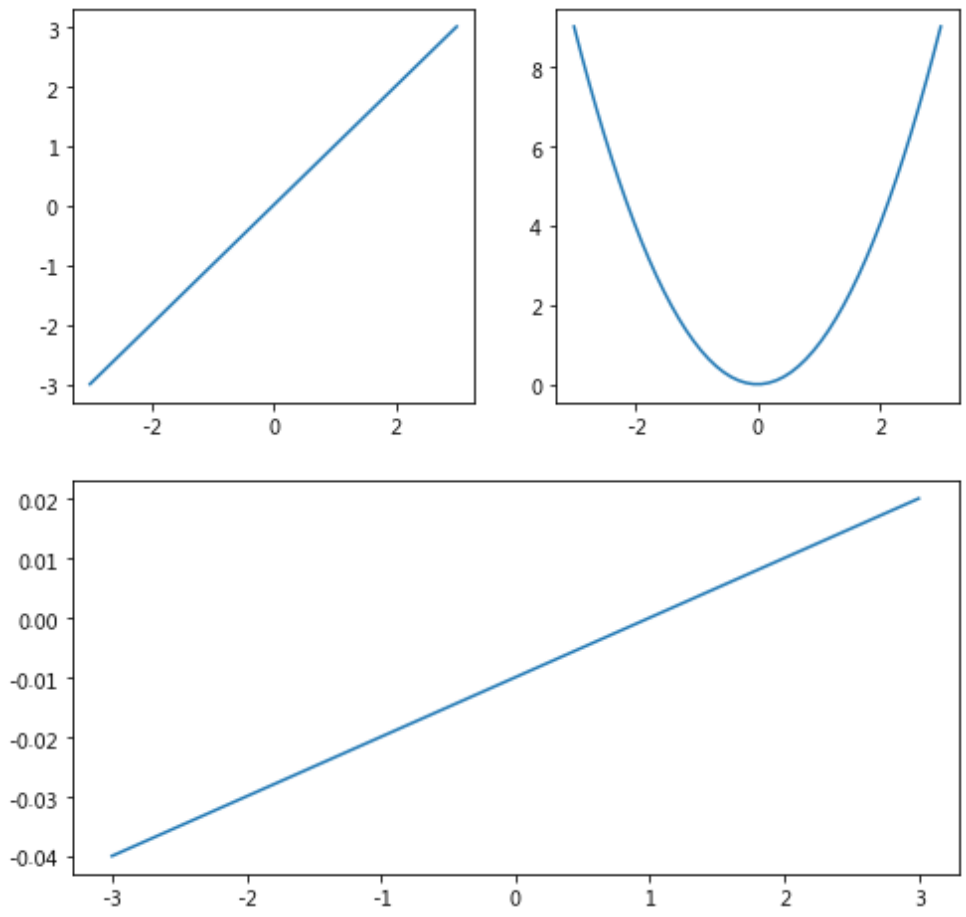
相信通过本章的学习小伙伴们已经能够进行多图合并显示了，下面，请选择以上任意一种方法，画出3个函数：y = x; y = x^2; y = 0.01*x - 0.01

In [41]:

```
#参考答案:
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import numpy as np

x = np.linspace(-3, 3, 50)
y1 = x
y2 = x**2
y3 = 0.01*x - 0.01

plt.figure(figsize=(8, 8))
gs = gridspec.GridSpec(2, 2)

ax1 = plt.subplot(gs[0, 0])
ax2 = plt.subplot(gs[0, 1])
ax3 = plt.subplot(gs[1, :])

ax1.plot(x,y1)
ax2.plot(x,y2)
ax3.plot(x,y3)
```

Out[41]:

```
[<matplotlib.lines.Line2D at 0x7f6180086e48>]
```


3
2
1
0
-1
-2
-3
-2
0
2
8
6
4
2
0
-2
0
2
0.02
0.01
0.00
-0.01
-0.02
-0.03
-0.04
-3
-2
-1
0
1
2
3