

00基本使用

一图胜千言，人是一个视觉敏感的动物，大多数人对数字无法在较短的时间内找到规律和业务意义，可视化就势在必行。

视觉化效应 (Visual effects) 是指人类认知过程中，只要将非视觉性信息转化成视觉信息，可以大大增强海马体的记忆与前额叶皮质的思维反应速度。

Matplotlib 是一个 Python 的 2D 绘图库，通过 Matplotlib，开发者可以仅需要几行代码，便可以生成折线图，直方图，条形图，饼状图，散点图等各种可视化图表。

安装：

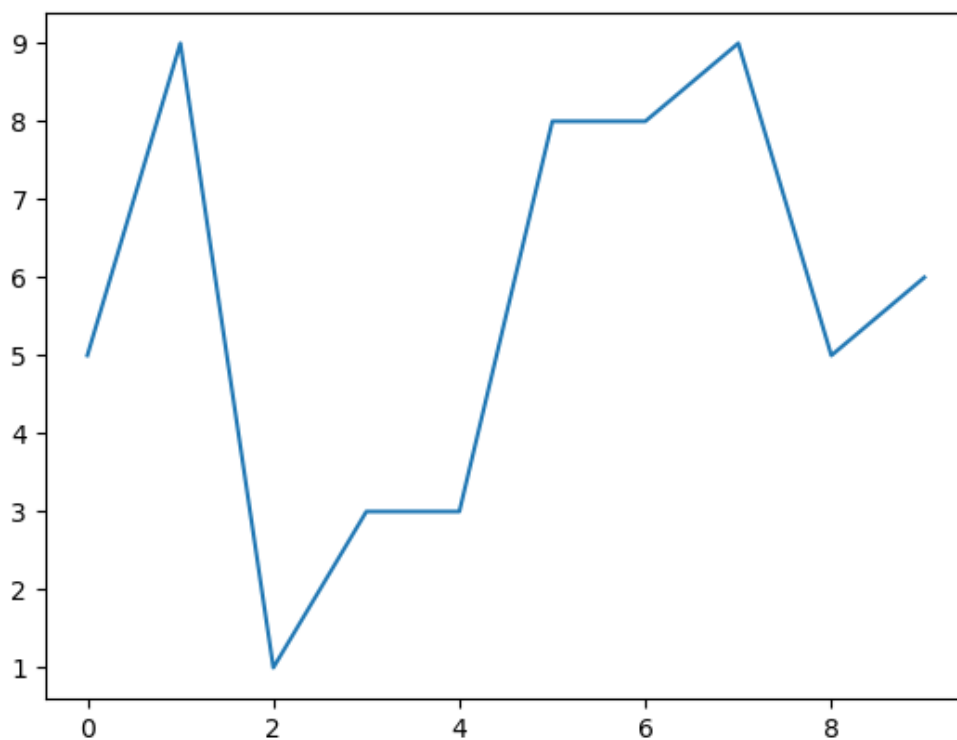
如果是用 Anaconda，可以通过 `conda install matplotlib` 或者通过 `pip install matplotlib` 进行安装。

基本使用：

首先先看以下例子：

```
#首先导入matplotlib库中的pyplot模块，以及numpy库
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10)
y = np.random.randint(0,10,size=(10,))
plt.plot(x,y)
plt.show()
```

那么就会出现以下图：



其中 `plot` 是一个画图的函数，他的参数为 `plot(x,y,"fmt")`。其中 `fmt` 可以传一个字符串，用来给这个图做一些样式修改的。默认的绘制样式是 `"b-"`，也就是蓝色实体线条。比如我想将原来的图的线条改成点状，那么可以通过以下代码实现：

```
plt.plot(x,y,":")  
plt.show()
```

其中使用 `:` 代表点线，是 `matplotlib` 的一个缩写。这些缩写还有以下的：

字符	类型	字符	类型
'-'	实线	'--'	虚线
'-.'	虚点线	'.'	点线
'.'	点	'.'	像素点
'o'	圆点	'v'	下三角点
'^'	上三角点	'<'	左三角点
'>'	右三角点	'1'	下三叉点
'2'	上三叉点	'3'	左三叉点
'4'	右三叉点	's'	正方点
'p'	五角点	'*'	星形点
'h'	六边形点1	'H'	六边形点2
'+'	加号点	'x'	乘号点
'D'	实心菱形点	'd'	瘦菱形点
'_'	横线点		

除了设置线条的形状外，我们还可以设置点的颜色。示例代码如下：

```
plt.plot(x,y,'r') #名称缩写，将颜色线条设置成红色
plt.plot(x,y,color='red') #名称，将颜色设置成红色
plt.plot(x,y,color='#000000') #十六进制，将颜色设置成纯黑色
plt.plot(x,y,color=(0,0,0,0)) #rgba，将颜色设置成纯黑色
```

给线条设置颜色总体来说有三种方式，第一种是使用颜色名称（`r` 是 `red` 的缩写）的形式，第二种是使用十六进制的方式，第三种是使用 `RGB` 或 `RGBA` 的方式。如果使用的是颜色名称，那么可以和线的形状写在同一个字符串中。比如使用红色的五角点，那么可以使用如下的方式实现：

```
plt.plot(x,y,'rp') #将颜色线条设置成红色
```

其中可以表示颜色的缩写字符有如下：

字符	颜色
'b'	蓝色, blue
'g'	绿色, green
'r'	红色, red
'c'	青色, cyan
'm'	品红, magenta
'y'	黄色, yellow
'k'	黑色, black
'w'	白色, white

设置线条样式：

使用plot方法：plot方法就是用来绘制线条的，因此可以在绘制的时候就把线条相关的样式通过参数传进去。示例代码如下：

```
plt.plot(x,y,linewidth=2) #设置线的宽度
plt.plot(x,y,alpha=0.2) #设置透明度
```

设置图的信息：

现在我们添加图后，没有指定x轴代表什么，y轴代表什么，以及这个图的标题是什么。因此以下我们通过一些属性来设置一下。

设置轴和标题：

1. 设置轴名称：可以通过 `plt.xlabel` 和 `plt.ylabel` 来设置 x 轴和 y 轴的名称。示例代码如下：

```
plt.plot(x,y,linewidth=10,color='red')
plt.xlabel("x轴")
plt.ylabel("y轴")
```

默认情况下是显示不了中文的。需要设置字体。可以通过添加以下代码来实现：

```
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.plot(x,y,linewidth=10,color='red')
plt.xlabel("x轴")
plt.ylabel("y轴")
```

2. 设置标题：可以通过 `plt.title` 方法来实现。示例代码如下：

```
plt.title("hello world")
```

3. 设置 x 轴和 y 轴的坐标轴范围：使用 `plt.xlim` 设置x坐标轴范围：(0, 12)；使用 `plt.ylim` 设置y坐标轴范围：(0, 13)；

```
plt.xlim((0, 12))
plt.ylim((0, 13))
```

4. 设置 x 轴和 y 轴的刻度：之前我们画的图，x 轴和 y 轴的刻度都是 matplotlib 自动生成的。如果想要在生成图的时候手动的指定，那么可以通过 plt.xticks 和 plt.yticks 来实现：

```
plt.xticks(range(0,20,2)) #在x轴上的刻度是0,2,4,6...18
```

以上会把那个刻度显示在 x 轴上。如果想要显示字符串类型，那么可以再构造一个数组，这个数组的长度必须和 x 轴刻度的长度保持一致。然后传给 xticks 的第二个参数。示例代码如下：

```
x = range(0,20,2)
xticks = ["%d坐标"%i for i in x]
plt.xticks(x,xticks) #在x轴上的刻度是0坐标,2坐标...18坐标
```

同样 y 轴的刻度设置也是一样的。

figure图像

matplotlib 的 figure 就是一个单独的 figure 小窗口,小窗口里面还可以有更多的小图片.如果想要调整图片的大小和像素,可以通过 plt.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, frameon=True) 来实现。

其中 num 是图的编号, figsize 的单位是英寸, dpi 是每英寸的像素点, facecolor 是图片背景颜色, edgecolor 是边框颜色, frameon 代表是否绘制画板。

使用 np.linspace 定义 x: 范围是(-3,3);个数是50. 仿真一维数据组(x, y1)表示曲线1. 仿真一维数据组(x, y2)表示曲线2.

```
x = np.linspace(-3, 3, 50)
y1 = 2*x + 1
y2 = x**2
```

使用 plt.figure 定义一个图像窗口. 使用 plt.plot 画(x, y1)曲线.

```
plt.figure()
plt.plot(x, y1)
plt.show()
```

使用 plt.figure 定义一个图像窗口: 编号为3; 大小为(8, 5),dpi为80, 背景颜色是黄色. 使用 plt.plot 画(x, y2)曲线. 使用 plt.plot 画(x, y1)曲线, 曲线的颜色属性(color)为红色;曲线的宽度(linewidth)为1.0; 曲线的类型(linestyle)为虚线. 使用 plt.show 显示图像.

```
plt.figure(num=3, figsize=(8, 5),dpi=80,facecolor='y')
plt.plot(x, y2)
plt.plot(x, y1, color='red', linewidth=1.0, linestyle='--')
plt.show()
```

使用 plt.xlim 设置x坐标轴范围: (-1, 2); 使用 plt.ylim 设置y坐标轴范围: (-2, 3);

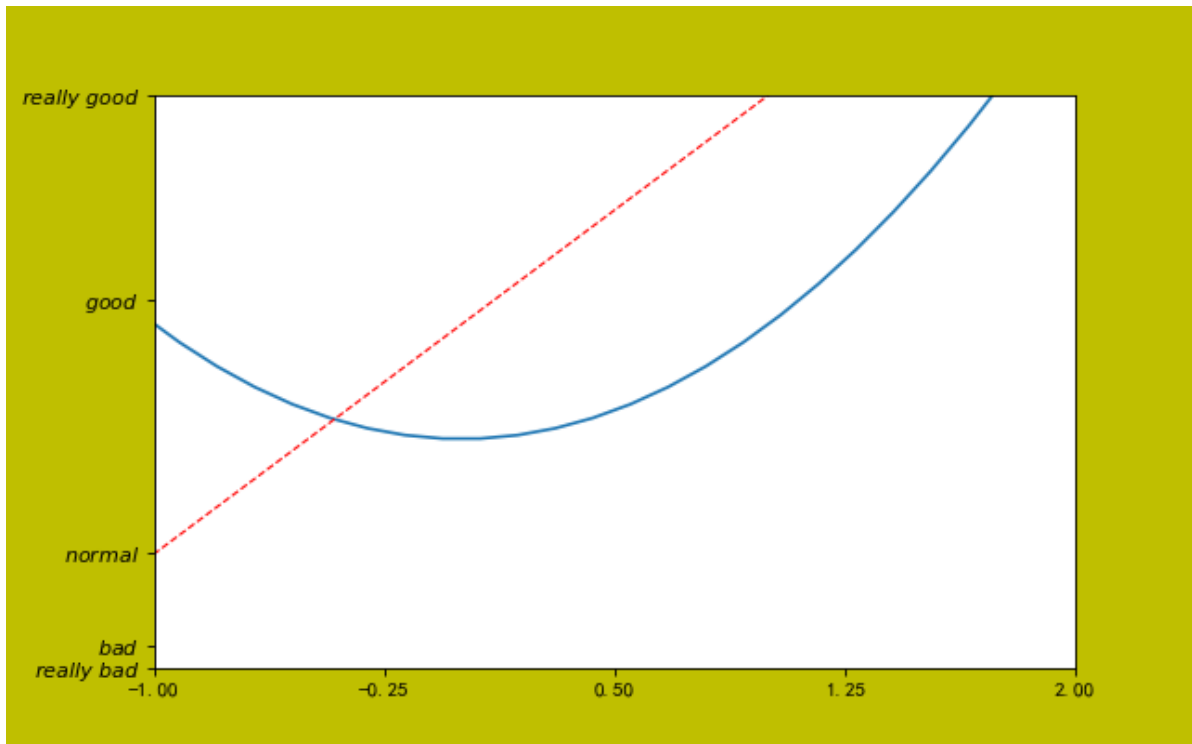
```
plt.xlim((-1, 2))
plt.ylim((-2, 3))
```

使用 `np.linspace` 定义范围以及个数：范围是(-1,2);个数是5,使用 `plt.xticks` 设置x轴刻度：范围是(-1,2);个数是5.

```
new_ticks = np.linspace(-1, 2, 5)
plt.xticks(new_ticks)
```

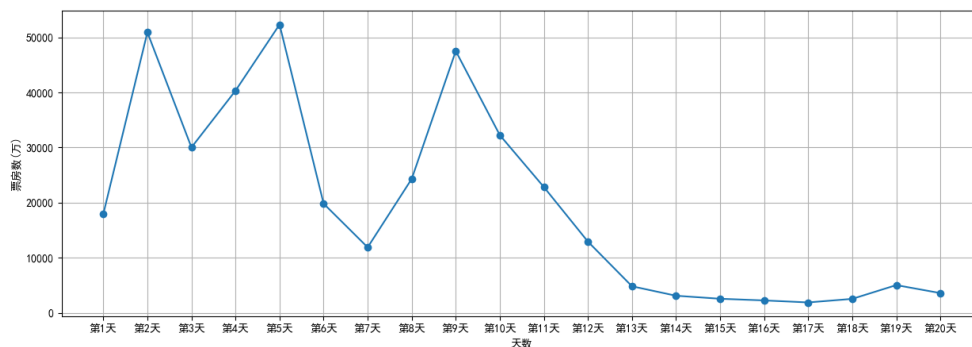
使用 `plt.yticks` 设置y轴刻度以及名称：刻度为[-2, -1.8, -1, 1.22, 3]; 对应刻度的名称为['really bad','bad','normal','good', 'really good']. 使用 `plt.show` 显示图像.

```
plt.yticks([-2, -1.8, -1, 1.22, 3],[r'$really\ bad$', r'$bad$', r'$normal$',
r'$good$', r'$really\ good$'])
plt.show()
```



电影票房案例

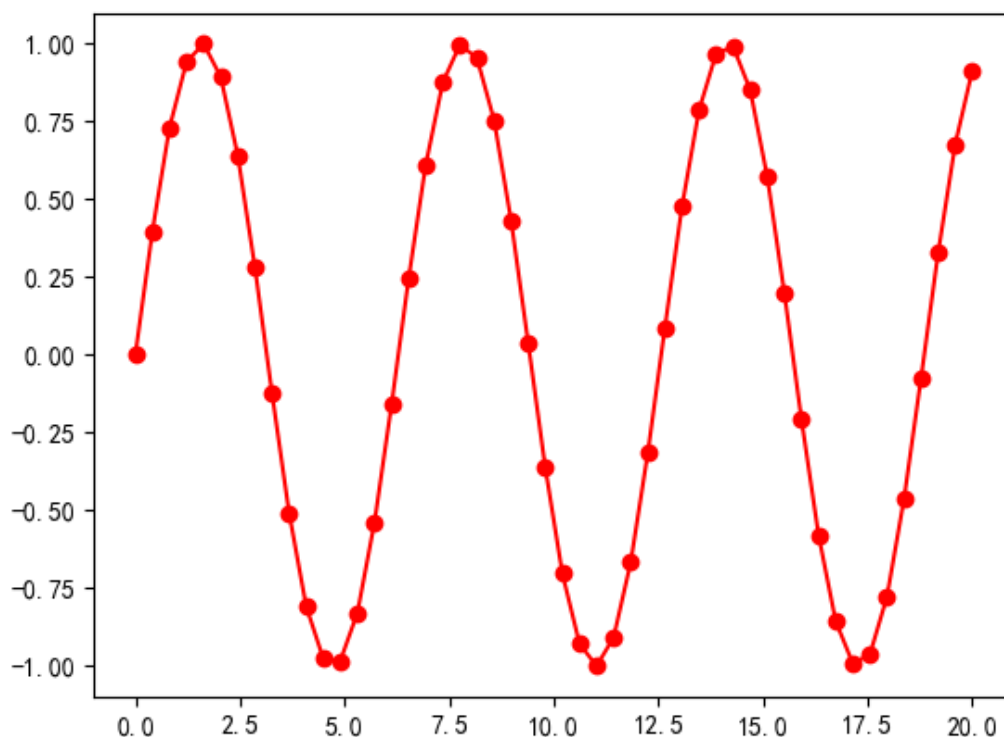
```
# 完整代码
avenger =
[17974.4, 50918.4, 30033.0, 40329.1, 52330.2, 19833.3, 11902.0, 24322.6, 47521.8, 32262.0,
22841.9, 12938.7, 4835.1, 3118.1, 2570.9, 2267.9, 1902.8, 2548.9, 5046.6, 3600.8]
plt.figure(figsize=(15, 5))
plt.plot(avenger, marker="o") #设置标记点
plt.xticks(range(20), ["第%d天"%x for x in range(1, 21)])
plt.xlabel("天数")
plt.ylabel("票房数(万)")
plt.grid() #显示网格
plt.show()
```



设置marker:

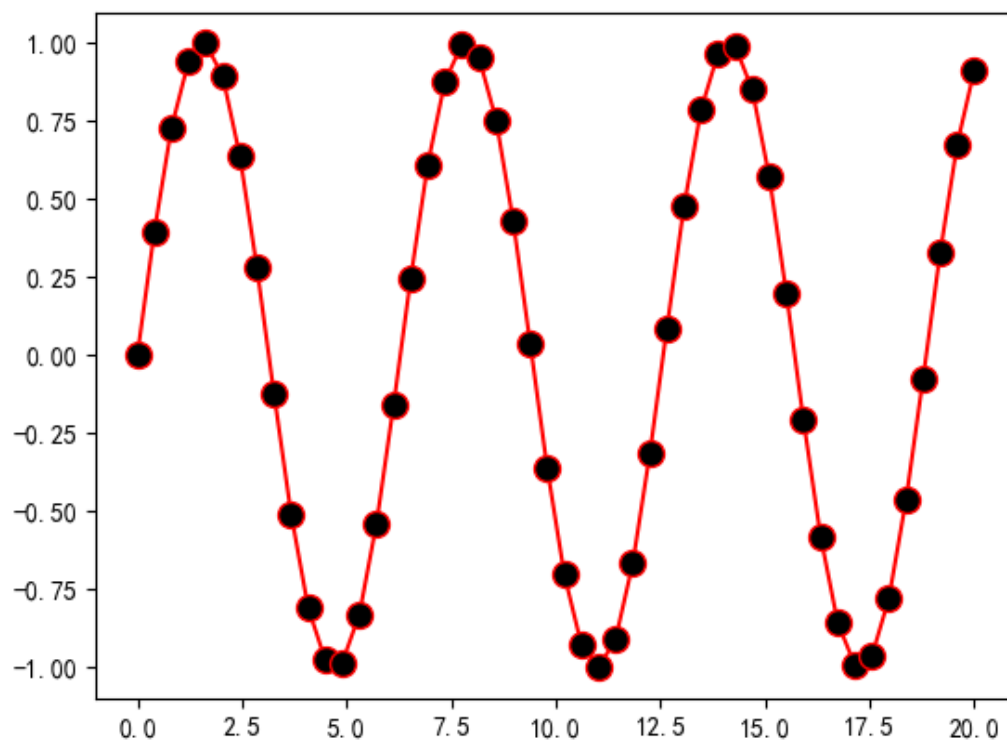
有时候，我们想要在一些关键点上重点标记出来。那么我们可以通过设置 `marker` 来实现。示例代码如下：

```
x = np.linspace(0,20)
y = np.sin(x)
plt.plot(x,y,color='r',marker="o")
```



我们设置了 `marker` 为 `o`，这样就是会在 (x,y) 的坐标点上显示出来，并且显示的是圆点。其中 `o` 跟之前的线条样式的简写是一样的。另外，还可以通过 `markerfacecolor` 属性和 `markersize` 来指定标记点的颜色和大小。示例代码如下：

```
# 以下设置标记点的颜色为黑色，尺寸为10
plt.plot(x,y,color='r',marker="o",markerfacecolor='k',markersize=10)
```



保存图片：

可以调用 `plt.savefig(path)` 来保存当前的图片。示例代码如下：

```
plt.savefig("./abc.png")
```

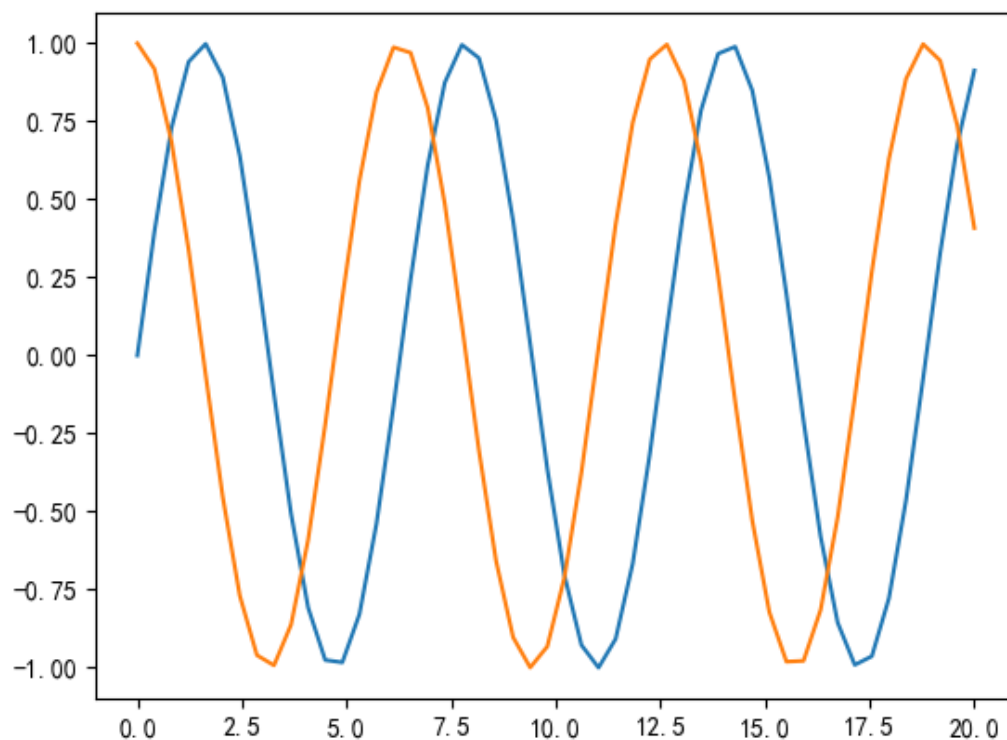
绘制多个图

绘制多个图有两种形式，第一种形式是在一张图中绘制多跟线条，第二种形式是绘制多个子图形。以下分别进行讲解。

绘制多根折线：

绘制多根线条，只要使用多次 `plt.plot` 绘制即可。示例代码如下：

```
x = np.linspace(0,20)
plt.plot(x,np.sin(x))
plt.plot(x,np.cos(x))
```

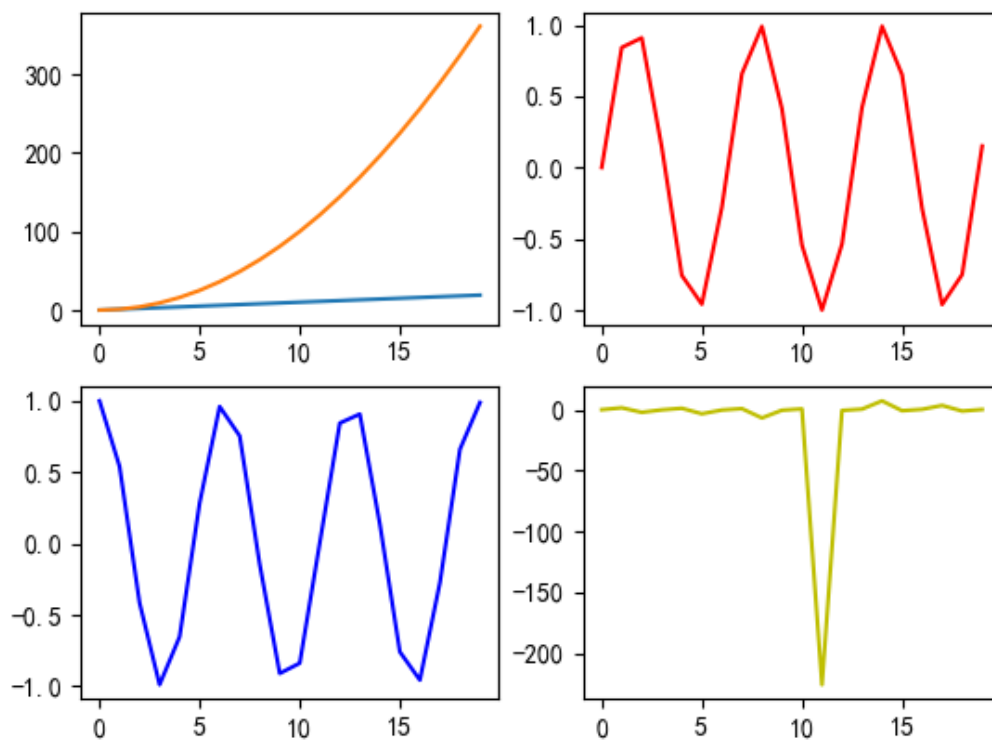
绘制多个子图：

绘制子图的时候，我们可以使用 `plt.subplot` 或 `plt.subplots` 来实现。

`plt.subplot`示例：

```
values = np.arange(20)
plt.subplot(221)
plt.plot(values)
plt.plot(values**2)
plt.subplot(222)
plt.plot(np.sin(values), 'r')
plt.subplot(223)
plt.plot(np.cos(values), 'b')
plt.subplot(224)
plt.plot(np.tan(values), 'y')
```

效果图如下：



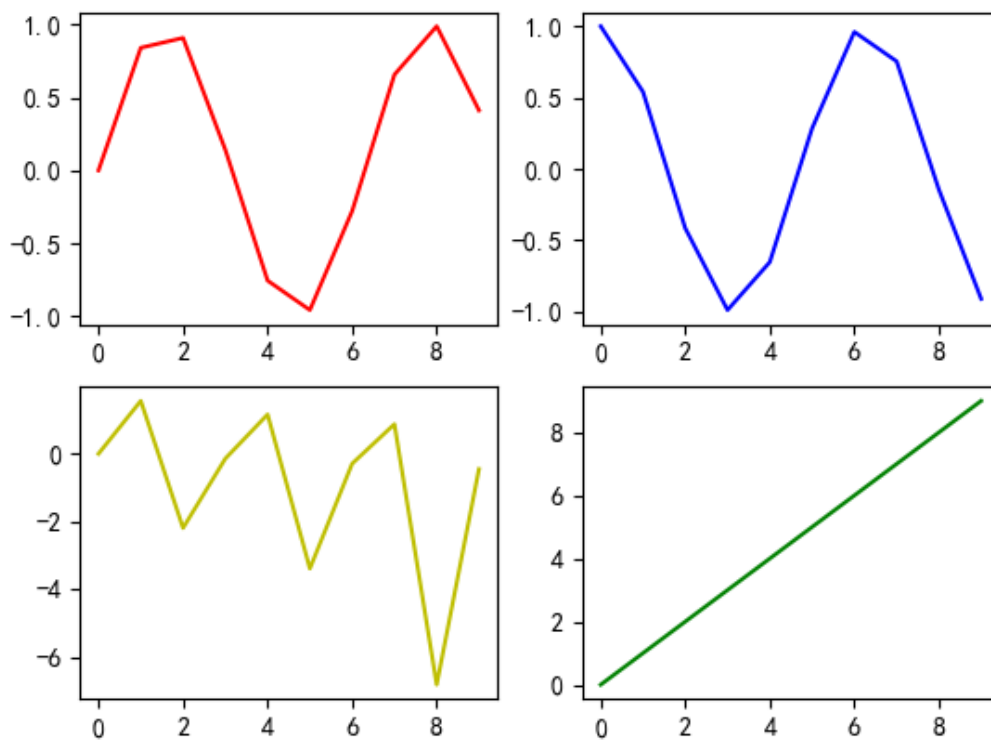
其中 `subplot` 中的 221 和 222 分别代表的意思是，第一个数表示这个大图中总共有 2 行，第二个数表示总共有 2 列，然后第三个数表示当前绘制第几个图。

`plt.subplots`

也可以使用 `figure, axes=plt.subplots(rows, cols)` 来绘制多个图形，返回值是一个元组, 示例代码如下：

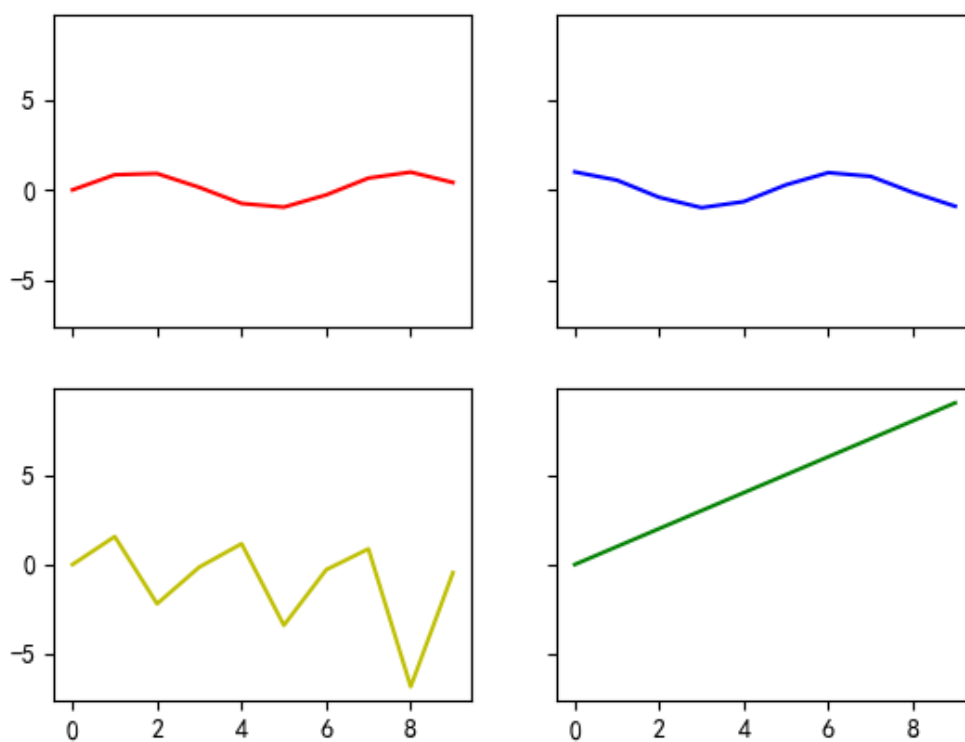
```
figure, axes = plt.subplots(2, 2)
axes[0, 0].plot(np.sin(np.arange(10)), c='r')
axes[0, 1].plot(np.cos(np.arange(10)), c='b')
axes[1, 0].plot(np.tan(np.arange(10)), c='y')
axes[1, 1].plot(np.arange(10), c='g')
```

效果图如下：



另外使用 `subplot` 和 `subplots` 都可以传递 `sharex/sharey` 参数，这两个参数表示是否需要共享X轴和Y轴。示例代码如下：

```
figure, axes = plt.subplots(2,2,sharex=True,sharey=True)
```



风格设置

matplotlib 图片默认内置了几种风格。我们可以通过 `plt.style.available` 来查看内置的所有风格:

```
['solarize_light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background',  
'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright',  
'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-  
darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper',  
'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-  
white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

可以使用 `plt.style.use` 方法来使用不同的风格。示例代码如下:

```
plt.style.use("dark_background")
```

matplotlib 内置的样式: https://tonysyu.github.io/raw_content/matplotlib-style-gallery/gallery.html