```
# 引入Matplotlib引入
from matplotlib import pyplot as plt
import numpy as np
```

基本配置设置

```
# 设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']
# 中文负号
plt.rcParams['axes.unicode_minus'] = False

# 设置分别率 为100
plt.rcParams['figure.dpi'] = 100

# 设置大小
plt.rcParams['figure.figsize'] = (5,3)
```

```

```

一.创建图形对象

在 Matplotlib 中，面向对象编程的核心思想是创建图形对象（figure object）。通过图形对象来调用其它的方法和属性，这样有助于我们更好地处理多个画布。在这个过程中，pyplot 负责生成图形对象，并通过该对象来添加一个或多个 axes 对象（即绘图区域）。

Matplotlib 提供了`matplotlib.figure`图形类模块，它包含了创建图形对象的方法。通过调用 pyplot 模块中 figure() 函数来实例化 figure 对象。

创建图形对象

figure方法如下：

```
plt.figure(
    num=None,----------------------------------> 图像编号或名称，数字为编号 ，字符串为名称
    figsize=None,------------------------------> 指定figure的宽和高，单位为英寸；
    dpi=None,----------------------------------> 定绘图对象的分辨率，即每英寸多少个像素，缺省值为72
    facecolor=None,----------------------------> 背景颜色
    edgecolor=None, ----------------------------> 边框颜色
    frameon=True, ------------------------------> 是否显示边框
    **kwargs,
)
```

```python
from matplotlib import pyplot as plt
# 创建图形对象
fig = plt.figure()
```

```python
# 之前通过配置更改图形的分辨率和宽高. 如今可以再创建图像对象是创建

fig = plt.figure('f1',figsize=(6,4),dpi=30)
plt.plot()
```

```python
fig = plt.figure('f1',figsize=(4,2),dpi=100)
plt.plot()
```

```python
x = np.arange(0,50)
y = x ** 2
# 创建图形对象, 图形对象的分辨率为100,背景颜色为:灰色
fig = plt.figure('f1',figsize=(4,2), dpi=100,facecolor='gray')
plt.plot(x,y)
```

二. 绘制多子图

figure是绘制对象(可理解为一个空白的画布)，一个figure对象可以包含多个Axes子图，一个Axes是一个绘图区域，不加设置时，Axes为1，且每次绘图其实都是在figure上的Axes上绘图。

接下来将学习绘制子图的几种方式:

- add_axes() : 添加区域
- subplot() : 均等地划分画布,只是创建一个包含子图区域的画布,(返回区域对象)

- subplots() : 既创建了一个包含子图区域的画布，又创建了一个 figure 图形对象.(返回图形对象和区域对象)

1.add_axes() : 添加区域

Matplotlib 定义了一个 axes 类（轴域类），该类的对象被称为 axes 对象（即轴域对象），它指定了一个有数值范围限制的绘图区域。在一个给定的画布（figure）中可以包含多个 axes 对象，但是同一个 axes 对象只能在一个画布中使用。

> 2D 绘图区域（axes）包含两个轴（axis）对象

语法:

add_axes(rect)

- 该方法用来生成一个 axes 轴域对象，对象的位置由参数rect决定
- rect 是位置参数，接受一个由 4 个元素组成的浮点数列表，形如 [left, bottom, width, height] ，它表示添加到画布中的矩形区域的左下角坐标(x, y)，以及宽度和高度。

如下所示：

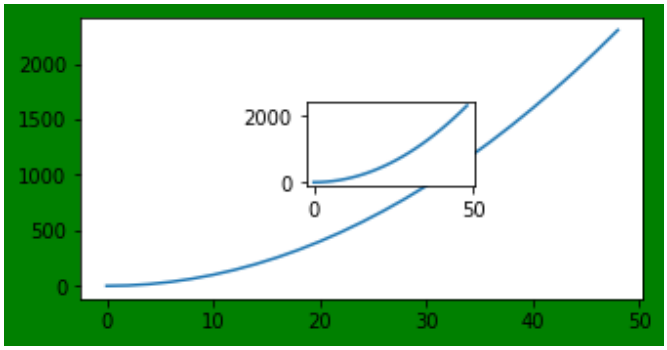
```
fig = plt.figure(figsize=(4,2),facecolor='g')

# ax1从画布起始位置绘制,宽高和画布一致
ax1=fig.add_axes([0,0,1,1])

# ax2 从画布 20% 的位置开始绘制, 宽高是画布的 50%
ax2=fig.add_axes([0.2,0.2,0.5,0.5])
```

> 注意：每个元素的值是画布宽度和高度的分数。即将画布的宽、高作为 1 个单位。比如，[0.2, 0.2, 0.5, 0.5]，它代表着从画布 20% 的位置开始绘制, 宽高是画布的 50%

```
# 创建绘图对象
fig = plt.figure(figsize=(4,2),facecolor='g')

# 创建x坐标
x = np.arange(0,50,2)
# 创建y坐标
y = x ** 2
# 创建区域1,和画布位置一致
ax1 = fig.add_axes([0.0,0.0,1,1])
```

```
10
11  # 区域绘制图形
12  ax1.plot(x,y)
13
14  # 创建区域ax2 从画布 40% 的位置开始绘制, 宽高是画布的 50%
15  ax2=fig.add_axes([0.4,0.4,0.3,0.3])
16
17  # 区域2中绘制图形
18  ax2.plot(x,y)
19
```

```
1  [<matplotlib.lines.Line2D at 0x2378d93cb88>]
```



练习:

修改上面代码,将区域2放到右下角.

```
1
```

```
1
```

区域中基本方法的使用

- 区域图表名称: set_title
- 区域中x轴和y轴名称:set_xlabel() set_ylabel()
- 刻度设置: set_xticks()
- 区域图表图例: legend()

```
1  # 创建绘图对象
2  fig = plt.figure(figsize=(4,2),facecolor='g')
3
4  # 创建x坐标
```

```
x = np.arange(0,50,2)
# 创建y坐标
y = x ** 2
# 创建区域1,和画布位置一致
ax1 = fig.add_axes([0.0,0.0,1,1])
# 设置图表名称
ax1.set_title("axes1")
# x轴名称
ax1.set_xlabel('X axis')

# 设置ax1横轴刻度
ax1.set_xticks(np.arange(0,50,3))

# 区域绘制图形
ax1.plot(x,y,label="ax1")
# 图例
ax1.legend()
# 创建区域ax2 从画布 40% 的位置开始绘制, 宽高是画布的 50%
ax2=fig.add_axes([0.2,0.2,0.4,0.4])
ax2.set_title("axes2")

# 区域2中绘制图形
ax2.plot(x,y,label='ax2')
# 图例,
ax2.legend()

```

练习：

- 1 使用add_axes()方法,在一个图形对象中创建2个区域,分别画一个曲线和一个直线
- 1 分别设置两个区域图形的名称和其他设置

```
1
```

2.subplot() 函数，它可以均等地划分画布

参数格式如下：

ax = `plt.subplot(nrows, ncols, index,*args, **kwargs)`

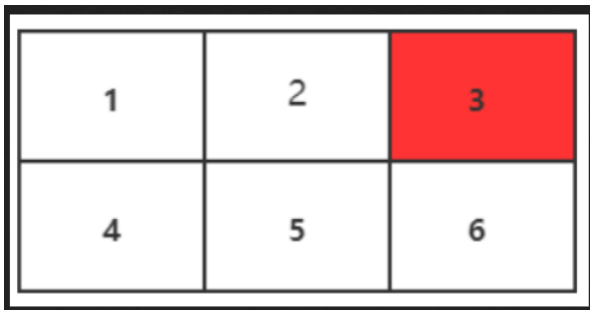
- nrows 行
- ncols 列
- index: 位置
- kwargs: title/xlabel/ylabel 等.....

> 也可以直接将几个值写到一起,如:subplot(233)

返回: 区域对象

nrows 与 ncols 表示要划分几行几列的子区域（nrows*nclos表示子图数量），index 的初始值为1，用来选定具体的某个子区域。

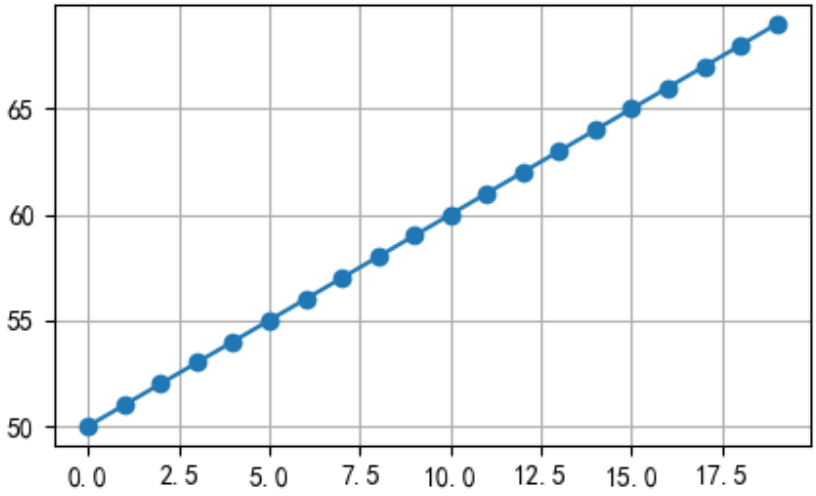
例如： subplot(233)表示在当前画布的右上角创建一个两行三列的绘图区域（如下图所示），同时，选择在第 3 个位置绘制子图。



plot(y) x可省略,默认[0,....,N-1]递增,N为y轴元素的个数.

```
plt.plot(range(50,70),marker="o")
plt.grid()
```

```
20
```




```
# 以上y轴的个数为len(range(50,70)),
# 上面代码y的长度.
y_len = len(range(50,70))  # 长度为20
print([i for i in range(50,70)])
np.linspace(0,y_len-1,y_len)
```

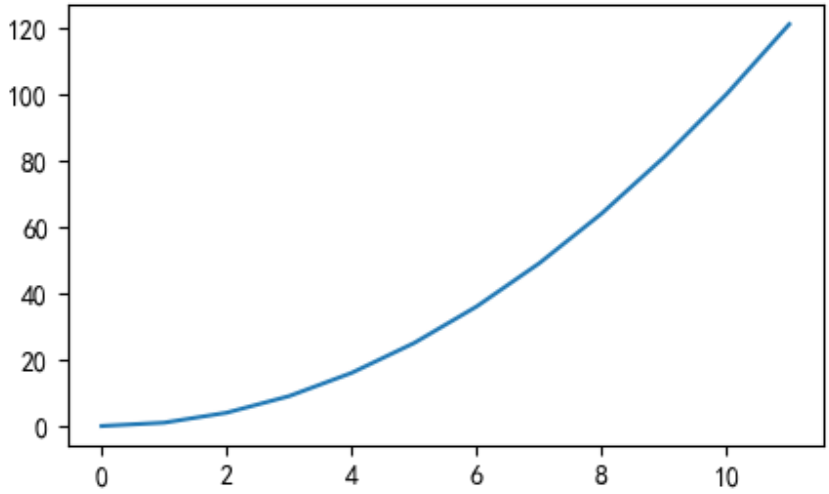
```
[50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
67, 68, 69]
```

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11.,
12.,
       13., 14., 15., 16., 17., 18., 19.])
```

```
print(np.linspace(0,11,12))

plt.plot(np.arange(12)**2)
```

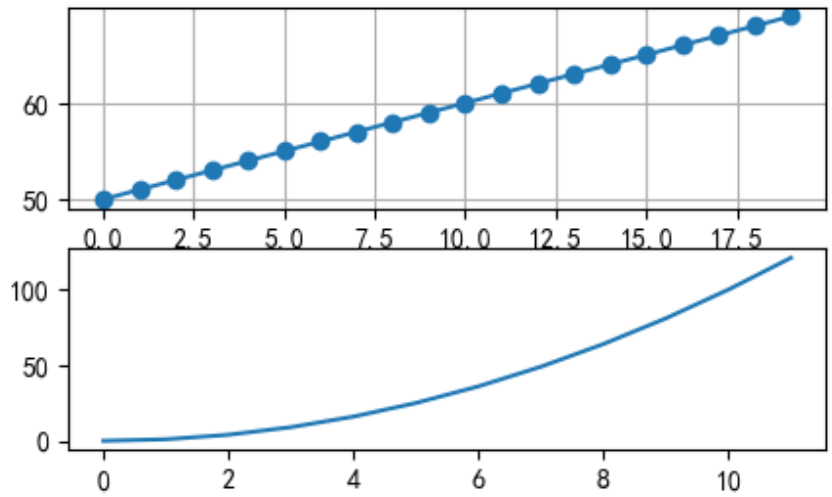
```
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11.]
```

```
[<matplotlib.lines.Line2D at 0x1d694361f88>]
```




```
# 默认画布分割为2行1列,当前所在第一个区域
plt.subplot(211)
# x可省略,默认[0,....,N-1]递增,N为y轴元素的个数.
plt.plot(range(50,70),marker="o")
plt.grid()

# 默认画布分割为2行1列,当前所在第二个区域
plt.subplot(212)

plt.plot(np.arange(12)**2)

```

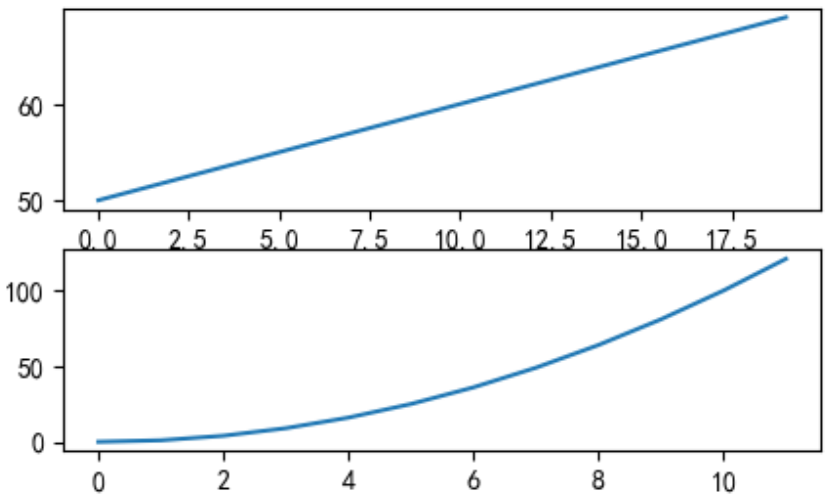
```
[<matplotlib.lines.Line2D at 0x1d694885288>]
```



如果新建的子图与现有的子图重叠，那么重叠部分的子图将会被自动删除，因为它们不可以共享绘图区域。

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

#现在创建一个子图，它表示一个有1行2列的网格的顶部图。
#因为这个子图将与第一个重叠，所以之前创建的图将被删除

plt.subplot(211)
# # x可省略,默认[0,1..,N-1]递增
plt.plot(range(50,70))

plt.subplot(212)

plt.plot(np.arange(12)**2)
```

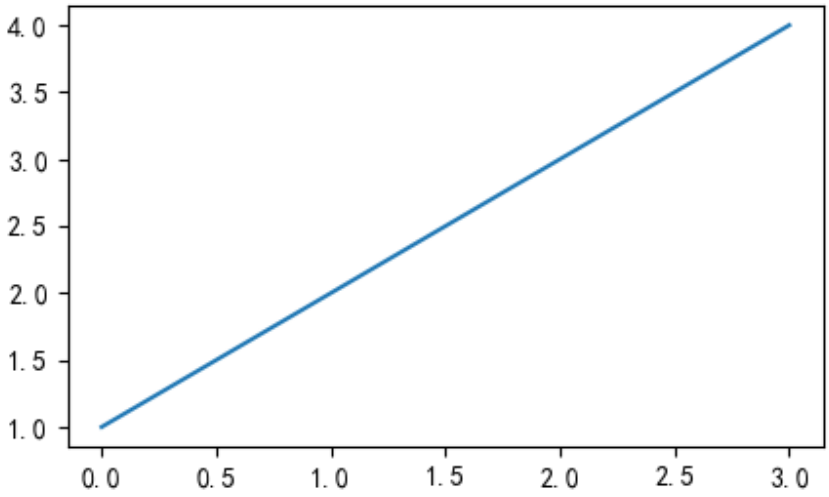
```
[<matplotlib.lines.Line2D at 0x1d694e19c08>]
```

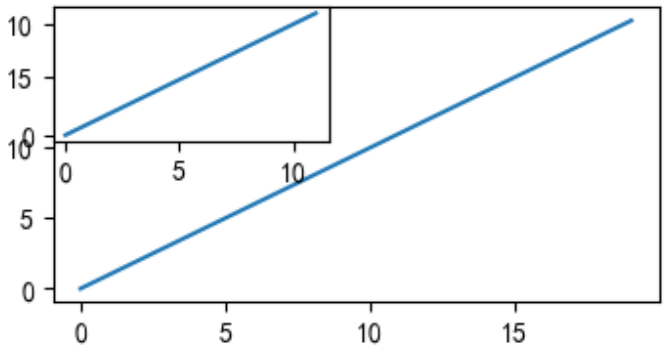


如果不想覆盖之前的图,需要先创建画布,也就是

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

# 还可以先设置画布的大小,再通过画布创建区域
fig = plt.figure(figsize=(4,2))

fig.add_subplot(111) # 分割成1*1 ,当前所在第一个位置

plt.plot(range(20))

fig.add_subplot(221) # 分割成2*2 .当前所在第一个位置

plt.plot(range(12))
```

```
[<matplotlib.lines.Line2D at 0x1d694eb4a08>]
```





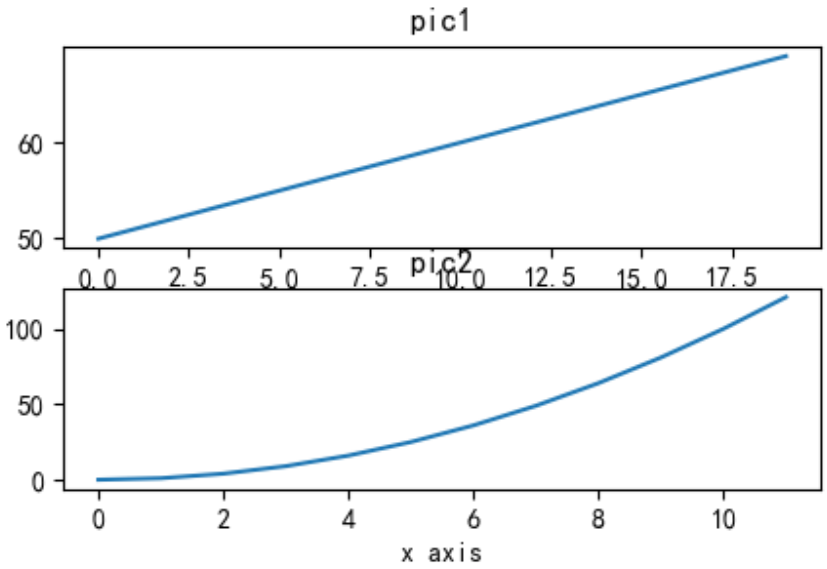
设置多图的基本信息方式:

a. 在创建的时候直接设置:

- 对于subplot关键词赋值参数的了解,可以将光标移动到subplot方法上,使用快捷键shift+tab查看具体内容

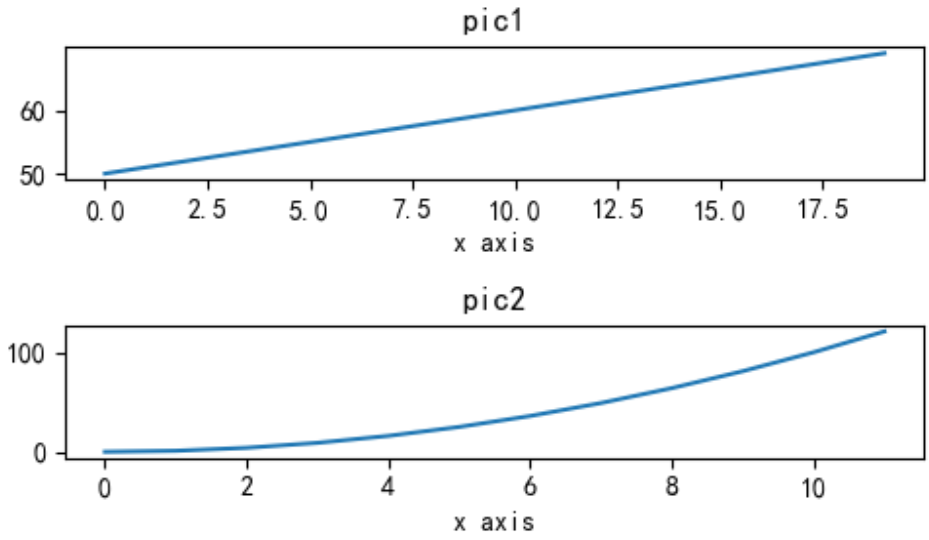
```
#现在创建一个子图，它表示一个有2行1列的网格的顶部图。
plt.subplot(211,title="pic1", xlabel="x axis")
# x可省略,默认[0,1..,N-1]递增
plt.plot(range(50,70))

plt.subplot(212, title="pic2", xlabel="x axis")

plt.plot(np.arange(12)**2)

```

```
[<matplotlib.lines.Line2D at 0x1d694f5bd48>]
```



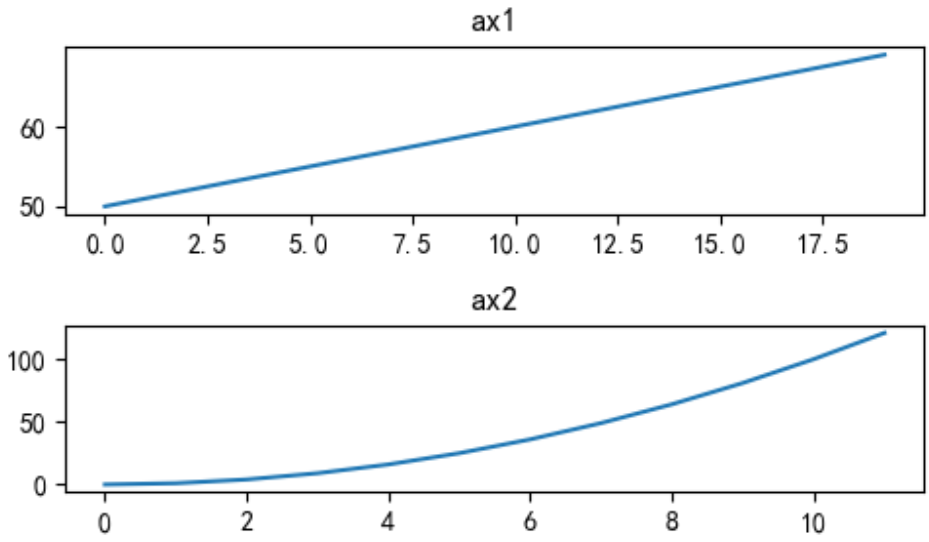
发现子图标题重叠,在最后使用 plt.tight_layout()

```
#现在创建一个子图，它表示一个有2行1列的网格的顶部图。
#--------------- 第一个区域---------------
plt.subplot(211,title="pic1", xlabel="x axis")
# x可省略,默认[0,1..,N-1]递增
plt.plot(range(50,70))

#--------------------第二区域-----------
plt.subplot(212, title="pic2", xlabel="x axis")

plt.plot(np.arange(12)**2)
# 紧凑的布局
plt.tight_layout()
```



b.使用pyplot模块中的方法设置后再绘制

```
#现在创建一个子图，它表示一个有2行1列的网格的顶部图。

#--------------- 第一个区域---------------
plt.subplot(211)
# 使用图形对象:
plt.title("ax1")
# x可省略,默认[0,1..,N-1]递增
plt.plot(range(50,70))

#--------------------第二区域-----------
plt.subplot(212)
plt.title("ax2")

#...其他的自己设置

plt.plot(np.arange(12)**2)
# 紧凑的布局
plt.tight_layout()
```



c.使用返回的区域对象设置.

> 注意区域对象的方法很多都是set_开头

- 使用区域对象将不存在 设置位置

```
#现在创建一个子图，它表示一个有2行1列的网格的顶部图。

###      plt.subplot----返回所在的区域

ax1 = plt.subplot(211)  # 创建第一个区域
ax2 = plt.subplot(212)  # 创建第二个区域

#--------------- 第一个区域 ax1设置---------------
# 使用区域对象:
ax1.set_title("ax1")
# x可省略,默认[0,1..,N-1]递增
ax1.plot(range(50,70))

#--------------------第二区域 ax2设置-----------
ax2.set_title("ax2")
#...其他的自己设置
ax2.plot(np.arange(12)**2)
# 紧凑的布局
plt.tight_layout()
```

3.subplots()函数详解

matplotlib.pyplot模块提供了一个 subplots() 函数，它的使用方法和 subplot() 函数类似。其不同之处在于，subplots() 既创建了一个包含子图区域的画布，又创建了一个 figure 图形对象，而 subplot() 只是创建一个包含子图区域的画布。

subplots 的函数格式如下：

```
fig , ax = plt.subplots(nrows, ncols)
```

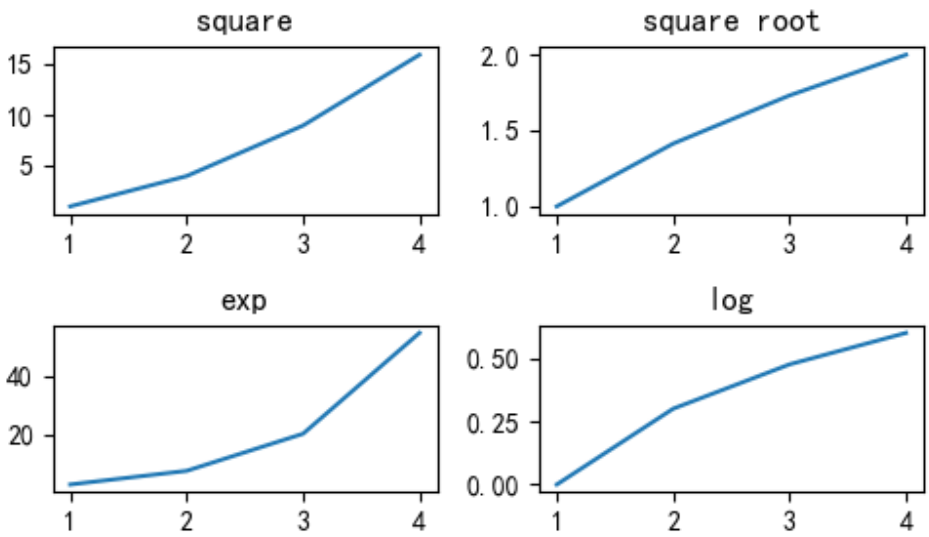
- nrows 与 ncols 表示两个整数参数，它们指定子图所占的行数、列

函数的返回值是一个元组，包括一个图形对象和所有的 axes 对象。其中 axes 对象的数量等于 nrows * ncols，且每个 axes 对象均可通过索引值访问（从0开始）,如下2行2列数据:

下面我们创建了一个 2 行 2 列的子图，并在每个子图中显示 4 个不同的图像。

```
# 引入模块
import matplotlib.pyplot as plt
import numpy as np

# 创建2行2列的子图,返回图形对象(画布),所有子图的坐标轴
fig, axes =  plt.subplots(2,2)

# 第一个区域 ax1
ax1 = axes[0][0]

# x轴
x = np.arange(1,5)

#绘制平方函数
ax1.plot(x, x*x)
ax1.set_title('square')

#绘制平方根图像
axes[0][1].plot(x, np.sqrt(x))
axes[0][1].set_title('square root')

#绘制指数函数
axes[1][0].plot(x, np.exp(x))
axes[1][0].set_title('exp')

#绘制对数函数
axes[1][1].plot(x,np.log10(x))
axes[1][1].set_title('log')
```

```
29
30  # 处理标题遮挡
31  plt.tight_layout()
32
```




```
1
```

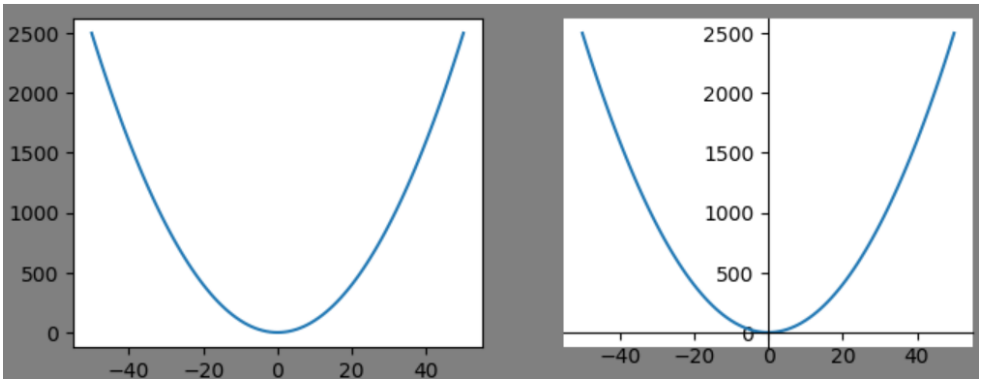
```
1
```

subplots与subplot实例:

x轴为-50到50

y轴为x轴的平方

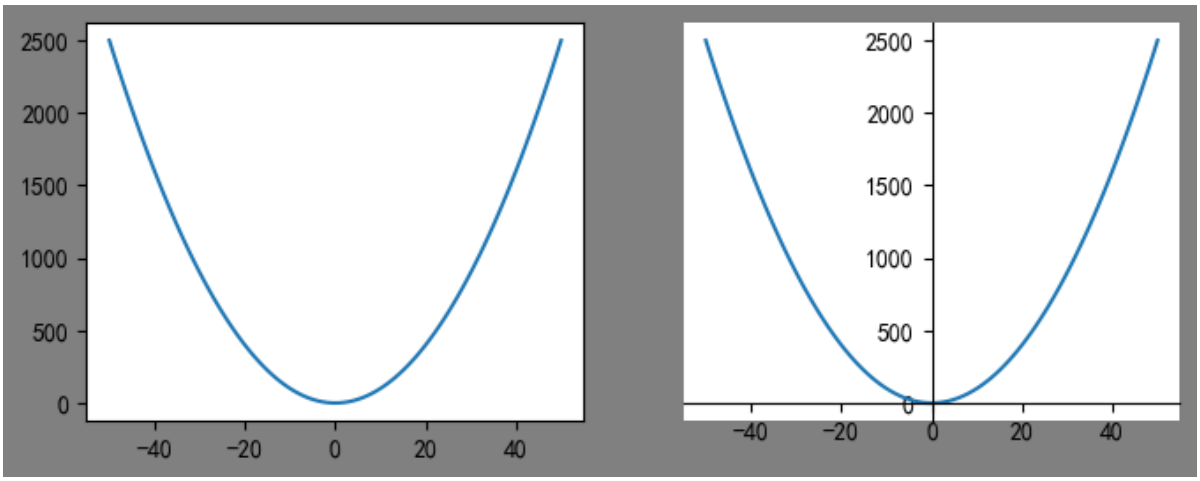
一个画布中绘制2个图.一个是正常图,一个是将纵向坐标向有移动一半距离,效果如下




```
1  fig,axes =  plt.subplots(1,2)
2  # 设置画布的高和宽, 注意:只为英寸 ,默认分别率为72
3  fig.set_figheight(3) # 实际高度为 73*3 像素
```

```
fig.set_figwidth(8) # 实际宽度为 73*8 像素
# 设置背景:
fig.set_facecolor('gray')
# 分别定义x y
x = np.arange(-50,51)
y = x ** 2

#--------------------绘制图形1 -----------
axes[0].plot(x, y)

# ------------处理图形2的绘制----------------
# 不需要右侧和上侧线条,则可以设置他的颜色
axes[1].spines['right'].set_color("none")
axes[1].spines['top'].set_color("none")
# 移动下轴到指定位置
# 在这里，position位置参数有三种，data , outward(向外-可自行尝试) , axes
# axes:0.0 - 1.0之间的值，整个轴上的比例
axes[1].spines['left'].set_position(('axes',0.5))

# 移动下轴到指定位置
# 'data'表示按数值挪动，其后数字代表挪动到Y轴的刻度值
axes[1].spines['bottom'].set_position(('data',0.0))

axes[1].plot(x, y)
```

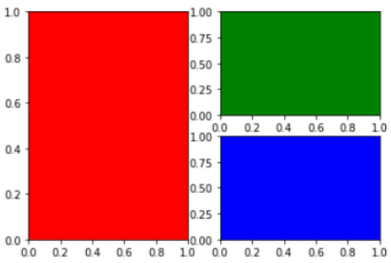
```
[<matplotlib.lines.Line2D at 0x1d6962b1bc8>]
```



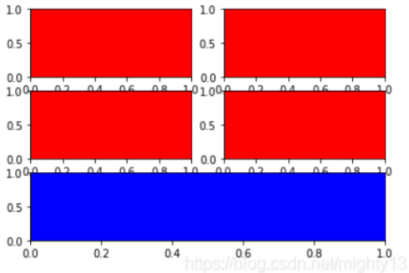
可以看到top和right边被隐藏了

```

```

绘制一个特殊的图形,可以多次调用subplot,如:



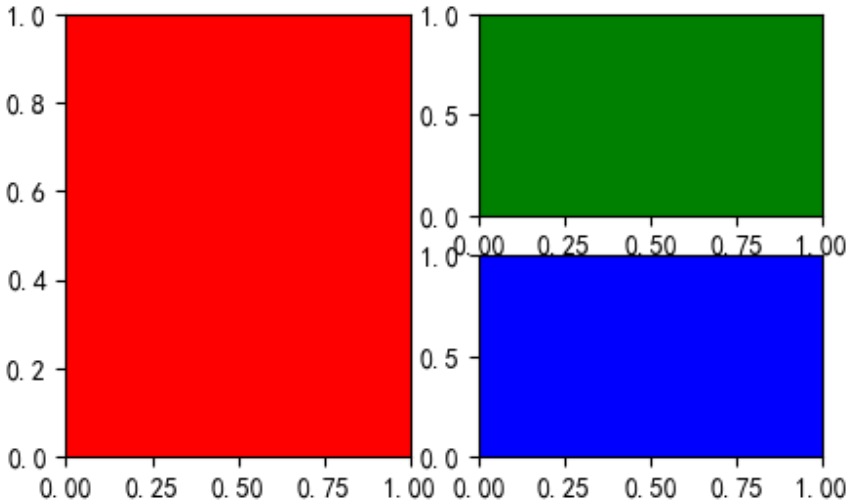
1



2

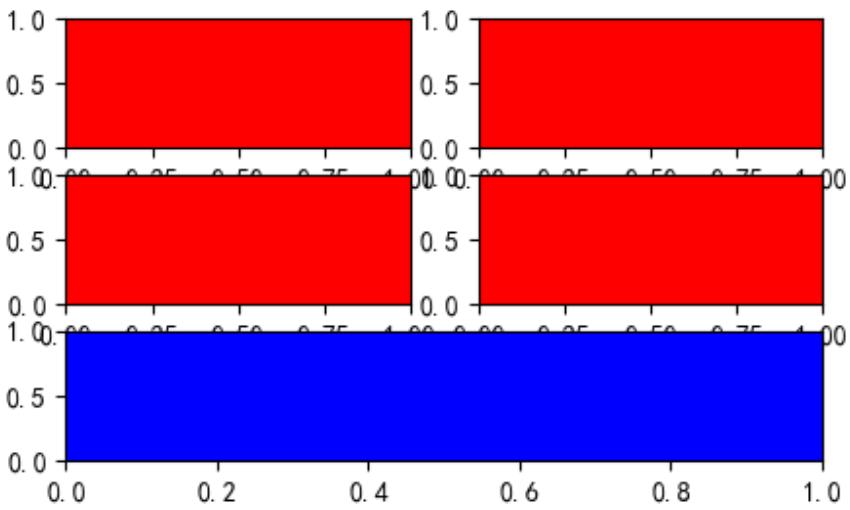
```
# 绘制1行2列子图中的第1个子图
plt.subplot(121,facecolor='r')

# 绘制2行2列子图中的第2个子图
plt.subplot(222,facecolor='g')

# 绘制2行2列子图中的第4个子图
plt.subplot(224,facecolor='b')

```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d694f03308>
```



```python
# 绘制3行2列子图中的第1个子图
plt.subplot(321,facecolor='r')
# 绘制3行2列子图中的第2个子图
plt.subplot(322,facecolor='r')
# 绘制3行2列子图中的第3个子图
plt.subplot(323,facecolor='r')
# 绘制3行2列子图中的第4个子图
plt.subplot(324,facecolor='r')
# # 绘制3行1列子图中的第3个子图
plt.subplot(313,facecolor='b')

```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d694fa0548>
```



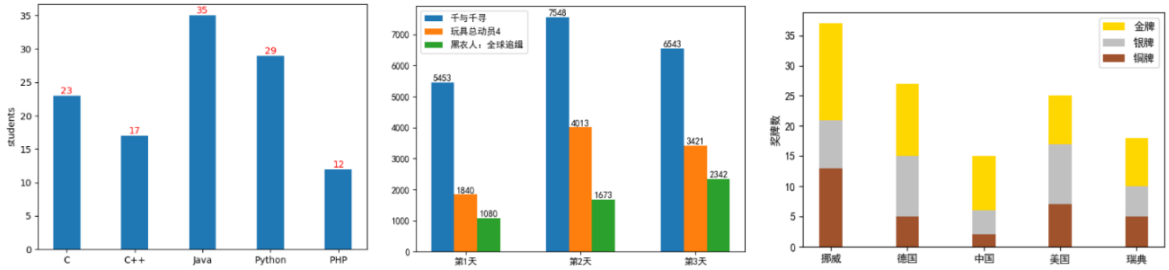

```python

```

```python

```

柱状图:

- 柱状图是一种用矩形柱来表示数据分类的图表。
- 柱状图可以垂直绘制，也可以水平绘制。
- 它的高度与其所表示的数值成正比关系。

- 柱状图显示了不同类别之间的比较关系，图表的水平轴 X 指定被比较的类别，垂直轴 Y 则表示具体的类别值



柱状图的绘制

```
matplotlib.pyplot.bar(x, height, width: float = 0.8, bottom = None, *, align: str = 'center', data = None, **kwargs)
```

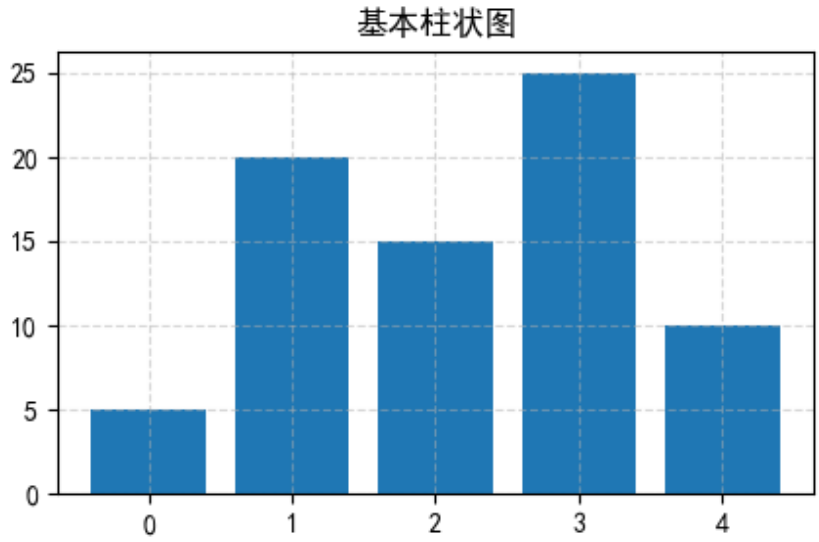
- x 表示x坐标，数据类型为float类型，一般为np.arange()生成的固定步长列表
- height 表示柱状图的高度，也就是y坐标值，数据类型为float类型，一般为一个列表，包含生成柱状图的所有y值
- width 表示柱状图的宽度，取值在0~1之间，默认值为0.8
- bottom 柱状图的起始位置，也就是y轴的起始坐标，默认值为None
- align 柱状图的中心位置，“center”，"lege"边缘，默认值为’center’
- color 柱状图颜色，默认为蓝色
- alpha 透明度，取值在0~1之间，默认值为1
- label 标签，设置后需要调用plt.legend()生成
- edgecolor 边框颜色 (ec)
- linewidth 边框宽度，浮点数或类数组，默认为None (lw)
- tick_label：柱子的刻度标签，字符串或字符串列表，默认值为None。
- linestyle :线条样式 (ls)

1. 基本的柱状图

```
1  import matplotlib.pyplot as plt
2
3  # x轴数据
```

```
x = range(5)

# y轴数据
data = [5, 20, 15, 25, 10]

# 设置图形标题
plt.title("基本柱状图")

# 绘制网格
plt.grid(ls="--", alpha=0.5)

# bar绘制图形,x 表示x坐标  data为表示柱状图的高度
plt.bar(x, data)
```

```
<BarContainer object of 5 artists>
```



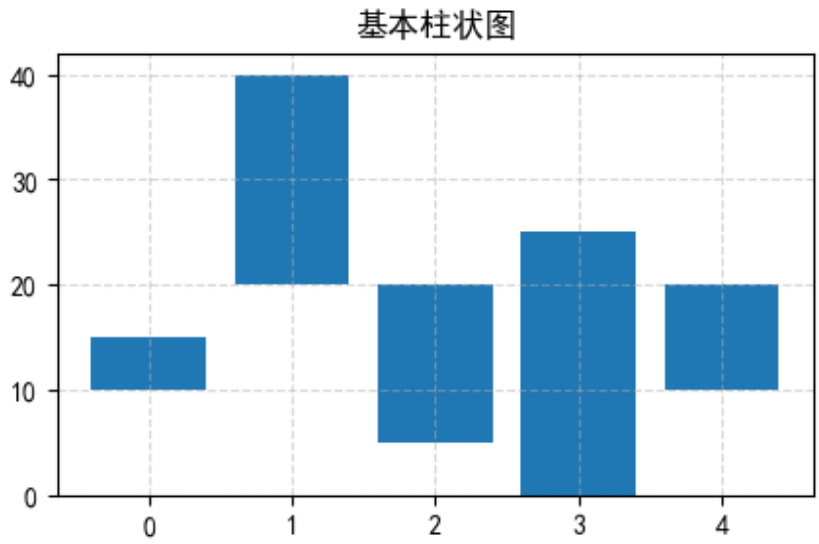
- bottom参数

柱状图的起始位置，也就是y轴的起始坐标，默认值为None

```
import matplotlib.pyplot as plt

# x轴数据
x = range(5)

# y轴数据
data = [5, 20, 15, 25, 10]

# 设置图形标题:
plt.title("基本柱状图")

```

```
# 绘制网格
plt.grid(ls="--", alpha=0.5)

# bar绘制图形,x 表示x坐标  data为表示柱状图的高度
plt.bar(x, data, bottom=[10, 20, 5, 0, 10])

# 注意data = [5, 20, 15, 25, 10] ----对应的bottom——>[10, 20, 5, 0, 10]
'''
--- a.形状要一致
--- b.每个图形y轴的起始位置
'''
```

```
'\n--- a.形状要一致\n--- b.每个图形y轴的起始位置\n'
```

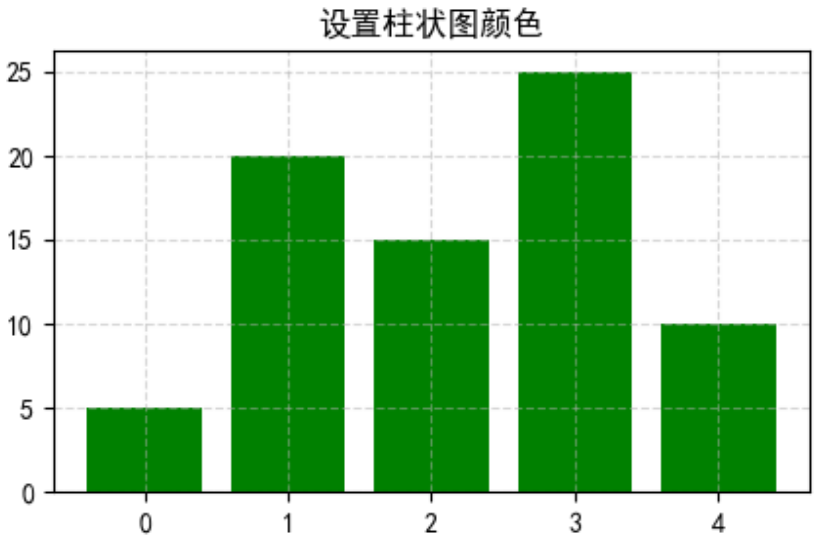


- 柱状图颜色

```
import matplotlib.pyplot as plt

# x轴数据
x = range(5)

# y轴数据
data = [5, 20, 15, 25, 10]

# 设置图形标题:
plt.title("设置柱状图颜色")

# 绘制网格
plt.grid(ls="--", alpha=0.5)
```

```
14
15  # bar绘制图形,x 表示x坐标  data为表示柱状图的高度
16  plt.bar(x, data ,facecolor="green")
17  #plt.bar(x, data ,color="green")
```

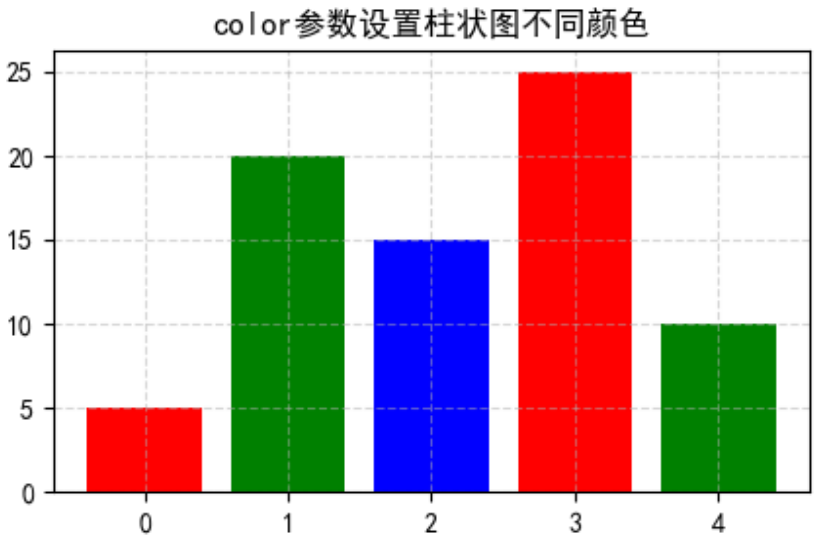
```
1  <BarContainer object of 5 artists>
```




```
1  import matplotlib.pyplot as plt
2
3  # x轴数据
4  x = range(5)
5
6  # y轴数据
7  data = [5, 20, 15, 25, 10]
8
9  # 设置图形标题:
10 plt.title("color参数设置柱状图不同颜色")
11
12 # 绘制网格
13 plt.grid(ls="--", alpha=0.5)
14
15 # bar绘制图形,x 表示x坐标  data为表示柱状图的高度
16 #plt.bar(x, data ,color=['r', 'g', 'b'])
17 '''
18 facecolor和color设置单个颜色时使用方式一样
19 color可以设置多个颜色值,facecolor不可以
20 '''
21 plt.bar(x, data ,color=['r', 'g', 'b'])
```

```
22
23
```

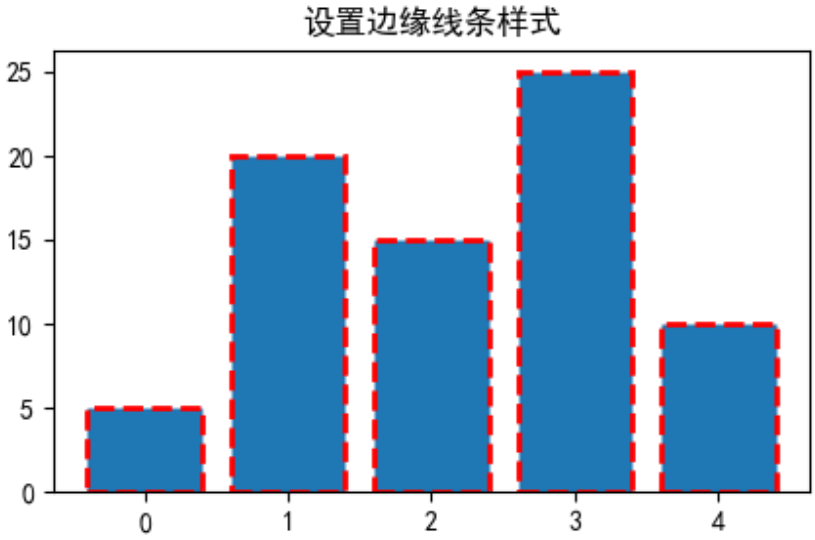
```
<BarContainer object of 5 artists>
```



- 描边
  -相关的关键字参数为：
- edgecolor 或 ec
- linestyle 或 ls
- linewidth 或 lw

```
import matplotlib.pyplot as plt

data = [5, 20, 15, 25, 10]

plt.title("设置边缘线条样式")

plt.bar(range(len(data)), data, ec='r', ls='--', lw=2)

```

```
<BarContainer object of 5 artists>
```



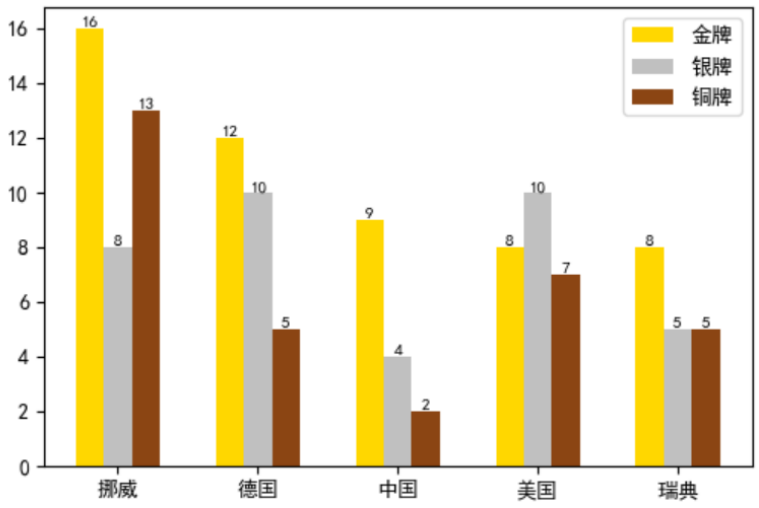

```
1
```

2.同位置多柱状图

同一 x 轴位置绘制多个柱状图，主要通过调整柱状图的宽度和每个柱状图x轴的起始位置

排名	国家/地区	金牌	银牌	铜牌	总数
1	挪威	16	8	13	37
2	德国	12	10	5	27
3	中国	9	4	2	15
4	美国	8	10	7	25
5	瑞典	8	5	5	18

需要绘制如下图形:

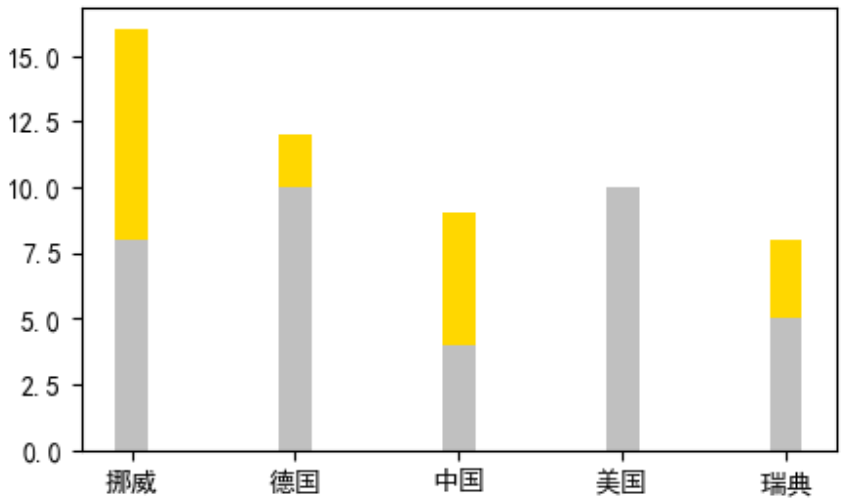


分析：

- 本实例需要对x轴进行计算，因此需要将x轴转数值
- 确定同一x轴中，每个柱状图x轴的起始位置。
- 需要设置图形的宽度
- 图形2的起始位置=图形2起始位置+图形的宽度
- 图形3的起始位置=图形3起始位置+2倍图形的宽度
- 需要给每个柱状图循环显示文本内容
- 显示图例

```
# 国家
countries = ['挪威', '德国', '中国', '美国', '瑞典']

# 金牌个数
gold_medal = [16, 12, 9, 8, 8]
# 银牌个数
silver_medal = [8, 10, 4, 10, 5]
# 铜牌个数
bronze_medal = [13, 5, 2, 7, 5]
width = 0.2
plt.bar(countries, gold_medal,color="gold", width=width)
plt.bar(countries,silver_medal,color="silver", width=width)
```

```
<BarContainer object of 5 artists>
```

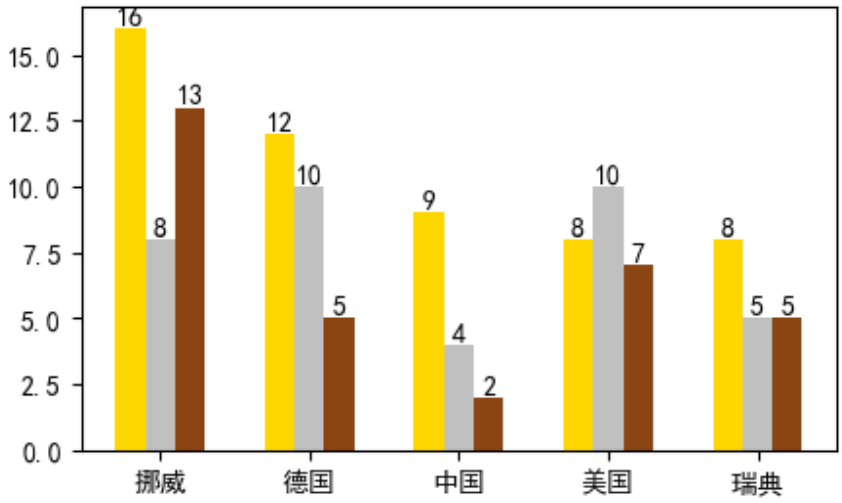


绘制图形

```python
from matplotlib import pyplot as plt
import numpy as np

# 设置基本属性.....
# ........... 省略..........

# 国家
countries = ['挪威', '德国', '中国', '美国', '瑞典']

# 金牌个数
gold_medal = [16, 12, 9, 8, 8]
# 银牌个数
silver_medal = [8, 10, 4, 10, 5]
# 铜牌个数
bronze_medal = [13, 5, 2, 7, 5]

# 1.将x轴转换为数值

x = np.arange(len(countries))
print(x)
# 2.设置图形的宽度
width = 0.2

# ===============确定x起始位置==========
# 金牌起始位置
gold_x = x

# 银牌的起始位置
silver_x = x + width

# 铜牌的起始位置
bronze_x = x + 2 * width


# ==================分别绘制图形
# 金牌图形
plt.bar(gold_x,gold_medal,width=width,color="gold")

# 银牌图形
plt.bar(silver_x,silver_medal,width=width,color="silver")

# 铜牌图形
plt.bar(bronze_x,bronze_medal,width=width, color="saddlebrown")

```

```
# ================将x轴的坐标变回来

# 注意x标签的位置未居中
plt.xticks(x+width, labels=countries)

#-----------显示高度文本----------------
for i in range(len(countries)):
    # 金牌的文本设置

 plt.text(gold_x[i],gold_medal[i],gold_medal[i],va="bottom",ha="center")

 plt.text(silver_x[i],silver_medal[i],silver_medal[i],va="bottom",ha="center")

 plt.text(bronze_x[i],bronze_medal[i],bronze_medal[i],va="bottom",ha="center")
# 金牌


# 银牌牌


# 铜牌


# 显示图例
```

```
[0 1 2 3 4]
```




```

```

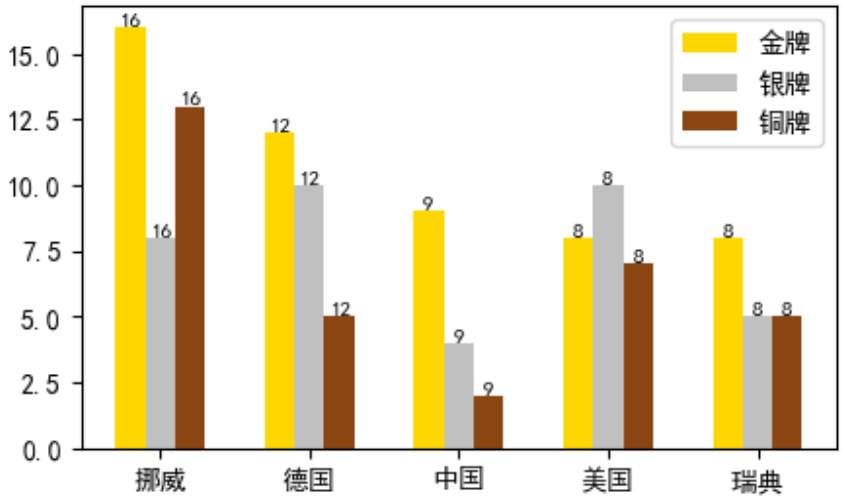
```
1
```

```
1
```

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

# 设置基本属性.....
# ........... 省略..........

# 国家
countries = ['挪威', '德国', '中国', '美国', '瑞典']

# 金牌个数
gold_medal = [16, 12, 9, 8, 8]
# 银牌个数
silver_medal = [8, 10, 4, 10, 5]
# 铜牌个数
bronze_medal = [13, 5, 2, 7, 5]

# 1.将x轴转换为数值
x_int = np.arange(len(countries))

# 2.设置图形的宽度
width = 0.2

# 确定x起始位置
gold_x = x_int  # 金牌起始位置

silver_x = x_int+width  # 银牌的起始位置

bronze_x = x_int + 2*width # 铜牌的起始位置

# 分别绘制图形

plt.bar(gold_x, gold_medal, width=width,color="gold", label="金牌")
 # 金牌图形

plt.bar(silver_x, silver_medal, width=width,
color="silver",label="银牌") # 银牌图形

plt.bar(bronze_x, bronze_medal, width=width,
color="saddlebrown",label="铜牌") # 铜牌图形

# 将x轴的坐标变回来

# plt.xticks(x_int,labels=countries)
```

```
# 移动x标记的位置,再替换内容
plt.xticks(x_int + width,labels=countries)

#-----------显示高度文本----------------
# # 金牌
# for x,y in zip(gold_x,gold_medal):
#     plt.text(x,y,y,va="bottom",ha="center",fontsize=8)

# # 银牌牌
# for x,y in zip(silver_x, silver_medal):
#     plt.text(x,y,y,va="bottom",ha="center",fontsize=8)

# # 铜牌
# for x,y in zip(bronze_x, bronze_medal):
#     plt.text(x,y,y,va="bottom",ha="center",fontsize=8)

  # 金牌 # 银牌 # 铜牌
for i in range(len(countries)):
    # 金牌
    plt.text(gold_x[i],gold_medal[i],
gold_medal[i],va="bottom",ha="center",fontsize=8)
    # 银牌
    plt.text(silver_x[i],silver_medal[i],
gold_medal[i],va="bottom",ha="center",fontsize=8)
    # 铜牌
    plt.text(bronze_x[i],bronze_medal[i],
gold_medal[i],va="bottom",ha="center",fontsize=8)

# 显示图例
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d69aff1748>
```



绘制各国的金牌榜 银牌榜和铜牌榜 总榜单

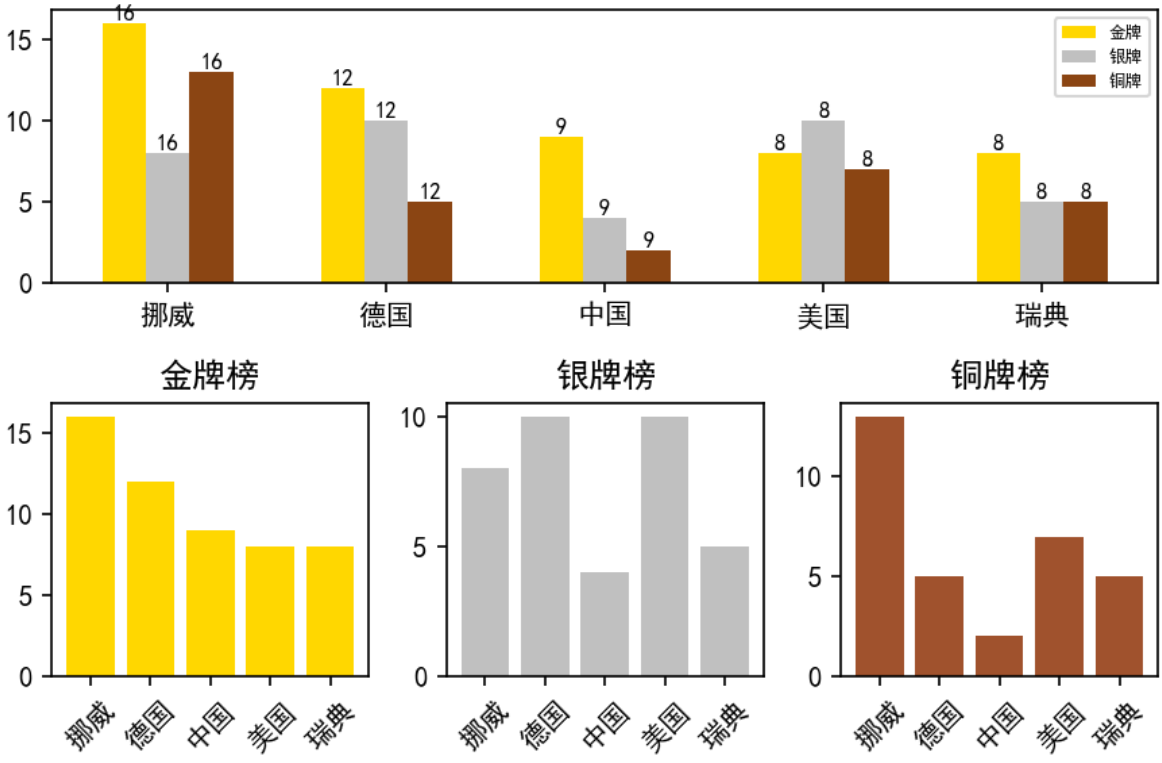
其他知识点:在 Matplotlib 中旋转 X 轴刻度标签文本

- plt.xticks(rotation=) 旋转 Xticks 标签文本
- fig.autofmt_xdate(rotation=) 旋转 Xticks 标签文本
- ax.set_xticklabels(xlabels, rotation=) 旋转 Xticks 标签文本
- plt.setp(ax.get_xticklabels(), rotation=) 旋转 Xticks 标签文本
- ax.tick_params(axis='x', labelrotation=) 旋转 Xticks 标签文本

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

# 设置基本属性.....
# ........... 省略..........

# 国家
countries = ['挪威', '德国', '中国', '美国', '瑞典']

# 金牌个数
gold_medal = [16, 12, 9, 8, 8]
# 银牌个数
silver_medal = [8, 10, 4, 10, 5]
# 铜牌个数
bronze_medal = [13, 5, 2, 7, 5]

# 设置画布
fig = plt.figure(figsize=(6,4),dpi=150)

```

```python
# 一个画布分为2行3列,定位第一个
ax1 = fig.add_subplot(234)

ax1.set_title("金牌榜")
# 旋转x标签
ax1.tick_params(axis="x",rotation=45)

# 金牌榜
ax1.bar(countries, gold_medal, color="gold")


# 一个画布分为2行3列,定位第二个
ax2 = fig.add_subplot(235)

ax2.set_title("银牌榜")
# 旋转x标签
ax2.tick_params(axis="x",rotation=45)

# 银牌榜
ax2.bar(countries, silver_medal, color="silver")



# 一个画布分为2行3列,定位第二个
ax3 = fig.add_subplot(236)

ax3.set_title("铜牌榜")
# 旋转x标签
ax3.tick_params(axis="x",rotation=45)

# 铜牌榜
ax3.bar(countries, bronze_medal, color="#A0522D")


# ================绘制总图:绘制2行一列,移动到第一行======
ax = fig.add_subplot(211)

# 1.将x轴转换为数值
x_int = np.arange(len(countries))

# 2.设置图形的宽度
width = 0.2

# 确定x起始位置
gold_x = x_int  # 金牌起始位置

```

```python
silver_x = x_int+width  # 银牌的起始位置

bronze_x = x_int + 2*width # 铜牌的起始位置

# 分别绘制图形

ax.bar(gold_x, gold_medal, width=width,color="gold", label="金牌")
 # 金牌图形

ax.bar(silver_x, silver_medal, width=width,
color="silver",label="银牌") # 银牌图形

ax.bar(bronze_x, bronze_medal, width=width,
color="saddlebrown",label="铜牌") # 铜牌图形

# 将x轴的坐标变回来

# plt.xticks(x_int,labels=countries)
# 移动x标记的位置,再替换内容
ax.set_xticks(x_int + width)

ax.set_xticklabels(countries )

#-----------显示高度文本----------------
# 金牌 # 银牌 # 铜牌
for i in range(len(countries)):
    # 金牌
    ax.text(gold_x[i],gold_medal[i],
gold_medal[i],va="bottom",ha="center",fontsize=8)
    # 银牌
    ax.text(silver_x[i],silver_medal[i],
gold_medal[i],va="bottom",ha="center",fontsize=8)
    # 铜牌
    ax.text(bronze_x[i],bronze_medal[i],
gold_medal[i],va="bottom",ha="center",fontsize=8)


# 显示图例
ax.legend(fontsize=6 )

# 处理标题覆盖
plt.tight_layout()
```



> 可以思考如何将金牌榜 银牌榜 和铜牌榜 都按照从大到小排序

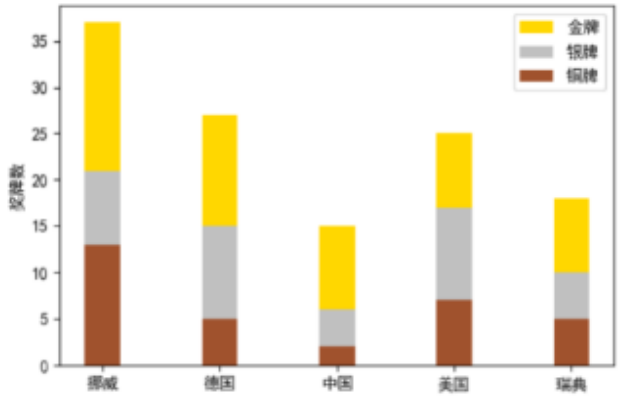
```
1
```

```
1
```

堆叠柱状图

所谓堆叠柱状图就是将不同数组别的柱状图堆叠在一起，堆叠后的柱状图高度显示了两者相加的结果值。

如图:



分析:

- 金牌榜的起始高度为：铜牌数据+银牌数据
- 银牌榜的起始高度为：银牌高度
- 铜牌榜的起始高度为：0
- 起始位置的数据相加需要使用numpy的相关知识
- 需要确定柱状图的颜色
- 显示图例

```
countries = ['挪威', '德国', '中国', '美国', '瑞典']
# 金牌个数
gold_medal = np.array([16, 12, 9, 8, 8])
# 银牌个数
silver_medal = np.array([8, 10, 4, 10, 5])
# 铜牌个数
bronze_medal = np.array([13, 5, 2, 7, 5])

# 绘制堆叠图

# 宽度
width = 0.3
# 绘制金牌
plt.bar(countries, gold_medal, color='gold', label='金牌',
        bottom=silver_medal + bronze_medal,width=width)

# 绘制银牌
plt.bar(countries, silver_medal, color='silver', label='银牌',
bottom=bronze_medal,width=width)

# 绘制铜牌
plt.bar(countries, bronze_medal, color='#A0522D', label='铜
牌',width=width)

# 设置坐标轴
plt.ylabel('奖牌数')

# 设置图例
plt.legend(loc='upper right')


# 设置文本值
```

```
for i in range(len(countries)):
    max_y = bronze_medal[i]+silver_medal[i]+gold_medal[i]
    plt.text(countries[i], max_y, max_y, va="bottom", ha="center")
```

```

```

```

```

作业2

三天中3部电影的票房变化

movie = ['千与千寻', '玩具总动员4', '黑衣人：全球追缉']

real_day1 = [4053, 7548, 6543]

real_day2 = [1840, 4013, 3421]

real_day3 = [2080, 1673, 2342]

作业,按照以上数据

- 1.绘制同位置多柱状图
- 2.绘制堆叠图

```

```