Matplotlib

Matplotlib是一个Python 2D绘图库，它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。Matplotlib可用于Python脚本，Python和IPython (opens new window)Shell、Jupyter (opens new window)笔记本，Web应用程序服务器和四个图形用户界面工具包。

In [1]:

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

1. Matplotlib绘图基础

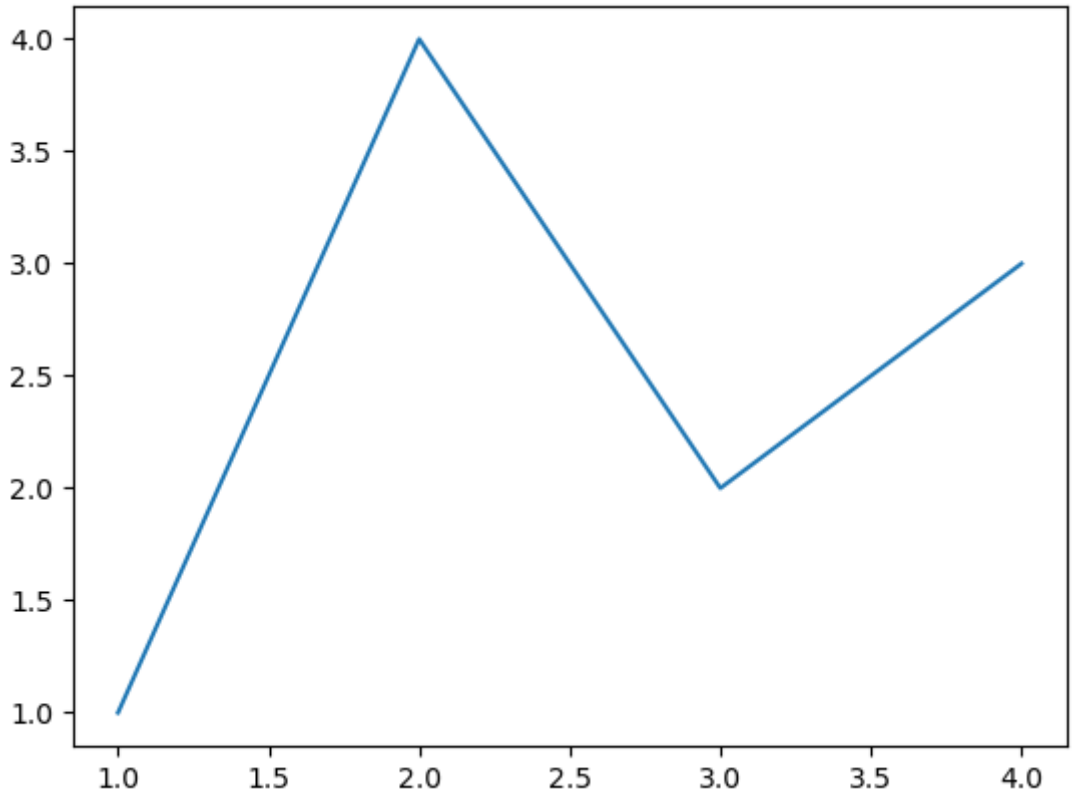
常用的几个函数包括：

- 通过plt.figure()产生一个空的绘制面板
- 通过plt.plot()函数绘制简单的曲线
- 通过show()函数显示绘制区域
- 通过savefig()函数保存绘制图像

In [2]:

```
fig = plt.figure()
plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
fig.show()
fig.savefig('first.png')
```

```
C:\Users\lvpeng\AppData\Local\Temp\ipykernel_6748\732193854.py:3: UserWarning: Ma
tplotlib is currently using module://matplotlib_inline.backend_inline, which is a
non-GUI backend, so cannot show the figure.
  fig.show()
```



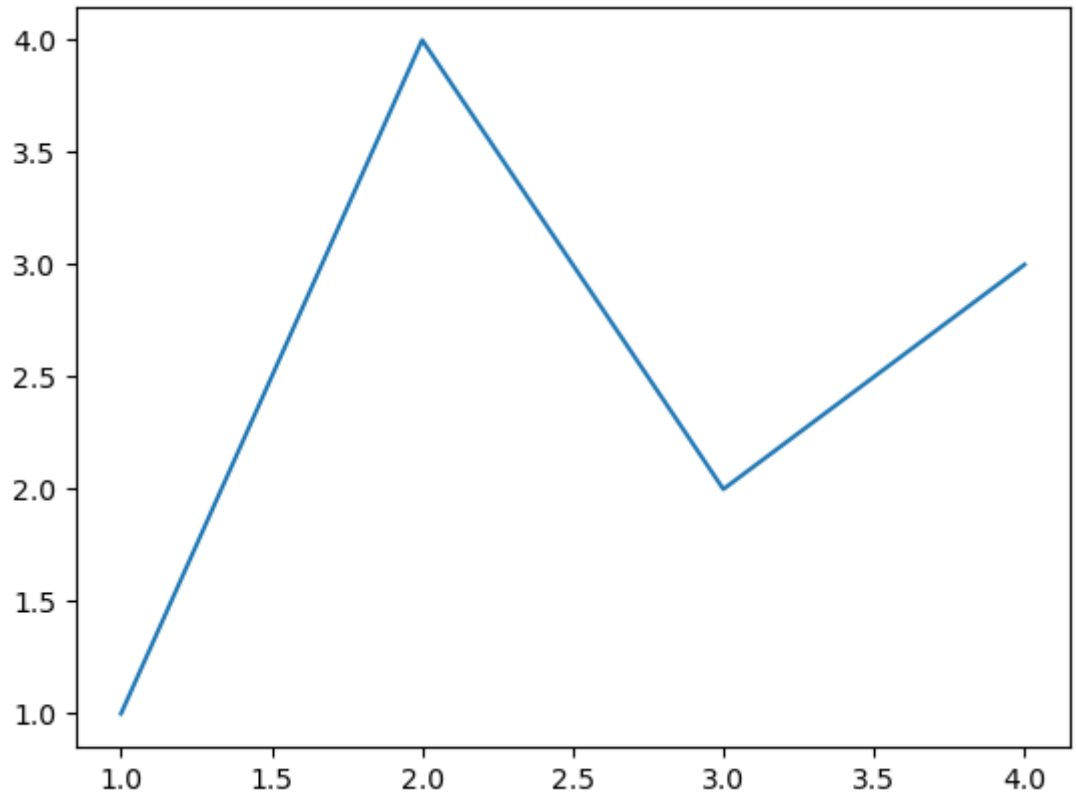
或者时直接采用subplots返回绘制面板的句柄对象，然后直接对对象进行操作

In [3]:

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the axes.
```

Out[3]:

```
[<matplotlib.lines.Line2D at 0x290e2966f40>]
```



2. figure() 具体参数

```
def figure( num=None,  # autoincrement if None, else integer from 1-N
            figsize=None,  # defaults to rc figure.figsize
            dpi=None,  # defaults to rc figure.dpi

            facecolor=None,  # defaults to rc figure.facecolor
            edgecolor=None,  # defaults to rc figure.edgecolor
            frameon=True,
            FigureClass=Figure,
            clear=False,
            **kwargs
            )
```

以下三种形式都是合法的:

```
fig = figure()
fig = figure(1)
fig = figure(figsize=[6,6],dpi=90)
```

3. 通过subplots()和subplot() 来决定绘制面板的子面板

（1）若要绘制多个子图像时，可以使用subplots()来创建绘图面板，并决定绘制区域中有几个子图，进而通过返回的子图句柄来在相应子面板上进行绘制

> ` python def subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None, **fig_kw): """ Create a figure and a set of subplots. This utility wrapper makes it convenient to create common layouts of subplots, including the enclosing figure object, in a single call.

```
Parameters
----------
    nrows, ncols : int, optional, default: 1, Number of rows/columns of the subplot gri
d.
Returns
----------
    fig: Figure
    ax: Axes or array of Axes
```

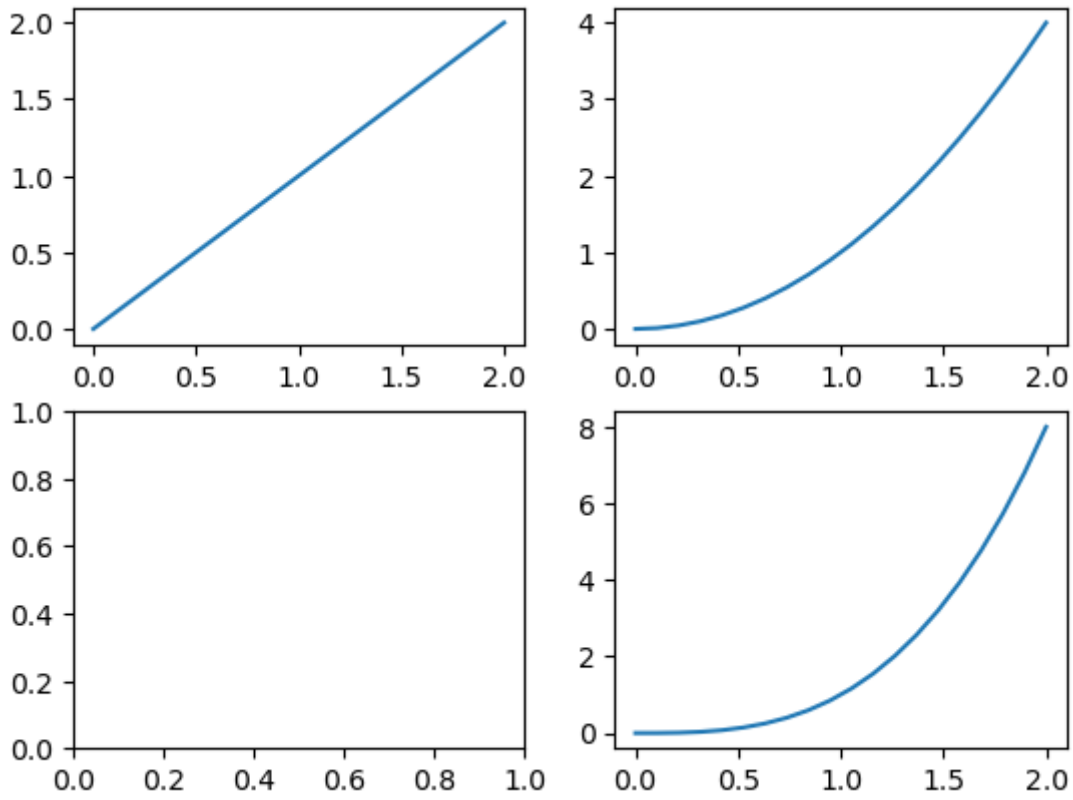
`

In [4]:

```
x = np.linspace(0, 2, 20)
fig, axs = plt.subplots(2,2)  #  带有四个子图的绘制区域
axs[0,0].plot(x,x)
axs[0,1].plot(x,x**2)
axs[1,1].plot(x,x**3)
```

Out[4]:

```
[<matplotlib.lines.Line2D at 0x290e31896d0>]
```



(2) 也可使用plt.subplot()函数，直接在绘图主面板上添加子面板，并在子面板上进行绘制

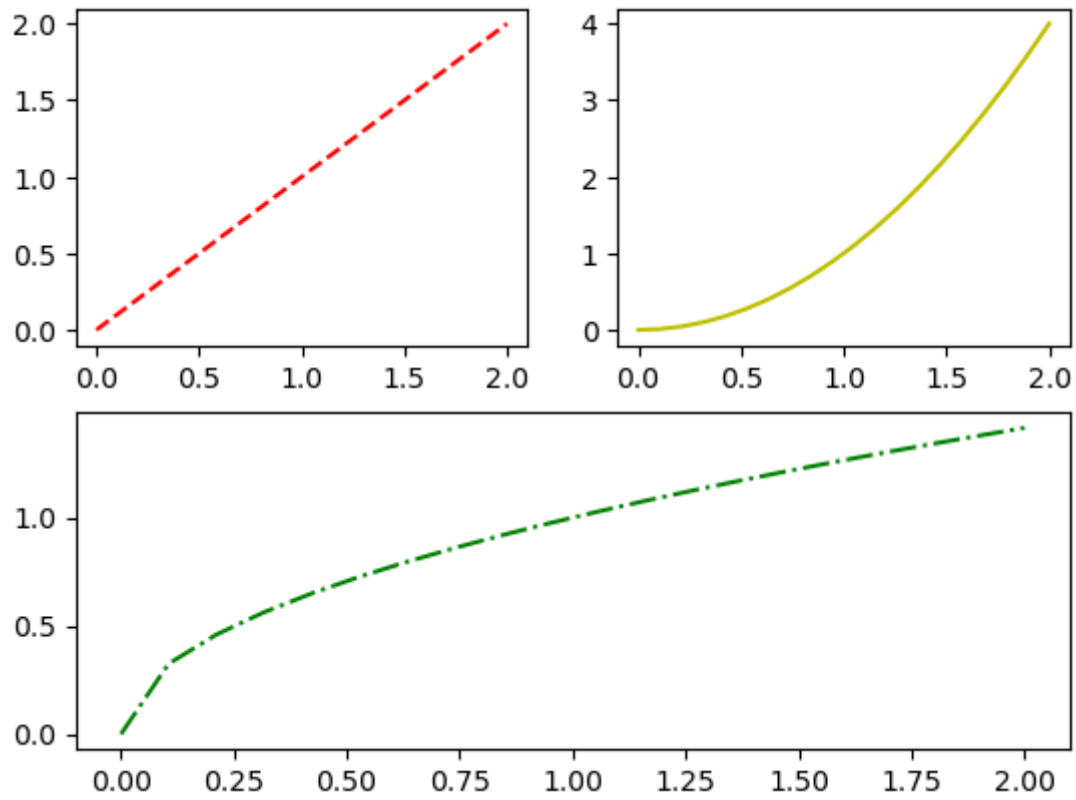
```
subplot(nrows, ncols, index, **kwargs)
subplot(pos, **kwargs)
subplot(ax)
```

In [5]:

```
import  matplotlib.pyplot as plt
plt.figure()
plt.subplot(221)
plt.plot(x,x, color="r",linestyle = "--")
plt.subplot(222)
plt.plot(x,x**2,color="y",linestyle = "-")
plt.subplot(212)
plt.plot(x,x**(0.5),color="g",linestyle = "-.")
```

Out[5]:

```
[<matplotlib.lines.Line2D at 0x290e3153370>]
```



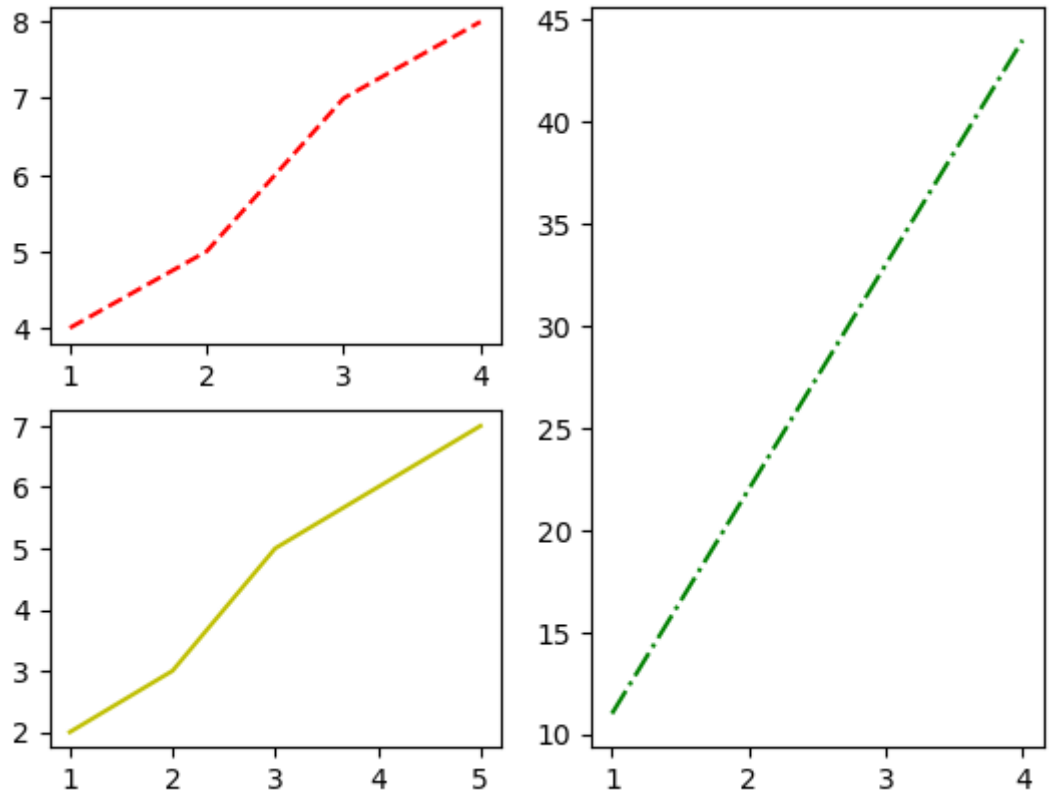
通常情况下，subplot()更加常用，因为其能够对面板进行灵活的操作，请仔细体会以下几个例子

In [6]:

```
plt.figure()

ax1 = plt.subplot(221)
plt.plot([1,2,3,4],[4,5,7,8], color="r",linestyle = "--")

ax2 = plt.subplot(223)
plt.plot([1,2,3,5],[2,3,5,7],color="y",linestyle = "-")

ax3 = plt.subplot(122)
plt.plot([1,2,3,4],[11,22,33,44],color="g",linestyle = "-.")
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x290e29b7370>]
```



4. figure()和subplot()的组合使用

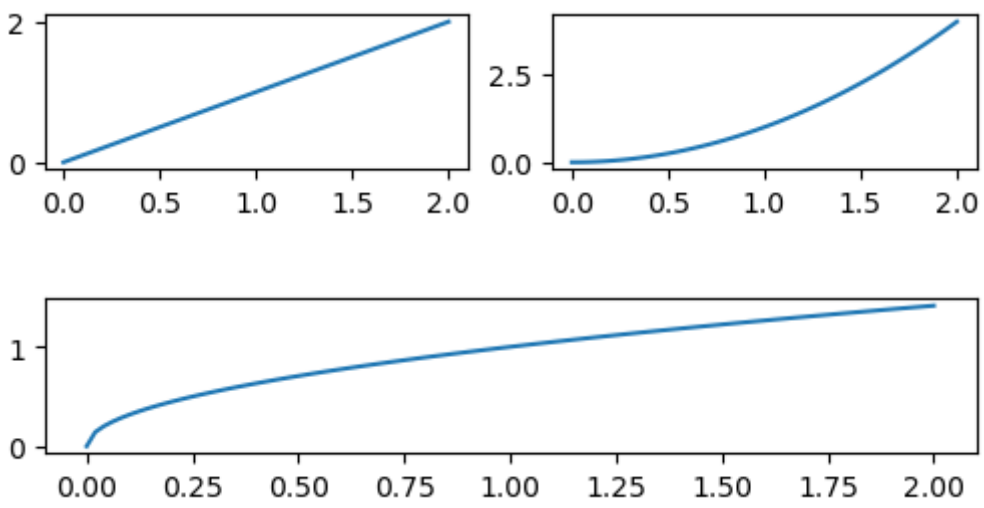
通过figure()以及subplot()可以快速定位到想要绘制的面板，并且顺序可以进行改变，不影响绘制结果

In [7]:

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

plt.figure(1,figsize=[6,1]) # 创建第一个画板（figure）

plt.subplot(121) # 第一个画板的第一个子图
plt.plot(x,x)

plt.subplot(122) # 第二个画板的第二个子图
plt.plot(x,x**2)

plt.figure(2,figsize=[6,1]) #创建第二个画板
plt.plot(x,x**(0.5)) # 默认子图命令是subplot(111)

#plt.figure(1) # 调取画板1; subplot(122)仍然被调用中
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x290e35ab460>]
```



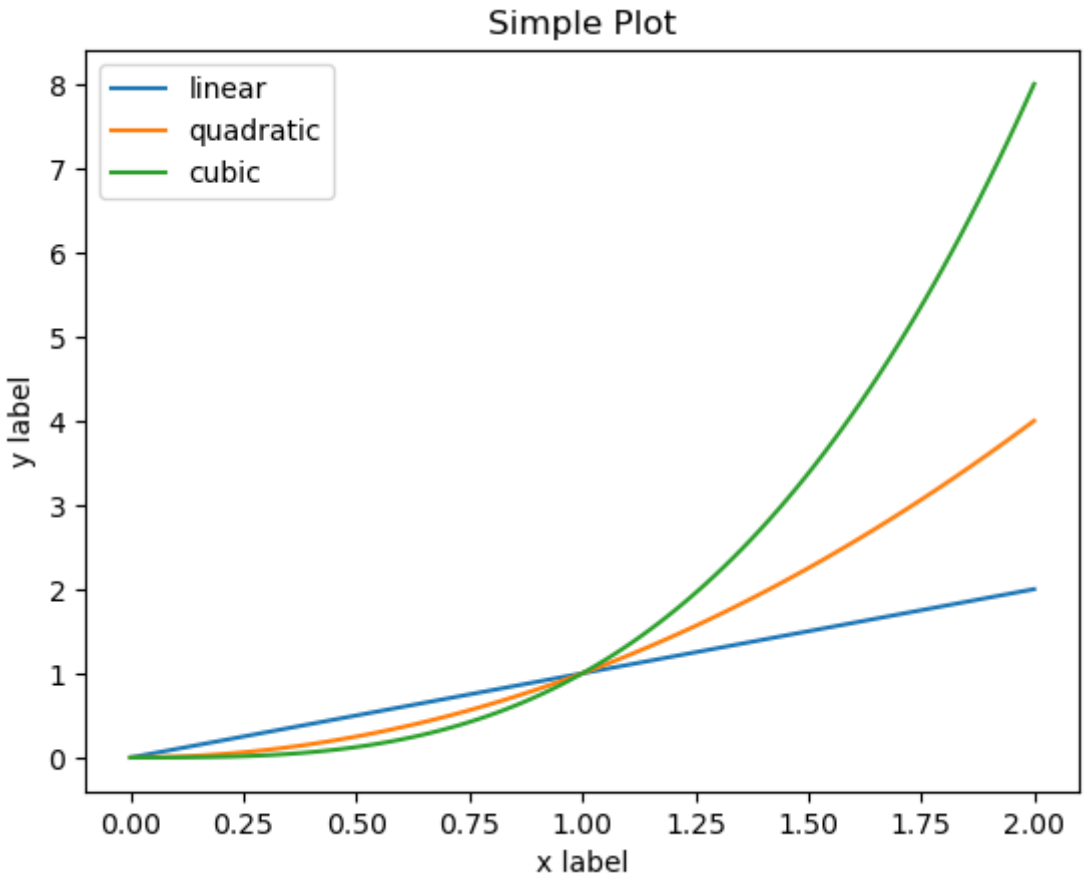
基于上面的内容，我们可以采用 OO-style方式来绘制几条曲线

In [8]:

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

fig, ax = plt.subplots()
ax.plot(x, x, label='linear')  # 在当前子图上绘制一些点（线性）
ax.plot(x, x**2, label='quadratic')  # 在当前子图上绘制一些点（二次方）
ax.plot(x, x**3, label='cubic')  # 在当前子图上绘制一些点（三次方）
ax.set_xlabel('x label')  # 设置坐标轴的名称
ax.set_ylabel('y label')  # 设置坐标轴的名称
ax.set_title("Simple Plot")  # 设置绘制子图的标题区域
ax.legend()  # 添加图例
```

Out[8]:

```
<matplotlib.legend.Legend at 0x290e363dc70>
```



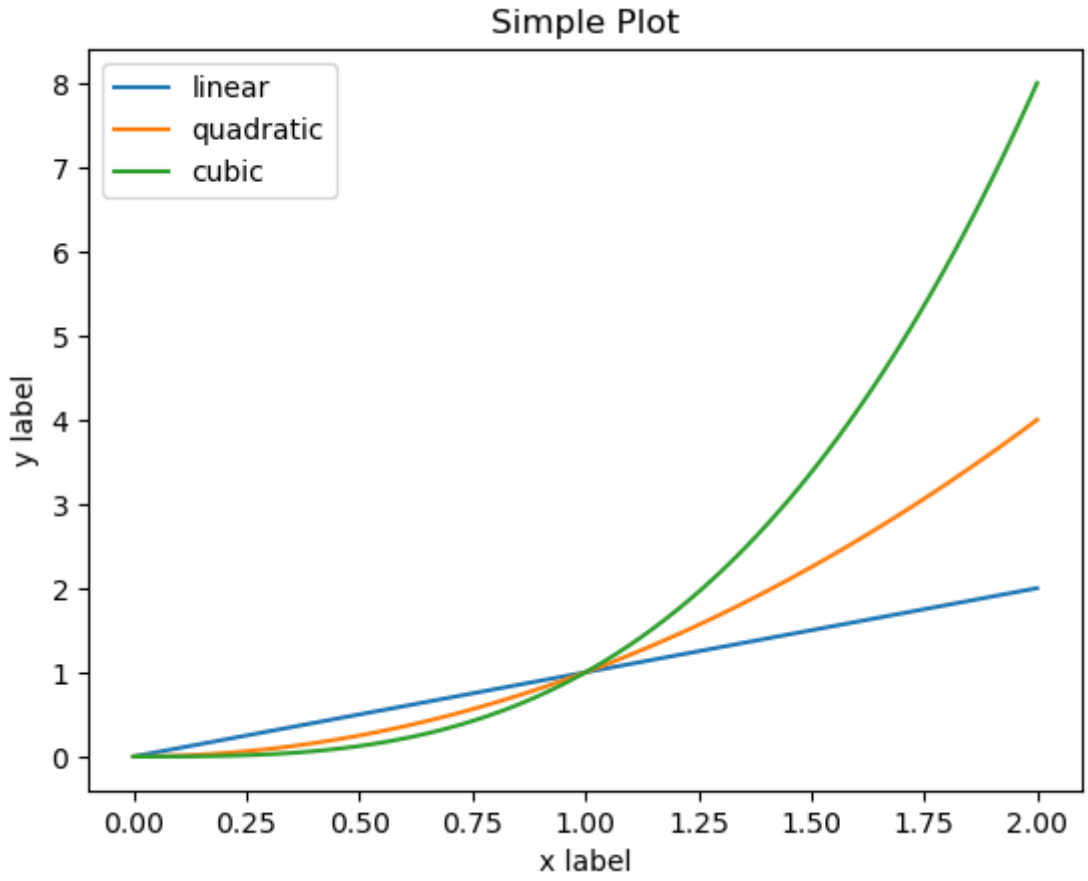
或者是采用plt-style方式来绘制

In [9]:

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

plt.plot(x, x, label='linear')  # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```

Out[9]:

<matplotlib.legend.Legend at 0x290e3627d60>



5. 坐标轴设置

设置坐标轴的取值范围

```
plt.xlim(args, kwargs)
plt.ylim(args, kwargs)
```

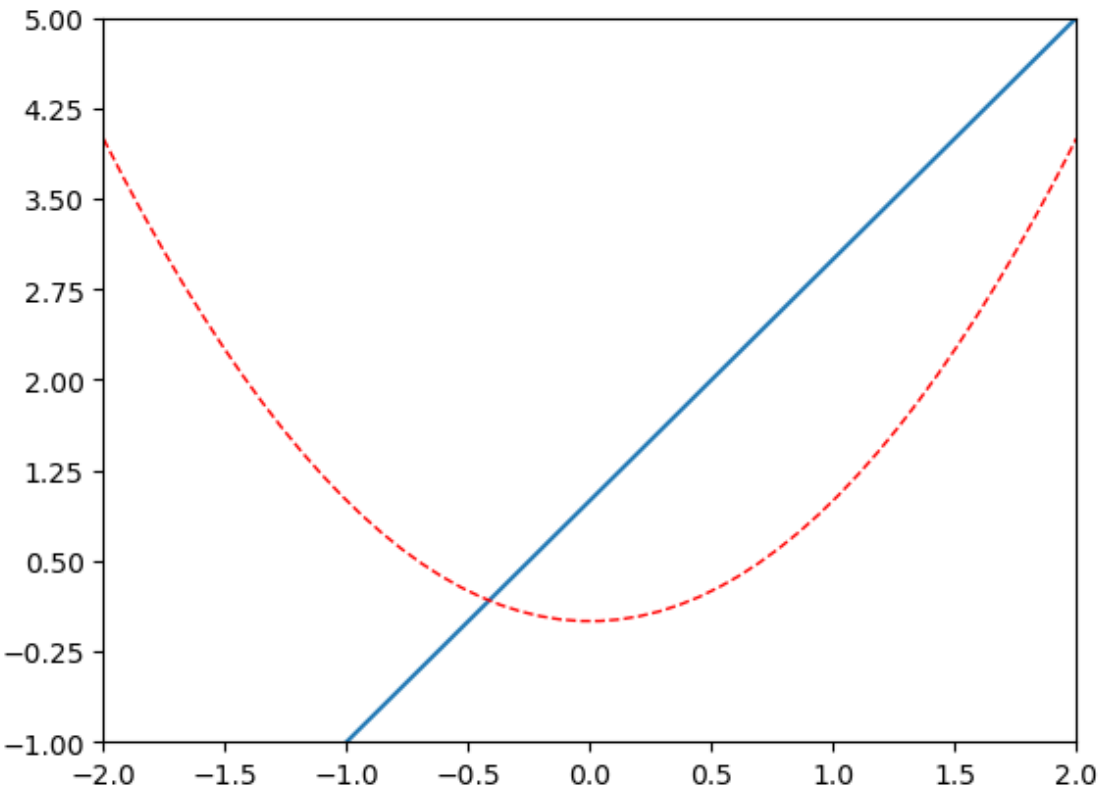
设置坐标轴的刻度

```
plt.xticks(ticks=None, labels=None, ...)
plt.yticks(ticks=None, labels=None, ...)
```

- 第一个参数为数组类型，存放要显示的刻度列表
- 当第一个参数为空时，移除坐标轴的刻度列表

In [10]:

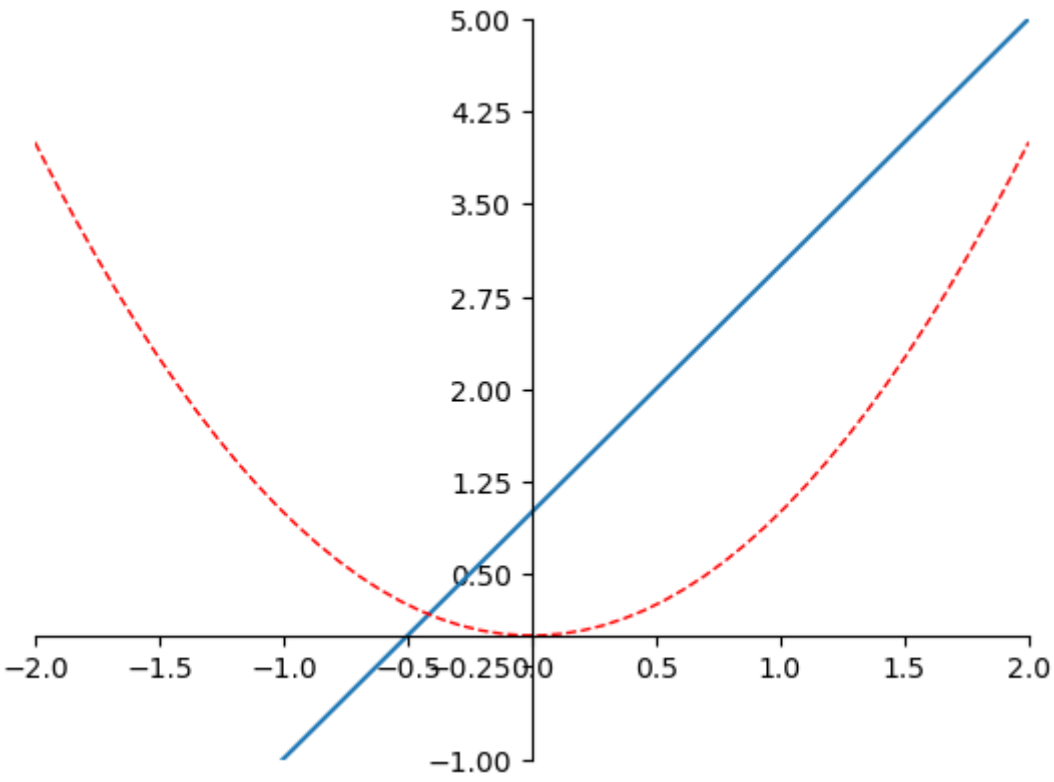
```
# 绘制普通图像
x = np.linspace(-2, 2, 50)
plt.figure()
plt.plot(x, 2 * x + 1)
plt.plot(x, x**2, color = 'red', linewidth = 1.0, linestyle = '--')

# 设置坐标轴的取值范围
plt.xlim((-2, 2))
plt.ylim((-1, 5))

# 设置坐标轴刻度,
plt.xticks(np.linspace(-2, 2, 9))
plt.yticks(np.linspace(-1, 5, 9))
plt.show()
```



此外，通过matplotlib提供的其他坐标轴的绘制函数，可以实现多种图形的绘制。

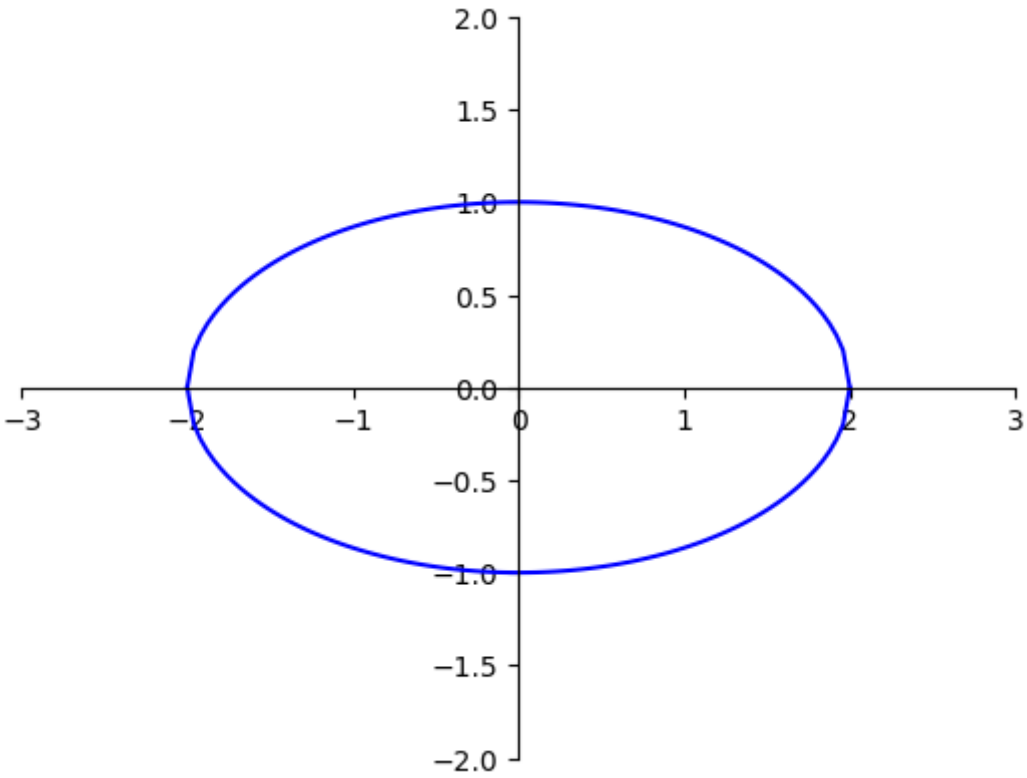
In [11]:

```
x = np.linspace(-2, 2, 50)
plt.figure()
plt.plot(x, 2 * x + 1)
plt.plot(x, x**2, color = 'red', linewidth = 1.0, linestyle = '--')
plt.xlim((-2, 2))
plt.ylim((-1, 5))
plt.xticks(np.linspace(-2, 2, 9))
plt.yticks(np.linspace(-1, 5, 9))


ax = plt.gca() # 获取当前的坐标轴, gca = get current axis
ax.spines['right'].set_color('none') # 设置右边框和上边框
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')# 设置x坐标轴为下边框
ax.yaxis.set_ticks_position('left')# 设置y坐标轴为左边框
ax.spines['bottom'].set_position(('data', 0))# 设置x轴, y轴在(0, 0)的位置
ax.spines['left'].set_position(('data', 0))
plt.show()
```



In [12]:

```
x = np.linspace(-2, 2, 100)
y = np.sqrt(1-x**2/4)
plt.figure()
plt.plot(x,y,color = 'blue')
plt.plot(x,-y,color = 'blue')
plt.xlim((-3, 3))
plt.ylim((-2, 2))
ax = plt.gca() # 获取当前的坐标轴, gca = get current axis
ax.spines['right'].set_color('none') # 设置右边框和上边框
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_position(('data', 0))# 设置x轴, y周在(0, 0)的位置
ax.spines['left'].set_position(('data', 0))
```



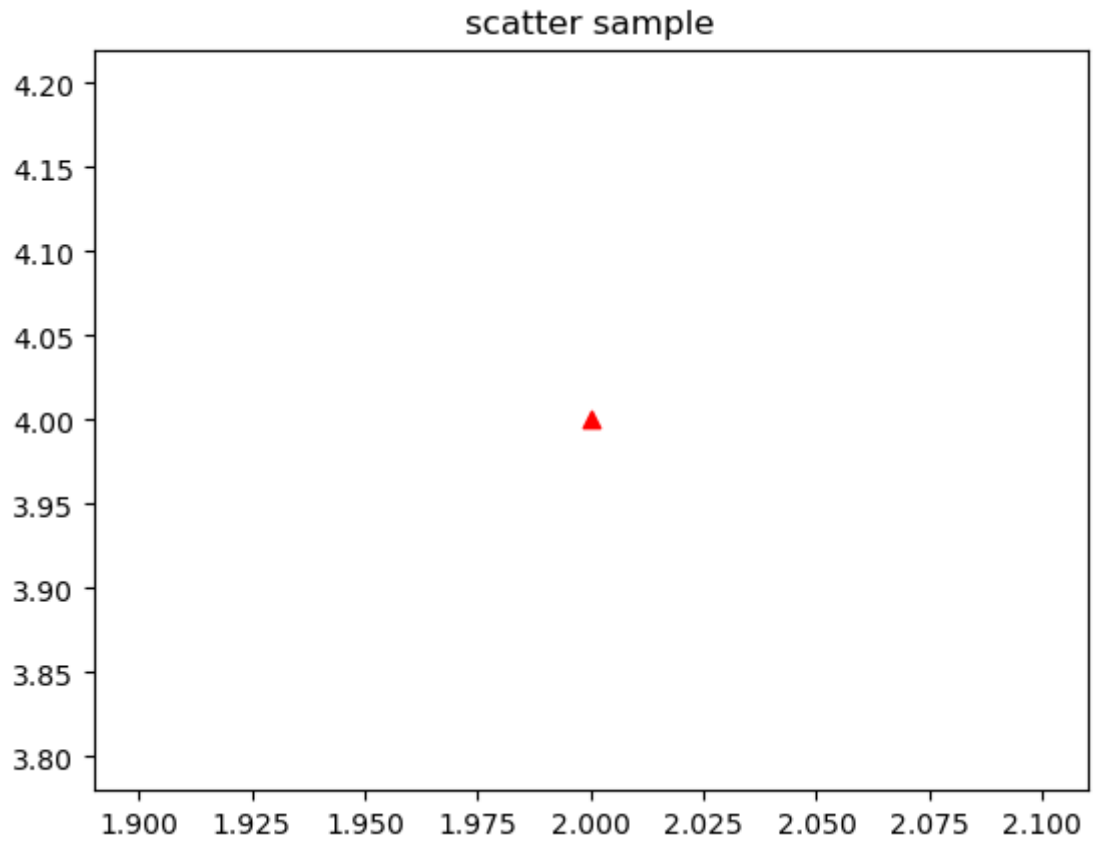
6. 散点图

散点图经常用来显示分布或者比较几个变量的相关性或者分组。

要绘制单个点需要使用scatter()函数，先看一个最简单的示例：

In [13]:

```
plt.title('scatter sample')
plt.scatter(2, 4, color='red', marker='^')
plt.show()
```



scatter函数参数官方文档释义：

> ` python matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, ...)

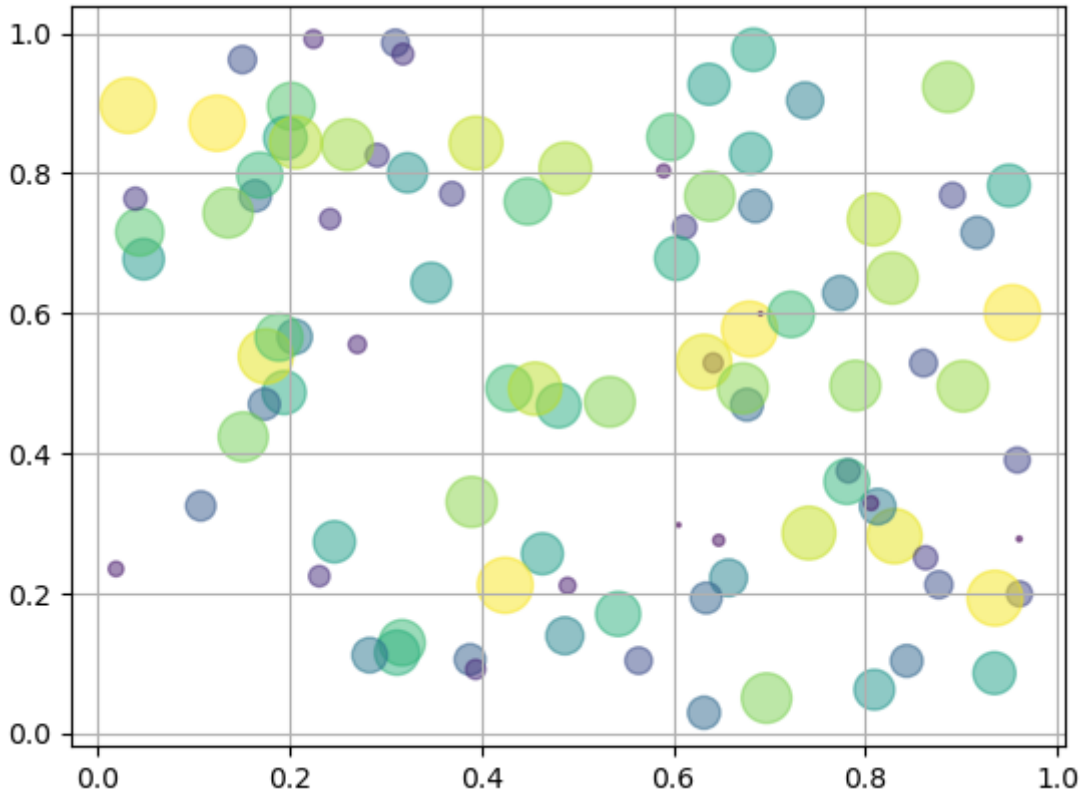
- x, y: float or array-like, shape (n,)The data positions.
- s: float or array-like, shape (n,), optional
- c: array-like or list of colors or color, optional
- marker: MarkerStyle, default: rcParams["scatter.marker"] (default: 'o')
- alpha： float, default: None, The alpha blending value, between 0 (transparent) and 1 (opaque).
- linewidths： float or array-like, default: rcParams["lines.linewidth"] (default: 1.5). The linewidth of the marker edges. Note: The default edgecolors is 'face'. You may want to change this as well.

参数	英文	中文
“.”	point	点
“,”	pixel	像素
“o”	circle	圆
“v”	triangle_down	下三角

参数	英文	中文
“^”	triangle_up	上三角
“<”	triangle_left	左三角
“>”	triangle_right	右三角
“8”	octagon	八角形
“s”	square	正方形
“p”	pentagon	五角星
“*”	star	星号
“h”	hexagon1	六边形1
“H”	hexagon2	六边形2
“+”	plus	加号
“x”	x	x
“D”	diamond	钻石
“d”	thin_diamond	细的钻石
“-“	hline	水平方向的线

In [14]:

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

n = 100
x,y,z = np.random.rand(3,n)

plt.figure()
plt.scatter(x, y, c=z*255, s=z*400, alpha=0.5) # 通过随机数控制每个点的颜色和大小
plt.grid(True)

plt.show()
```



7. 柱状图

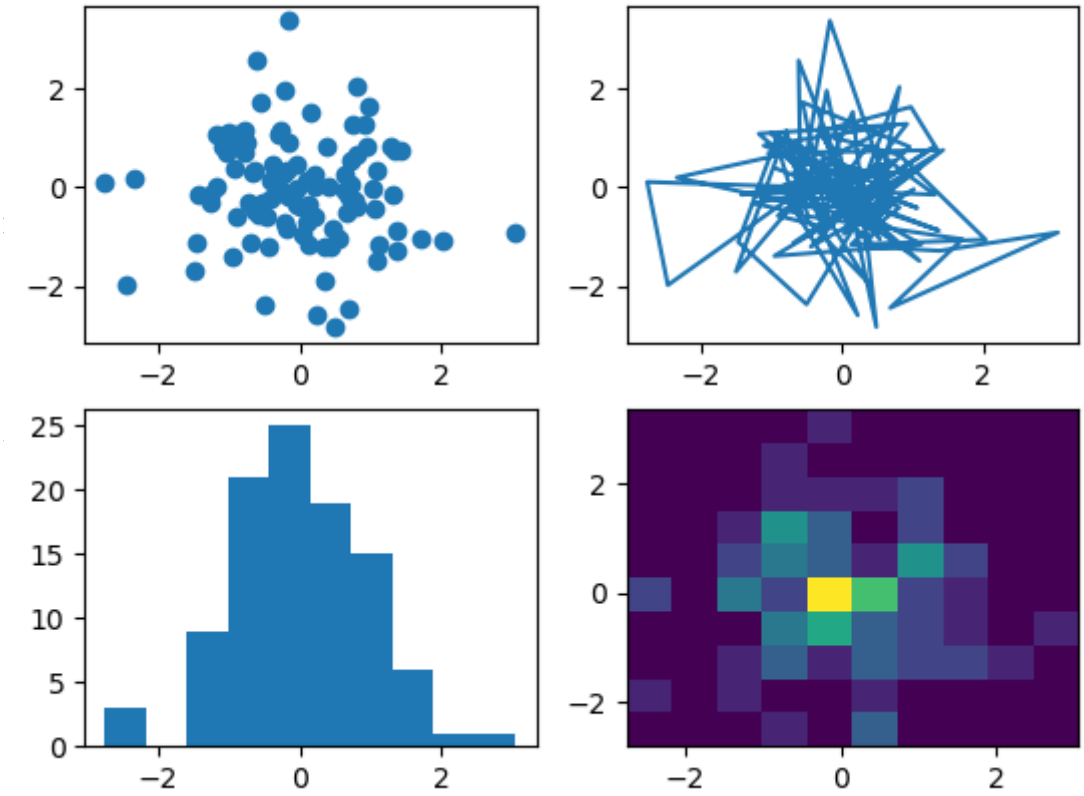
通常可以使用bar()和hist()函数来显示柱状图和直方图

In [15]:

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

x,y = np.random.randn(2, 100)
fig = plt.figure()

plt.subplot(221)
plt.scatter(x,y)

plt.subplot(222)
plt.plot(x, y)

plt.subplot(223)
plt.hist(x)

plt.subplot(224)
plt.hist2d(x,y)
```

Out[15]:

```
(array([[ 0.,  1.,  0.,  0.,  2.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  1.,  1.,  0.,  4.,  2.,  1.,  0.,  0.,  0.],
        [ 1.,  0.,  3.,  4.,  2.,  4.,  5.,  1.,  1.,  0.],
        [ 0.,  0.,  1.,  6., 10.,  3.,  3.,  1.,  0.,  1.],
        [ 3.,  1.,  3.,  3.,  7.,  1.,  0.,  1.,  0.,  0.],
        [ 0.,  0.,  2.,  2.,  2.,  5.,  2.,  2.,  0.,  0.],
        [ 0.,  0.,  2.,  1.,  1.,  2.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.]]),
 array([-2.75313573, -2.17208185, -1.59102796, -1.00997408, -0.4289202 ,
         0.15213368,  0.73318756,  1.31424144,  1.89529533,  2.47634921,
         3.05740309]),
 array([-2.82923263, -2.20907702, -1.5889214 , -0.96876578, -0.34861017,
         0.27154545,  0.89170107,  1.51185668,  2.1320123 ,  2.75216792,
         3.37232353]),
 <matplotlib.collections.QuadMesh at 0x290e4995370>)
```

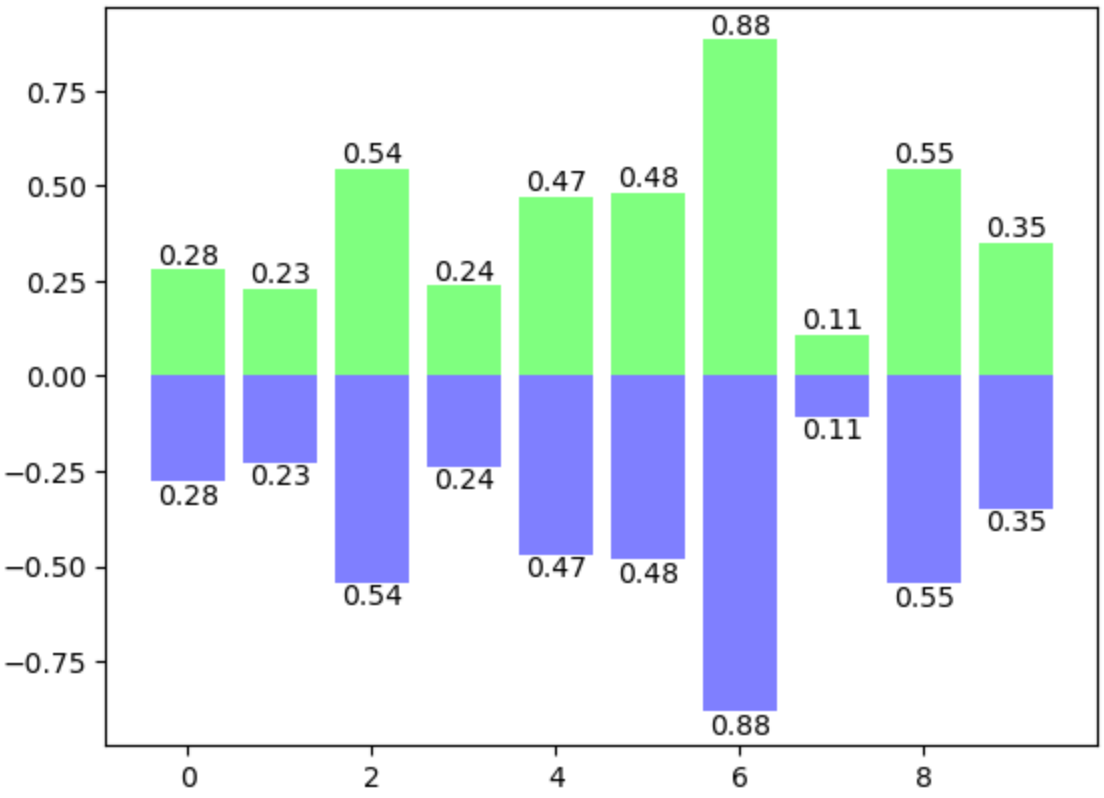


r',

of the bars to the

In [16]:

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

x = np.arange(10)
hig = np.random.random(10)

plt.bar(x,hig,facecolor = '#00FF00',alpha = 0.5)

for i in range(10):
    plt.text(x[i],hig[i],'%.2f'%hig[i],ha = 'center',va = 'bottom')

plt.bar(x,-hig,facecolor = '#0000FF',alpha = 0.5)
for i in range(10):
    plt.text(x[i],-hig[i],'%.2f'%hig[i],ha = 'center',va = 'top')
```



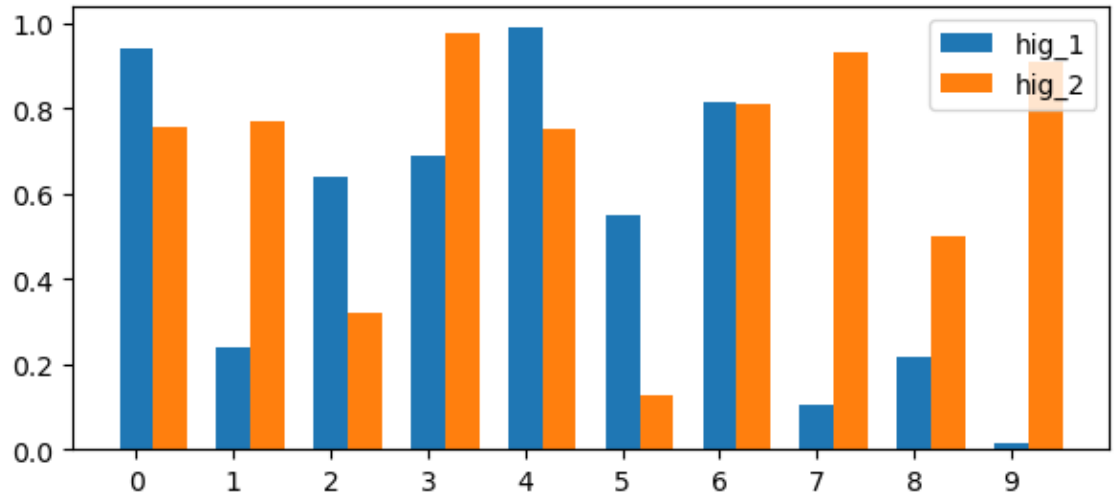
bar()也可以通过设置参数绘制多数据并列柱状图

In [17]:

```
plt.figure(figsize=[7,3])

x = np.arange(10)
hig_1 = np.random.random(10)
hig_2 = np.random.random(10)

plt.xticks(np.arange(10))

bar_width = 0.35
plt.bar(x,hig_1,bar_width,label = 'hig_1')
plt.bar(x+bar_width,hig_2,bar_width,label = 'hig_2')
plt.legend()
```

Out[17]:

```
<matplotlib.legend.Legend at 0x290e48f7a00>
```



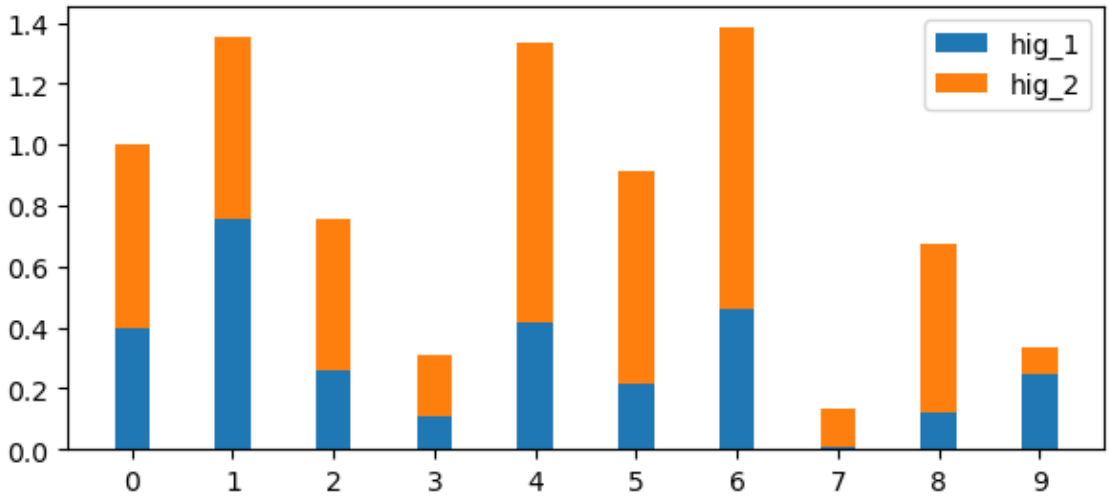
bar()也能通过设置参数绘制堆叠柱状图

In [18]:

```
plt.figure(figsize=[7,3])
x = np.arange(10)

hig_1 = np.random.random(10)
hig_2 = np.random.random(10)

plt.xticks(np.arange(10))

plt.bar(x,hig_1,bar_width,label = 'hig_1')
plt.bar(x,hig_2,bar_width,label = 'hig_2',bottom = hig_1)

plt.legend()
```

Out[18]:

```
<matplotlib.legend.Legend at 0x290e48310a0>
```



8. 饼图

使用pie()函数来实现饼图的绘制：

> ` python matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=None, radius=None, counterclock=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False, rotatelabels=False, *, data=None)

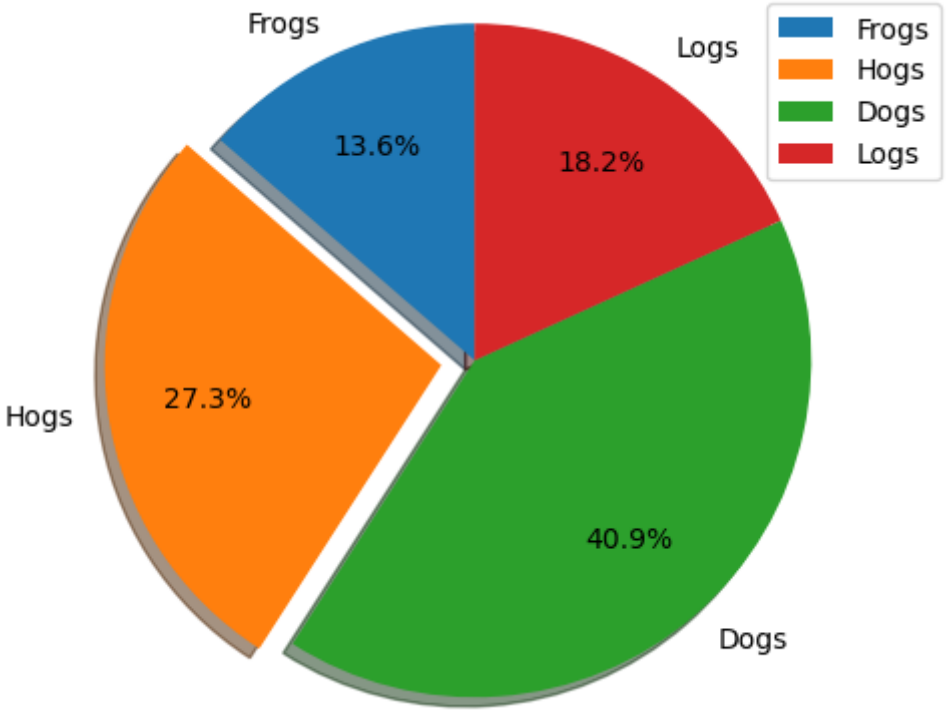
常用参数部分 & 例图

- x: 传入的数据
- explode: 默认x的饼图不爆炸。自定义确定哪一块爆炸&爆炸距离。
- labels和labeldistance: 默认x没有标签，标签位于1.1倍半径处。自定义每块饼的标签，和位置。
- autopct和pctdistance: 默认x不显示每块饼的百分比标注。
- autopct自定义是每块饼的百分比属性，如几位小数，
- pctdistance默认在半径0.6位置显示百分数，自定义百分数距离半径的比例。
- shadow: 默认x是二维平面饼图，没有阴影。自定义饼图是否有阴影属性。

- startangle: 默认x第一块饼和水平面的角度不固定。自定义第一块饼图和水平面的角度。

In [19]:

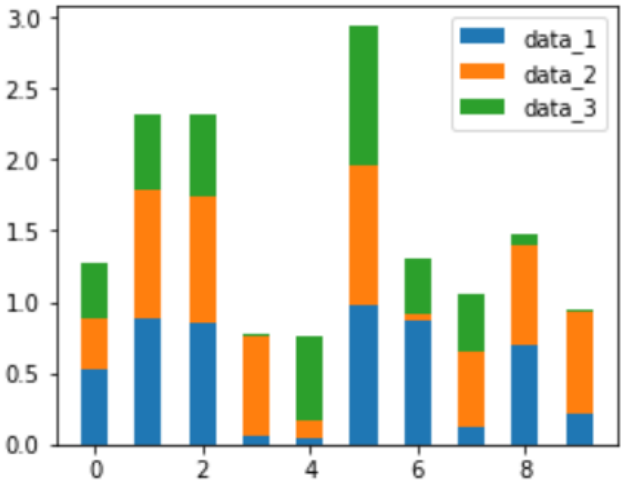
```
import matplotlib.pyplot as plt

labels = ['Frogs', 'Hogs', 'Dogs', 'Logs']
sizes = [15, 30, 45, 20]
explode = (0, 0.1, 0, 0)

fig = plt.figure()
plt.pie(sizes, explode=explode, labels=labels, autopct='%.1f%%', pctdistance=0.7, shadow=True,
plt.axis('equal') # 等价于 ax1.set(aspect='euqal'), 使得饼图在figure窗口放大缩小的过程中, 保持圆
plt.legend()
plt.show()
```

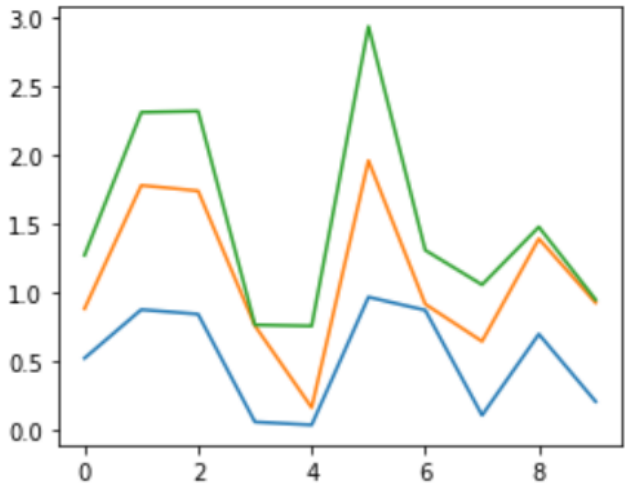


本节作业

在代码空白处填写代码来绘制下述图形，并注意满足以下两个条件

1. 完成基于3个随机数据的【叠加柱状图】和【线型图】的绘制
2. 要求两个子图的位置必须是对角





In [28]:

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

x = np.arange(10)
data_1 = np.random.random(10)
data_2 = np.random.random(10)
data_3 = np.random.random(10)
bar_width = 0.5

plt.figure(figsize=[10,8])

# 此处输入你的代码，并满足以下两个条件
# 1 - 完成基于以上3个随机数据的【叠加柱状图】和【线型图】的绘制
# 2 - 要求两个子图的位置必须是对角

plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```

In [ ]: