matplotlib

一、Matplotlib基础知识

Matplotlib中的基本图表包括的元素

- x轴和y轴 axis
  水平和垂直的轴线
- x轴和y轴刻度 tick
  刻度标示坐标轴的分隔，包括最小刻度和最大刻度
- x轴和y轴刻度标签 tick label
  表示特定坐标轴的值
- 绘图区域（坐标系） axes
  实际绘图的区域
- 坐标系标题 title
  实际绘图的区域
- 轴标签 xlabel ylabel
  实际绘图的区域

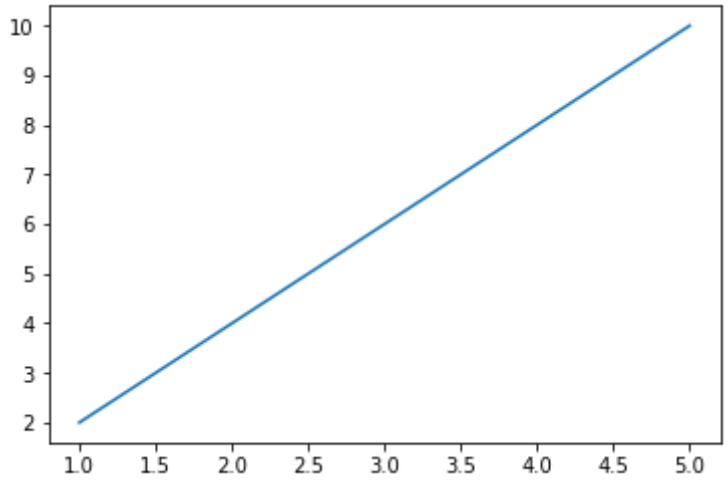
In [1]:
```
import numpy as np
import pandas as pd
from pandas import Series,DataFrame
from matplotlib import pyplot as plt
import matplotlib
```

包含单条曲线的图

- 注意：y,x轴的值必须为数字

In [2]:
```
x = [1,2,3,4,5]
y = [2,4,6,8,10]
plt.plot(x,y)
```

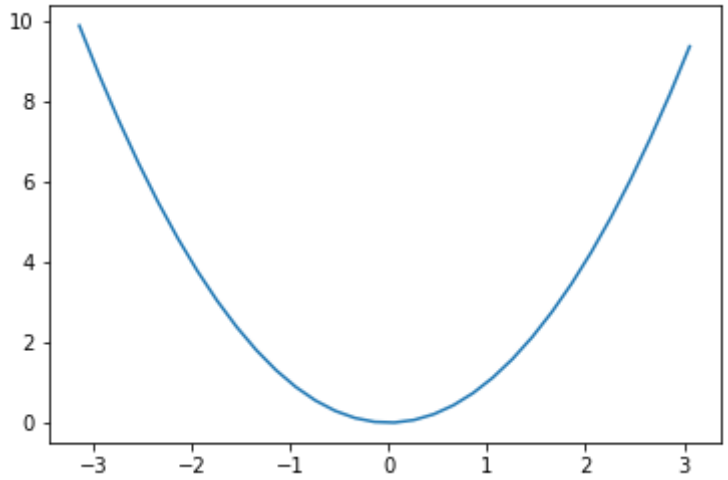
Out[2]: [<matplotlib.lines.Line2D at 0x1e7f6967e80>]



- 绘制抛物线

In [3]:
```
x = np.arange(-np.pi,np.pi,0.2)
y = x**2
plt.plot(x,y)
```

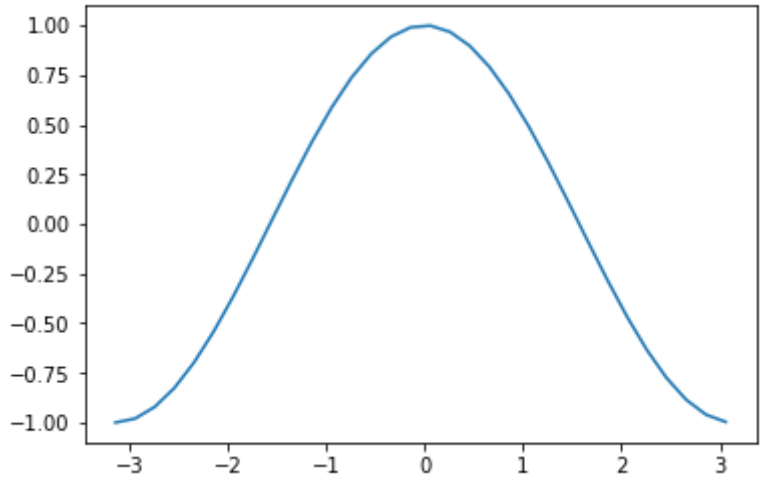
Out[3]: [<matplotlib.lines.Line2D at 0x1e7f6a234a8>]



- 绘制正弦曲线图

In [4]:
```
x
y = np.cos(x)
plt.plot(x,y)
```

Out[4]: [<matplotlib.lines.Line2D at 0x1e7f6a86518>]

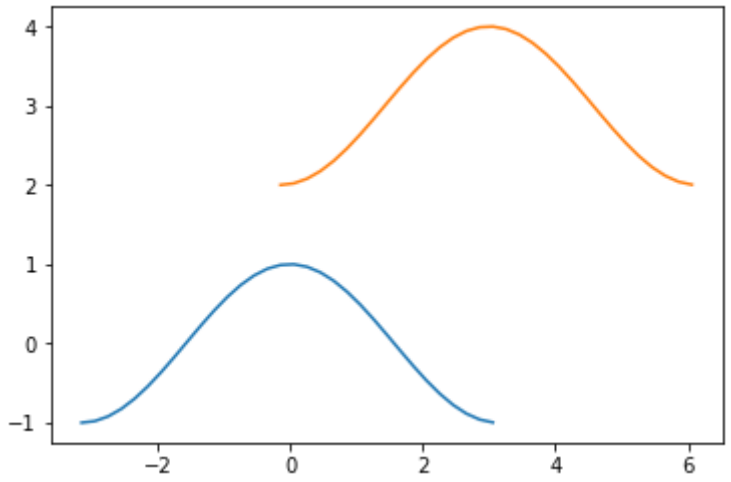


包含多个曲线的图

1、连续调用多次plot函数

In [5]:
```
plt.plot(x,y)
plt.plot(x+3,y+3)
```

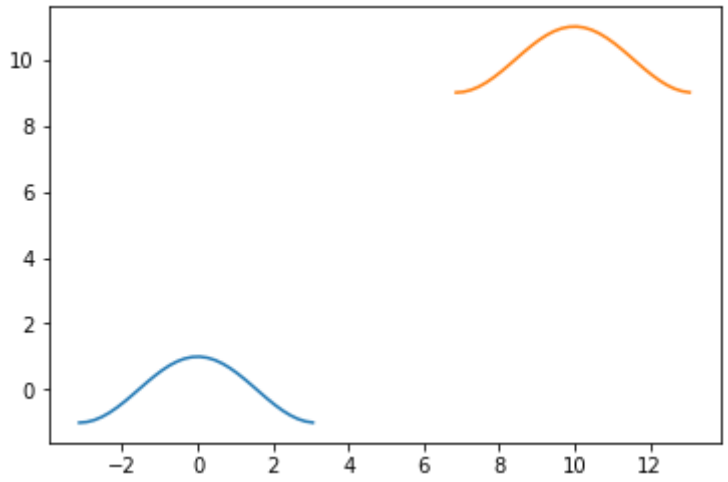
Out[5]:
```
[<matplotlib.lines.Line2D at 0x1e7f6a9c9e8>]
```



2、也可以在一个plot函数中传入多对X,Y值，在一个图中绘制多个曲线

In [6]:
```
plt.plot(x,y,x+10,y+10)
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x1e7f6b56048>,
 <matplotlib.lines.Line2D at 0x1e7f6b56198>]
```



将多个曲线图绘制在一个table区域中：对象形式创建表图

- a=plt.subplot（row,col,loc） 创建曲线图
- a.plot(x,y) 绘制曲线图

- 绘制一个两行两列的曲线图阵,并设置网格

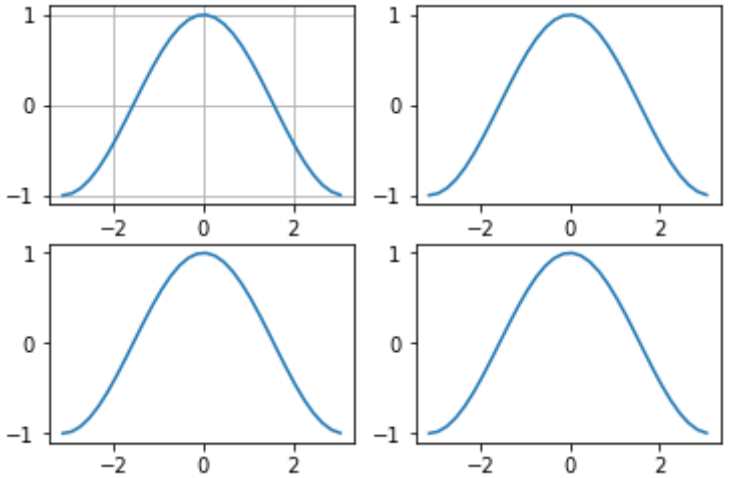
In [7]:
```
ax1 = plt.subplot(2,2,1)
ax1.plot(x,y)
ax1.grid()  # 显示网格

ax2 = plt.subplot(2,2,2)
ax2.plot(x,y)

ax3 = plt.subplot(2,2,3)
ax3.plot(x,y)

ax4 = plt.subplot(2,2,4)
ax4.plot(x,y)
```

Out[7]: [<matplotlib.lines.Line2D at 0x1e7f6c11c50>]



网格线 plt.gride(xxx)

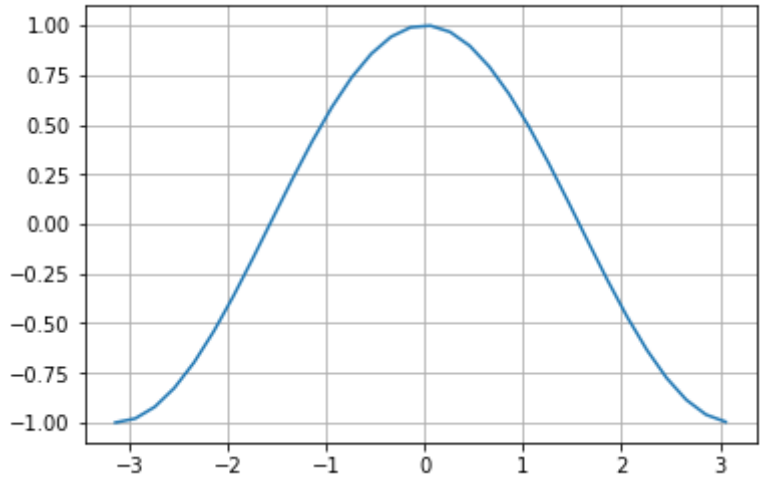
参数:

- axis
- color:支持十六进制颜色
- linestyle: -- -. :
- alpha

- 绘制一个正弦曲线图，并设置网格

In [8]:
```
plt.grid(axis='both')
plt.plot(x,y)
```

Out[8]: [<matplotlib.lines.Line2D at 0x1e7f6c9ccc0>]



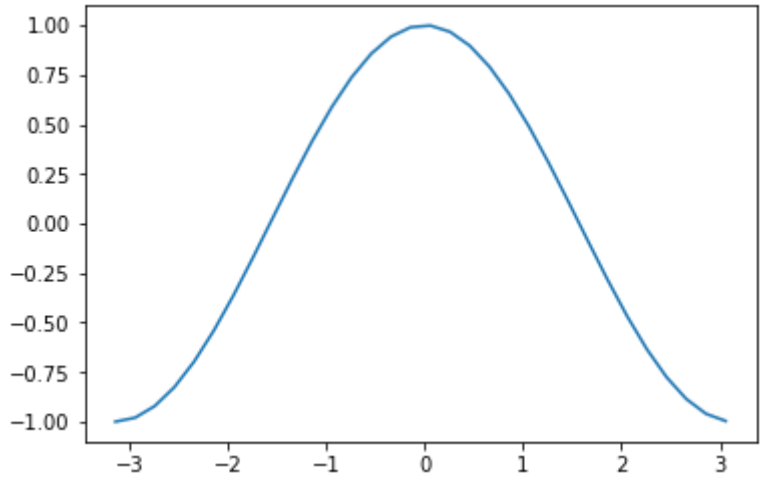
坐标轴界限

axis方法:设置x，y轴刻度值的范围

plt.axis([xmin,xmax,ymin,ymax])

In [9]:
```
plt.plot(x,y)
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x1e7f6d3ea20>]
```

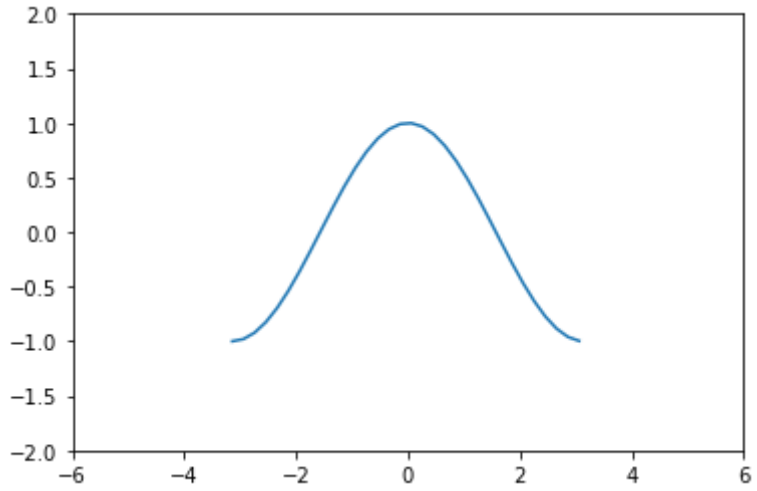


plt.axis('off')

关闭坐标轴

In [10]:
```
plt.axis([-6,6,-2,2])
plt.plot(x,y)
#plt.axis('off')
```

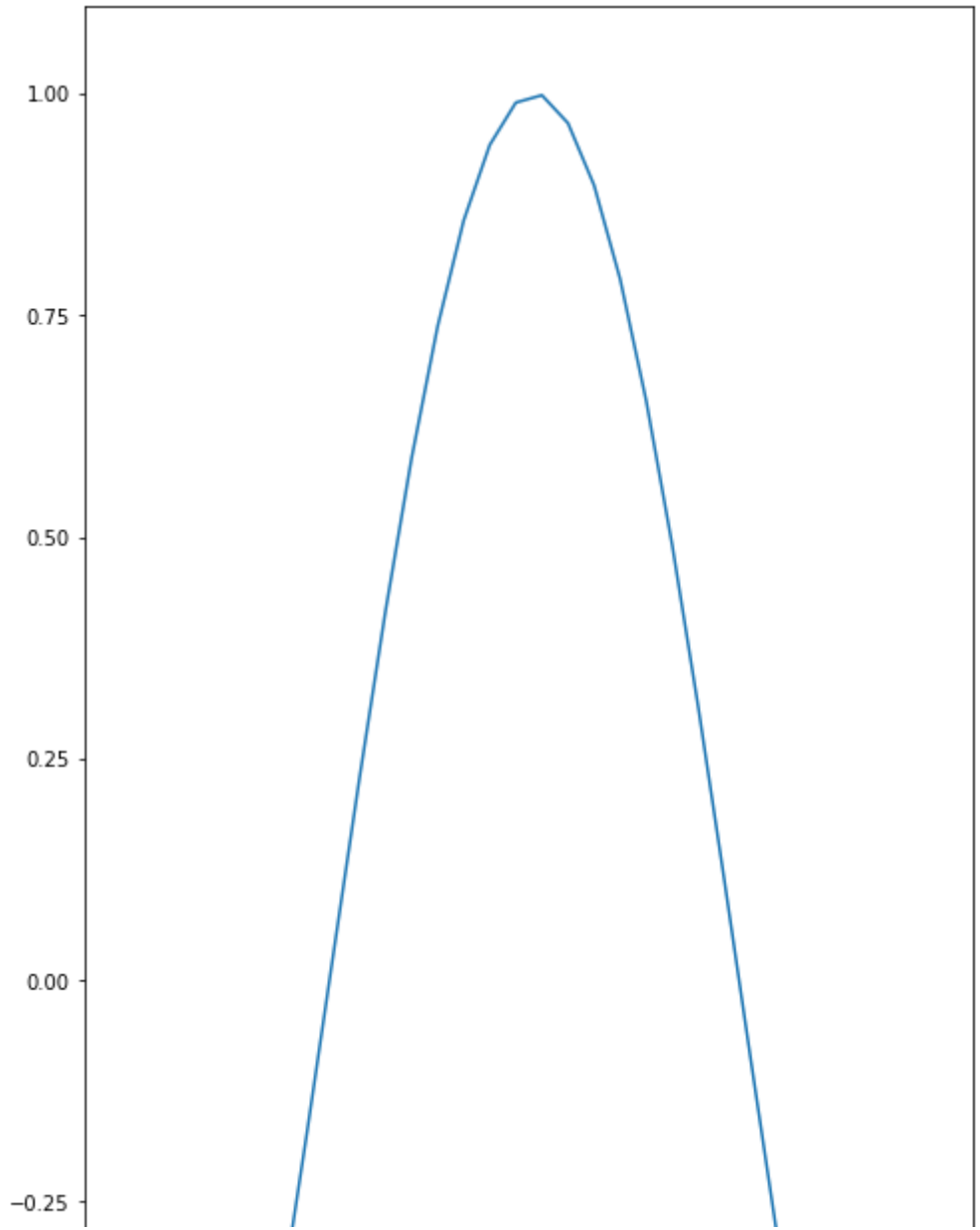
Out[10]: [<matplotlib.lines.Line2D at 0x1e7f6cefe10>]

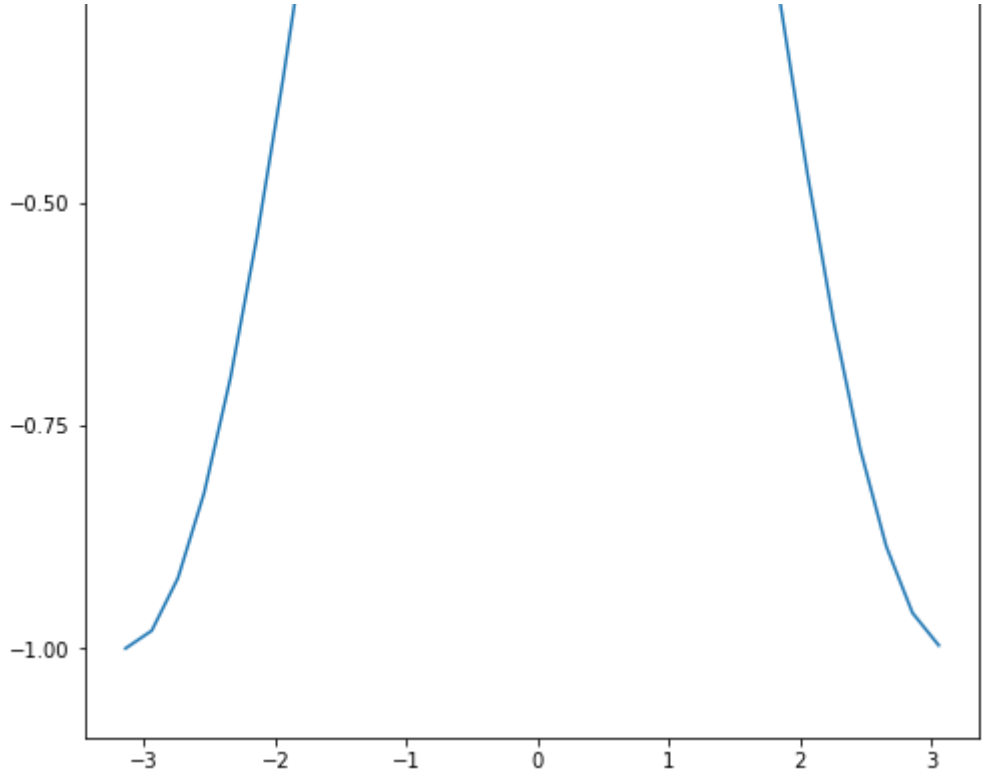


设置画布比例：plt.figure(figsize=(a,b)) a:x刻度比例 b：y刻度比例 （2:1）表示x刻度显示为y刻度显示的2倍

In [11]:
```
plt.figure(figsize=(8,18))
plt.plot(x,y)
```

Out[11]: [<matplotlib.lines.Line2D at 0x1e7f7dd1ef0>]



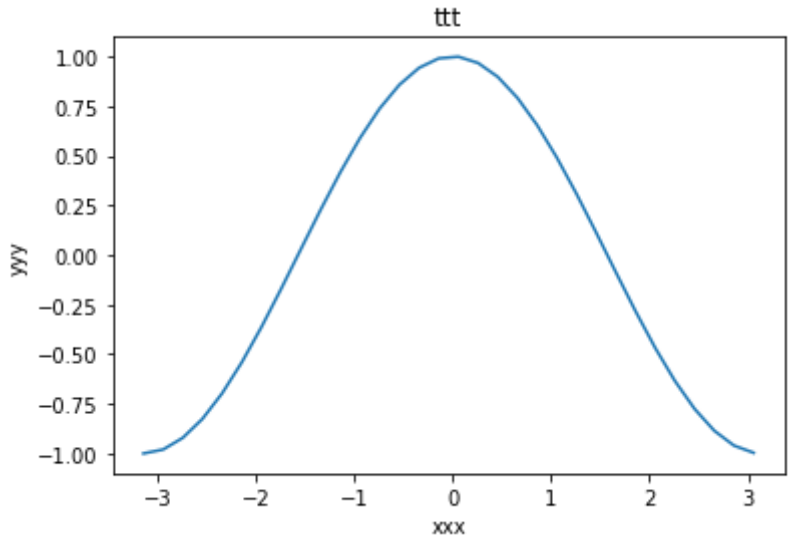


坐标轴标签

- s 标签内容
- color 标签颜色
- fontsize 字体大小
- rotation 旋转角度
- plt的xlabel方法和ylabel方法 title方法

In [12]:
```
plt.xlabel('xxx')
plt.ylabel('yyy')
plt.title('ttt')
plt.plot(x,y)
```

Out[12]: [<matplotlib.lines.Line2D at 0x1e7f8101be0>]



图例

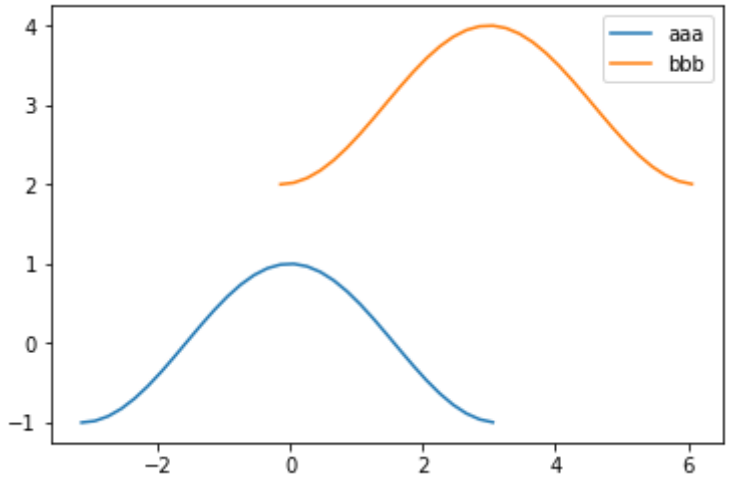
legend方法

两种传参方法:

- 分别在plot函数中增加label参数,再调用plt.legend()方法显示
- 直接在legend方法中传入字符串列表

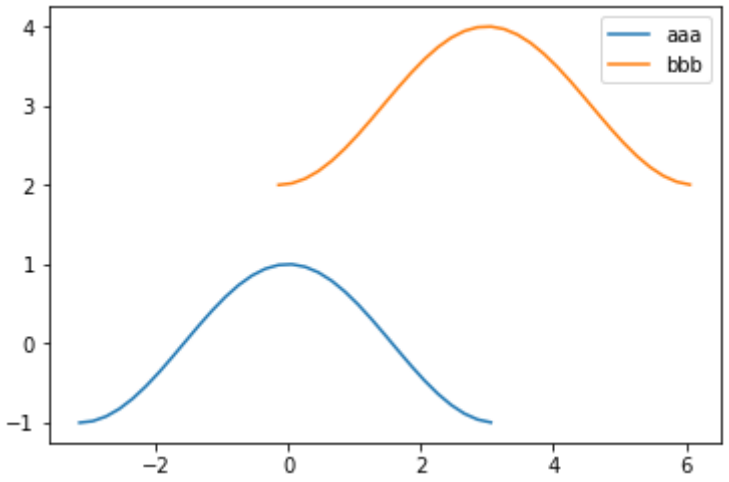
In [13]:
```
plt.plot(x,y,label='aaa')
plt.plot(x+3,y+3,label='bbb')
plt.legend()
```

Out[13]: <matplotlib.legend.Legend at 0x1e7f7e60908>



In [14]:
```
plt.plot(x,y,x+3,y+3)
plt.legend(['aaa','bbb'])
```

Out[14]: <matplotlib.legend.Legend at 0x1e7f7e98588>



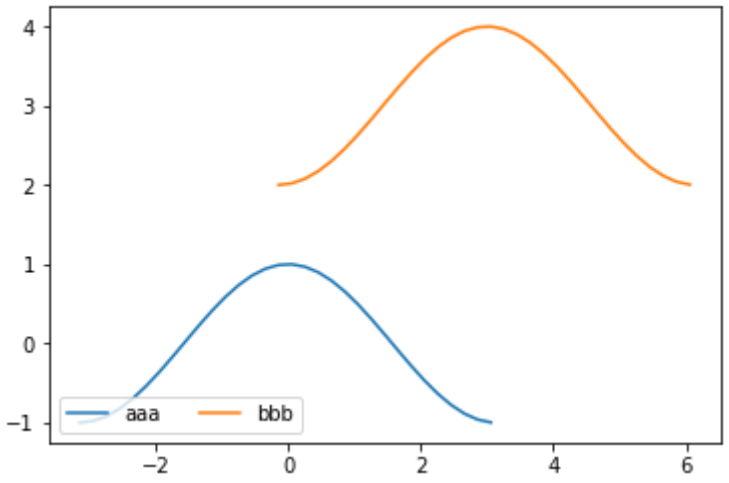
legend的参数

- loc参数

- loc参数用于设置图例标签的位置，一般在legend函数内
- matplotlib已经预定义好几种数字表示的位置

In [15]:
```
plt.plot(x,y,x+3,y+3)
plt.legend(['aaa','bbb'],loc=3,ncol=2)
```

Out[15]: <matplotlib.legend.Legend at 0x1e7f7efefd0>



字符串	数值	字符串	数值
best	0	center left	6
upper right	1	center right	7
upper left	2	lower center	8
lower left	3	upper center	9
lower right	4	center	10
right	5		

- ncol参数

ncol控制图例中有几列,在legend中设置ncol

保存图片

使用figure对象的savefig函数来保存图片

fig = plt.figure()---必须放置在绘图操作之前

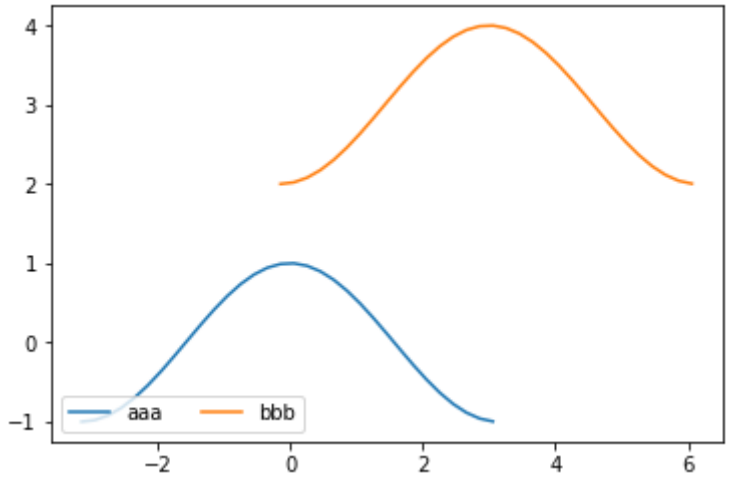
figure.savefig的参数选项

- filename
  含有文件路径的字符串或Python的文件型对象。图像格式由文件扩展名推断得出，例如，.pdf推断出PDF，.png推断出PNG（“png”、“pdf”、“svg”、“ps”、“eps”……）
- dpi
  图像分辨率（每英寸点数），默认为100
- facecolor ， 打开保存图片查看 图像的背景色，默认为“w”（白色）

In [16]:
```
fig = plt.figure()

plt.plot(x,y,x+3,y+3)
plt.legend(['aaa','bbb'],loc=3,ncol=2)

fig.savefig('./img.png',dpi=500)
```



设置plot的风格和样式

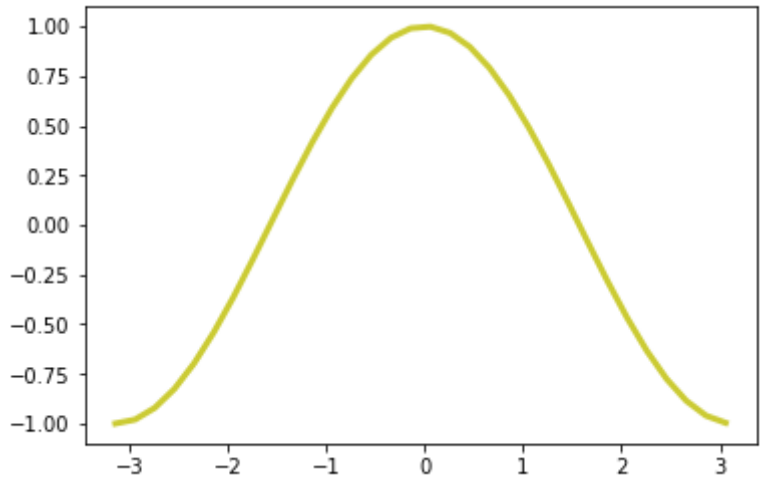
plot语句中支持除X,Y以外的参数，以字符串形式存在，来控制颜色、线型、点型等要素，语法形式为：
plt.plot(X, Y, 'format', ...)

颜色

参数color或c

In [17]:
```
plt.plot(x,y,c='y',alpha=0.8,lw=3)
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x1e7f800b6a0>]
```



颜色值的方式

- 别名
  - color='r'
- 合法的HTML颜色名
  - color = 'red'

颜色	别名	HTML颜色名	颜色	别名	HTML颜色名
蓝色	b	blue	绿色	g	green
红色	r	red	黄色	y	yellow

颜色	别名	HTML颜色名	颜色	别名	HTML颜色名
青色	c	cyan	黑色	k	black
洋红色	m	magenta	白色	w	white

- HTML十六进制字符串
  - color = '#eeefff'
- 归一化到[0, 1]的RGB元组
  - color = (0.3, 0.3, 0.4)

透明度

alpha参数

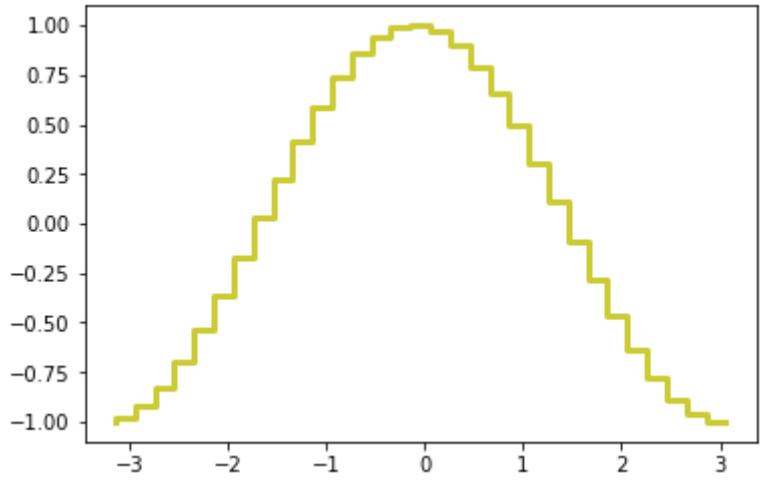
线型

参数linestyle或ls

线条风格	描述	线条风格	描述
'-'	实线	':'	虚线
'--'	破折线	'steps'	阶梯线
'-.'	点划线	'None' / ' ' / ''	什么都不画

In [18]:
```
plt.plot(x,y,c='y',alpha=0.8,lw=3,ls='steps')
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x1e7f8067da0>]
```



线宽

linewidth或lw参数

点型

- marker 设置点形
- markersize 设置点形大小

标记	描述	标记	描述
's'	正方形	'p'	五边形
'h'	六边形1	'H'	六边形2
'8'	八边形		

标记	描述	标记	描述

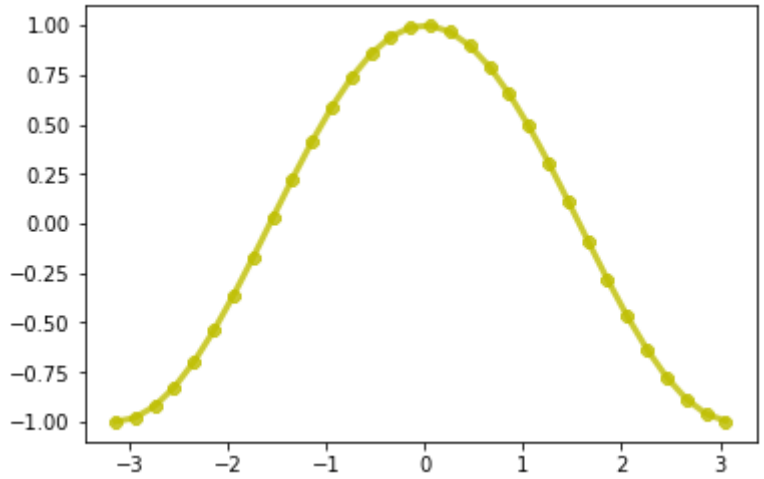
标记	描述	标记	描述
'.'	点	'x'	X
'*'	星号	'+'	加号
','	像素		

标记	描述	标记	描述
'o'	圆圈	'D'	菱形
'd'	小菱形	'','None',' ',None	无

标记	描述	标记	描述
'1'	一角朝下的三脚架	'3'	一角朝左的三脚架
'2'	一角朝上的三脚架	'4'	一角朝右的三脚架

In [19]:
```
plt.plot(x,y,c='y',alpha = 0.8,lw=3,marker='8')
```

Out[19]: [<matplotlib.lines.Line2D at 0x1e7f80ca588>]



In [20]:
```
# 绘制线       plt.plot(x1,y1,x2,y2)
# 网格线       plt.grid(True)  axes.grid(color,ls,lw,alpha)
# 获取坐标系   plt.subplot(n1,n2,n3)
# 坐标轴标签   plt.xlabel() plt.ylabel()
# 坐标系标题   plt.title()
# 图例         plt.legend([names],ncol=2,loc=1)  plt.plot(label='name')
# 线风格       --  -.  :  None  step
# 图片保存     figure.savefig()
# 点的设置     marker markersize markerfacecolor markeredgecolor\width
# 坐标轴刻度   plt.xticks(刻度列表，刻度标签列表) plt.yticks()
#              axes.set_xticks(刻度列表) axes.set_xticklabels(刻度标签列表)
```

三、2D图形

直方图

- 是一个特殊的柱状图，又叫做密度图。

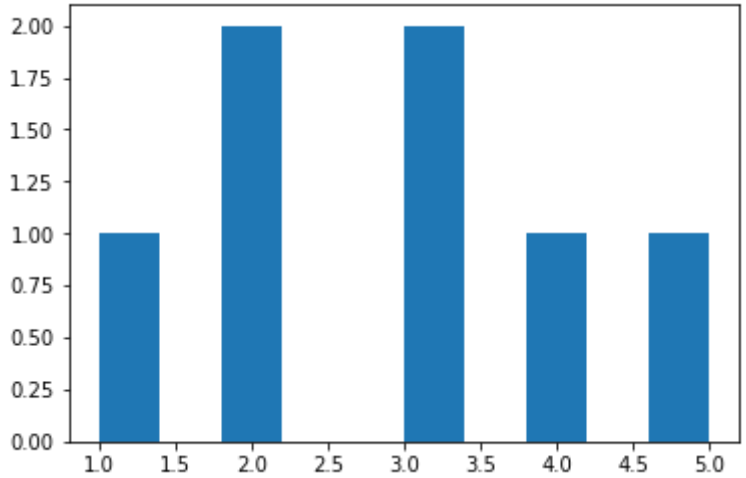
【直方图的参数只有一个x！！！不像条形图需要传入x,y】

plt.hist()的参数

- bins
  直方图的柱数，可选项，默认为10
- color
  指定直方图的颜色。可以是单一颜色值或颜色的序列。如果指定了多个数据集合,例如DataFrame对象，颜色序列将会设置为相同的顺序。如果未指定，将会使用一个默认的线条颜色
- orientation
  通过设置orientation为horizontal创建水平直方图。默认值为vertical

In [21]:
```
data = [1,2,3,3,4,2,5]
plt.hist(data,bins=10)
```

Out[21]:
```
(array([1., 0., 2., 0., 0., 2., 0., 1., 0., 1.]),
 array([1. , 1.4, 1.8, 2.2, 2.6, 3. , 3.4, 3.8, 4.2, 4.6, 5. ]),
 <a list of 10 Patch objects>)
```



返回值：

1: 直方图向量，是否归一化由参数normed设定

2: 返回各个bin的区间范围

3: 返回每个bin里面包含的数据，是一个list

条形图：plt.bar()

- 参数：第一个参数是索引。第二个参数是数据值。第三个参数是条形的宽度

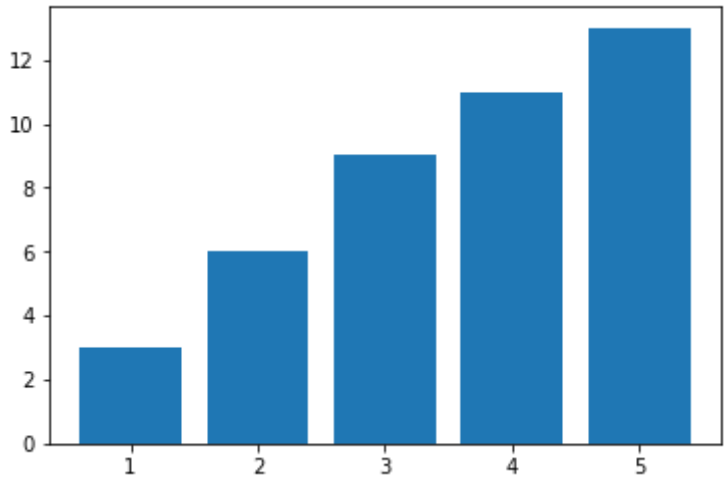
-【条形图有两个参数x,y】

- width 纵向设置条形宽度
- height 横向设置条形高度

bar()、barh()

In [22]:
```
x = [1,2,3,4,5]
y = [3,6,9,11,13]
plt.bar(x,y)
```

Out[22]: <BarContainer object of 5 artists>

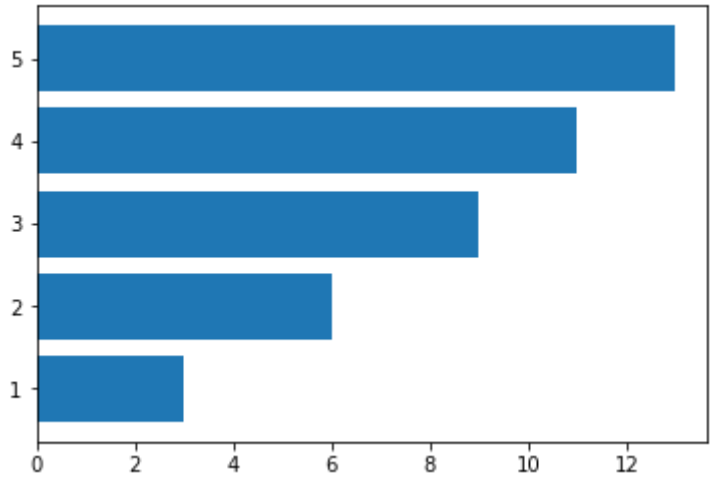


水平条形图

barh()

In [23]:
```
plt.barh(x,y)
```

Out[23]: <BarContainer object of 5 artists>



饼图

【饼图也只有一个参数x】

pie()
饼图适合展示各部分占总体的比例，条形图适合比较各部分的大小

普通各部分占满饼图

In [24]:
```
plt.pie([11,22,33])
```

Out[24]:
```
([<matplotlib.patches.Wedge at 0x1e7f84e2ef0>,
  <matplotlib.patches.Wedge at 0x1e7f84eb438>,
  <matplotlib.patches.Wedge at 0x1e7f84eb908>],
 [Text(0.9526279355804298, 0.5500000148652441, ''),
  Text(-0.5500000594609755, 0.9526279098330699, ''),
  Text(1.0298943251329445e-07, -1.0999999999999954, '')])
```

In [25]:
```
plt.pie([0.2,0.3,0.1])
```

Out[25]:
```
([<matplotlib.patches.Wedge at 0x1e7f8520eb8>,
  <matplotlib.patches.Wedge at 0x1e7f85283c8>,
  <matplotlib.patches.Wedge at 0x1e7f8528898>],
 [Text(0.8899186877588753, 0.6465637858537406, ''),
  Text(-0.64656382751384, 0.8899186574910392, ''),
  Text(-1.0461621345079049, -0.3399187966586502, '')])
```

普通未占满饼图：小数/比例

饼图阴影、分裂等属性设置

#labels参数设置每一块的标签;

#labeldistance参数设置标签距离圆心的距离（比例值）

#autopct参数设置比例值小数保留位(%.3f%%);

#pctdistance参数设置比例值文字距离圆心的距离

#explode参数设置每一块顶点距圆心的长度（比例值,列表）；

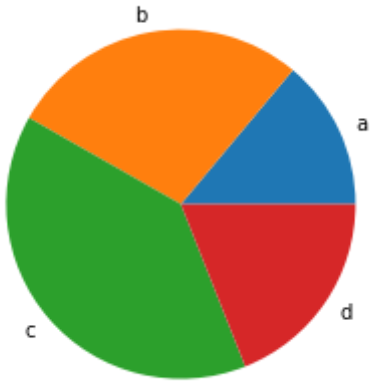
#colors参数设置每一块的颜色（列表）；

#shadow参数为布尔值，设置是否绘制阴影

#startangle参数设置饼图起始角度

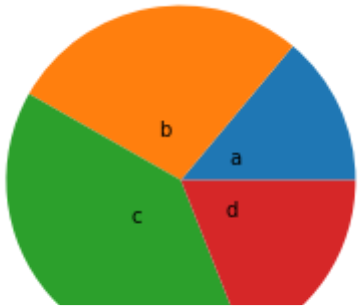
In [26]:
```
arr = [11,22,31,15]
plt.pie(arr,labels=['a','b','c','d'])
```

Out[26]:
```
([<matplotlib.patches.Wedge at 0x1e7f85660b8>,
  <matplotlib.patches.Wedge at 0x1e7f85665c0>,
  <matplotlib.patches.Wedge at 0x1e7f8566a90>,
  <matplotlib.patches.Wedge at 0x1e7f8566f60>],
 [Text(0.9964244374501308, 0.46598105160209097, 'a'),
  Text(-0.19579764419425466, 1.0824339622018426, 'b'),
  Text(-0.830021124093439, -0.7218482759961848, 'c'),
  Text(0.9100343885615038, -0.617929940717789, 'd')])
```



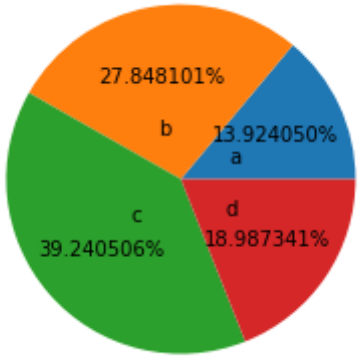
In [27]:
```
##labeldistance参数设置标签距离圆心的距离（比例值）
arr = [11,22,31,15]
plt.pie(arr,labels=['a','b','c','d'],labeldistance=0.3)
```

Out[27]:
```
([<matplotlib.patches.Wedge at 0x1e7f85a1588>,
  <matplotlib.patches.Wedge at 0x1e7f85a1a90>,
  <matplotlib.patches.Wedge at 0x1e7f85a1f60>,
  <matplotlib.patches.Wedge at 0x1e7f85a9470>],
 [Text(0.2717521193045811, 0.1270857413460248, 'a'),
  Text(-0.05339935750752399, 0.29520926241868434, 'b'),
  Text(-0.2263693974800288, -0.1968677116353231, 'c'),
  Text(0.24819119688041008, -0.16852634746848788, 'd')])
```



In [28]:
```
#autopct参数设置比例值小数保留位(%.3f%%);
arr = [11,22,31,15]
plt.pie(arr,labels=['a','b','c','d'],labeldistance=0.3,autopct='%.6f%%')
```

Out[28]:
```
([<matplotlib.patches.Wedge at 0x1e7f85ddb00>,
  <matplotlib.patches.Wedge at 0x1e7f85e7240>,
  <matplotlib.patches.Wedge at 0x1e7f85e7908>,
  <matplotlib.patches.Wedge at 0x1e7f85e7fd0>],
 [Text(0.2717521193045811, 0.1270857413460248, 'a'),
  Text(-0.05339935750752399, 0.29520926241868434, 'b'),
  Text(-0.2263693974800288, -0.1968677116353231, 'c'),
  Text(0.24819119688041008, -0.16852634746848788, 'd')],
 [Text(0.5435042386091622, 0.2541714826920496, '13.924050%'),
  Text(-0.10679871501504798, 0.5904185248373687, '27.848101%'),
  Text(-0.4527387949600576, -0.3937354232706462, '39.240506%'),
  Text(0.49638239376082016, -0.33705269493697576, '18.987341%')])
```

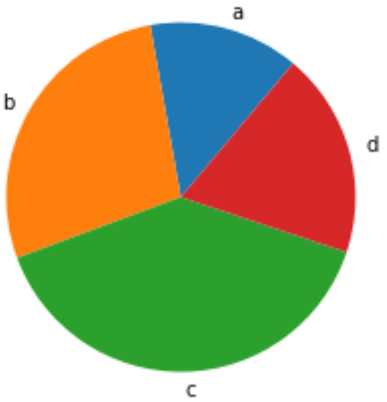


In [29]:
```
##explode参数设置每一块顶点距圆心的长度（比例值,列表）；
arr = [11,22,31,15]
plt.pie(arr,labels=['a','b','c','d'],labeldistance=0.3,shadow=True,explode=[0.2,0.3,0.2,0.4])
```

Out[29]:
```
([<matplotlib.patches.Wedge at 0x1e7f86228d0>,
  <matplotlib.patches.Wedge at 0x1e7f862a080>,
  <matplotlib.patches.Wedge at 0x1e7f862a7f0>,
  <matplotlib.patches.Wedge at 0x1e7f862af60>],
 [Text(0.4529201988409685, 0.21180956891004132, 'a'),
  Text(-0.10679871501504798, 0.5904185248373687, 'b'),
  Text(-0.37728232913338133, -0.32811285272553853, 'c'),
  Text(0.579112792720957, -0.3932281440931384, 'd')])
```

In [30]:
```
#startangle参数设置饼图起始角度
arr = [11,22,31,15]
plt.pie(arr,labels=['a','b','c','d'],startangle=50)
```

Out[30]:
```
([<matplotlib.patches.Wedge at 0x1e7f866f5f8>,
  <matplotlib.patches.Wedge at 0x1e7f866fb00>,
  <matplotlib.patches.Wedge at 0x1e7f866ffd0>,
  <matplotlib.patches.Wedge at 0x1e7f86774e0>],
 [Text(0.2835270872033046, 1.0628322496151543, 'a'),
  Text(-0.9550488214818597, 0.5457854418964573, 'b'),
  Text(0.01944056625653503, -1.0998281976670836, 'c'),
  Text(1.058320626679702, 0.2999290768569842, 'd')])
```



%m.nf
m 占位
n 小数点后保留几位
f 是以float格式输出

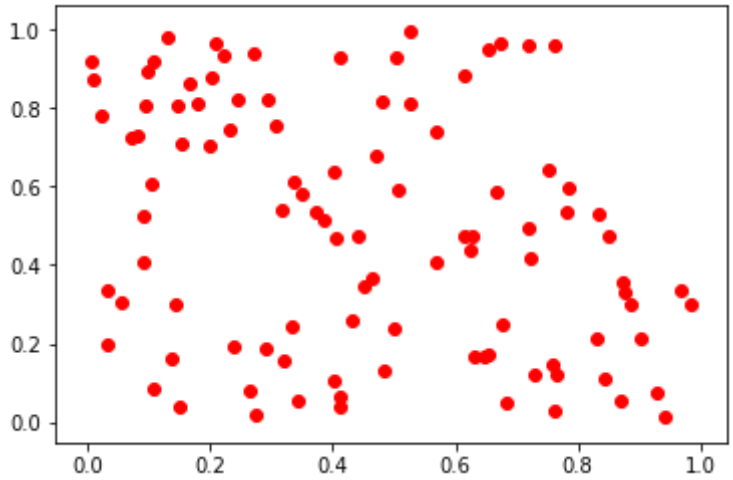
散点图：因变量随自变量而变化的大致趋势

【散点图需要两个参数x,y，但此时x不是表示x轴的刻度，而是每个点的横坐标！】

scatter()

In [31]:
```
x = np.random.random(size=(100,))
y = np.random.random(size=(100,))
plt.scatter(x,y,c='r')
```

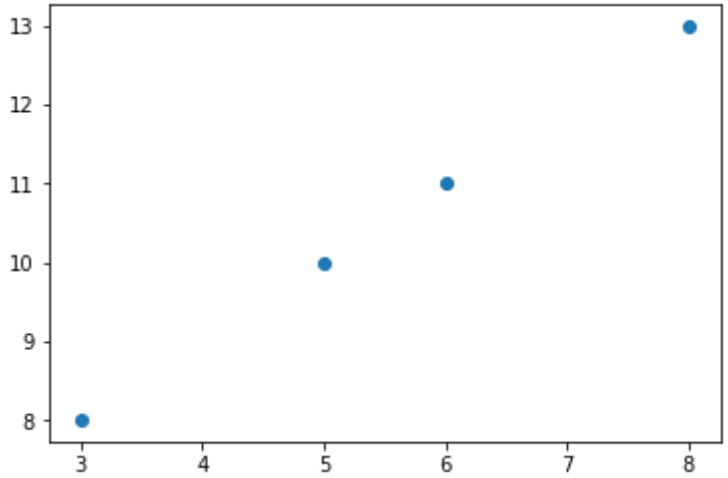
Out[31]: <matplotlib.collections.PathCollection at 0x1e7f86a8cf8>



In [32]:
```
x = np.array([3,6,8,5])
y = x + 5

plt.scatter(x,y)
```

Out[32]: <matplotlib.collections.PathCollection at 0x1e7f870eba8>



In [ ]: