**Plt.scatter() :**

catter散点图，函数的原型：

matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, \*, data=None, \*\*kwargs)

参数的解释：

x，y：表示的是大小为(n,)的数组，也就是我们即将绘制散点图的数据点

s:是一个实数或者是一个数组大小为(n,)，这个是一个可选的参数。

c:表示的是颜色，也是一个可选项。默认是蓝色'b',表示的是标记的颜色，或者可以是一个表示颜色的字符，或者是一个长度为n的表示颜色的序列等等，感觉还没用到过现在不解释了。但是c不可以是一个单独的RGB数字，也不可以是一个RGBA的序列。可以是他们的2维数组（只有一行）。

arker:表示的是标记的样式，默认的是'o'。

cmap:Colormap实体或者是一个colormap的名字，cmap仅仅当c是一个浮点数数组的时候才使用。如果没有申明就是image.cmap

norm:Normalize实体来将数据亮度转化到0-1之间，也是只有c是一个浮点数的数组的时候才使用。如果没有申明，就是默认为colors.Normalize。

vmin,vmax:实数，当norm存在的时候忽略。用来进行亮度数据的归一化。

alpha：实数，0-1之间。

linewidths:也就是标记点的长度。

import numpy as np

import matplotlib.pyplot as plt

np.random.seed(1)

x=np.random.rand(10)

y=np.random.rand(10)

colors=np.random.rand(10)

area=(30\*np.random.rand(10))\*\*2

plt.scatter(x,y,s=area,c=colors,alpha=0.5)

plt.show()

**np.array**

**为什么要用numpy**

    Python中提供了list容器，可以当作数组使用。但列表中的元素可以是任何对象，因此列表中保存的是对象的指针，这样一来，为了保存一个简单的列表[1,2,3]。就需要三个指针和三个整数对象。对于数值运算来说，这种结构显然不够高效。  
    Python虽然也提供了array模块，但其只支持一维数组，不支持**多维数组**(在TensorFlow里面偏向于矩阵理解)，也没有各种运算函数。因而不适合数值运算。  
    NumPy的出现弥补了这些不足。

**import numpy as np**

数组创建

## 常规创建方法  
a = np.array([2,3,4])  
b = np.array([2.0,3.0,4.0])  
c = np.array([[1.0,2.0],[3.0,4.0]])  
d = np.array([[1,2],[3,4]],dtype=complex) # 指定数据类型

**numpy.reshape**

这个方法是在不改变数据内容的情况下，改变一个数组的格式，参数及返回值，

**numpy.cov协方差矩阵计算**

在概率论和统计学中，协方差用于衡量两个变量的总体误差。而方差是协方差的一种特殊情况，即当两个变量是相同的情况。其定义的数学形式是：C o v ( X , Y ) = E [ ( X − E ( X ) ) ( Y − E ( Y ) ) ] = E [ X Y ] − E [ X ] E [ Y ] Cov(X,Y)=E[(X-E(X))(Y-E(Y))] =E[XY]-E[X]E[Y]*Cov*(*X*,*Y*)=*E*[(*X*−*E*(*X*))(*Y*−*E*(*Y*))]=*E*[*XY*]−*E*[*X*]*E*[*Y*] <https://blog.csdn.net/jeffery0207/article/details/83032325>

[**Python中Numpy mat的使用**](https://www.cnblogs.com/lfri/p/10561914.html)

前面介绍过用dnarray来模拟，但mat更符合矩阵，这里的mat与Matlab中的很相似。(mat与matrix等同)