In [64]:

```
import numpy as np
```

多维数组ndarray

利用array创建

In [65]:

```
newarray=np.array([[1,2,3,4],[2,3,4,5]]) #两个[]
newarray
```

Out[65]:

```
array([[1, 2, 3, 4],
       [2, 3, 4, 5]])
```

其他函数创建

In [66]:

```
np.zeros((3,3))  #两个括号
```

Out[66]:

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [67]:

```
np.ones((3,3))
```

Out[67]:

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [68]:

```
np.arange(1,10,2) #1到10等差为2的等差数列
```

Out[68]:

```
array([1, 3, 5, 7, 9])
```

In [69]:

```
np.linspace(1,10,5) #1到10,5个数
```

Out[69]:

```
array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

ndarray的数据类型转化

数据类型查阅

In [70]:

```
arr2=np.array([[1.0,2.0,3.0,4.0],[3.2,3.5,6.5,4.5]])
arr2.dtype
# int32 float64 <U1
```

Out[70]:

```
dtype('float64')
```

数据类型转化

In [71]:

```
int_arr2=arr2.astype(np.int32)
int_arr2
```

Out[71]:

```
array([[1, 2, 3, 4],
       [3, 3, 6, 4]])
```

In [72]:

```
int_arr3=arr2.astype(np.unicode_)
int_arr3
```

Out[72]:

```
array([['1.0', '2.0', '3.0', '4.0'],
       ['3.2', '3.5', '6.5', '4.5']], dtype='<U32')
```

数据索引，切片，赋值

In [73]:

```
#两个注意点 python从零开始计数，[1:3]指2和3行 前后都要减一
```

In [74]:

```
#将一个值赋值给一个切片时，该值会在整个切片上，并直接对原数据组进行修改
newarray[:1,:2]=10
newarray
```

Out[74]:

```
array([[10, 10,  3,  4],
       [ 2,  3,  4,  5]])
```

In [75]:

```
#如果避免修改原数据组 用copy函数
```

In [76]:

```
newarray.copy()[:1,:2]=100
newarray
```

Out[76]:

```
array([[10, 10,  3,  4],
       [ 2,  3,  4,  5]])
```

基本的数组运算

In [77]:

```
# array1 * array2   是指各项相乘
# array1.dot(array2)   是指矩阵相乘
newarray.dot(newarray.T)
```

Out[77]:

```
array([[225,  82],
       [ 82,  54]])
```

In [78]:

```
# 逆矩阵 lg.inv(a)
import numpy.linalg as lg
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(lg.inv(a))
```

```
[[ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]
 [-6.30503948e+15  1.26100790e+16 -6.30503948e+15]
 [ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]]
```

In [79]:

```
# 转置矩阵 a.T
print(newarray.T)
```

```
[[10  2]
 [10  3]
 [ 3  4]
 [ 4  5]]
```

矩阵信息获取

最大值最小值

In [80]:

```
a = np.array([[1,2,3],[4,5,6]])
print(a.max()) #获取整个矩阵的最大值 结果： 6
print(a.min()) #结果：1
```

```
6
1
```

In [81]:

```
# 可以指定关键字参数axis来获得行最大（小）值或列最大（小）值
# axis=0 行方向最大（小）值，即获得每列的最大（小）值
# axis=1 列方向最大（小）值，即获得每行的最大（小）值
# 例如
print(a.max(axis=0))
print(a.max(axis=1))
```

```
[4 5 6]
[3 6]
```

In [82]:

```
# 要想获得最大最小值元素所在的位置，可以通过argmax函数来获得
print(a.argmax(axis=1))
```

```
[2 2]
```

平均值 方差 标准差

In [83]:

```
print(a.mean())
# 同样地，可以通过关键字axis参数指定沿哪个方向获取平均值
print(a.mean(axis=0))
print(a.mean(axis=1))
#方差 .var   标注差 .st  中间值mean
```

```
3.5
[2.5 3.5 4.5]
[2. 5.]
```

中间值

In [84]:

```
#和其他的形式不同 变量在括号内
```

In [85]:

```
np.median(a)
```

Out[85]:

```
3.5
```

求和 累计求和

In [86]:

```
print(a.sum())            # 对整个矩阵求和
print(a.sum(axis=0)) # 对行方向求和
print(a.sum(axis=1)) # 对列方向求和
```

```
21
[5 7 9]
[ 6 15]
```

In [87]:

```
print(a.cumsum())            # 对整个矩阵累积求和
print(a.cumsum(axis=0)) # 对行方向累积求和
print(a.cumsum(axis=1)) # 对列方向累积求和
```

```
[ 1  3  6 10 15 21]
[[1 2 3]
 [5 7 9]]
[[ 1  3  6]
 [ 4  9 15]]
```

随机数 random

In [88]:

```
import numpy.random as npr
```

rand 随机多维数组 [0,1]

In [89]:

```
npr.rand(3,4)
```

Out[89]:

```
array([[0.90844877, 0.64149265, 0.99619093, 0.30439803],
       [0.0075783 , 0.8785172 , 0.9967626 , 0.91280385],
       [0.01204759, 0.39790881, 0.25736074, 0.56987945]])
```

In [90]:

```
#将区间进行转化 [2,4]
npr.rand(3,4)*(4-2)+2
```

Out[90]:

```
array([[3.51820837, 3.44842538, 3.09273309, 3.59638815],
       [3.50014796, 2.90177237, 2.09309359, 3.5814059 ],
       [2.97181975, 3.15493926, 2.9021559 , 3.2061316 ]])
```

其他随机数

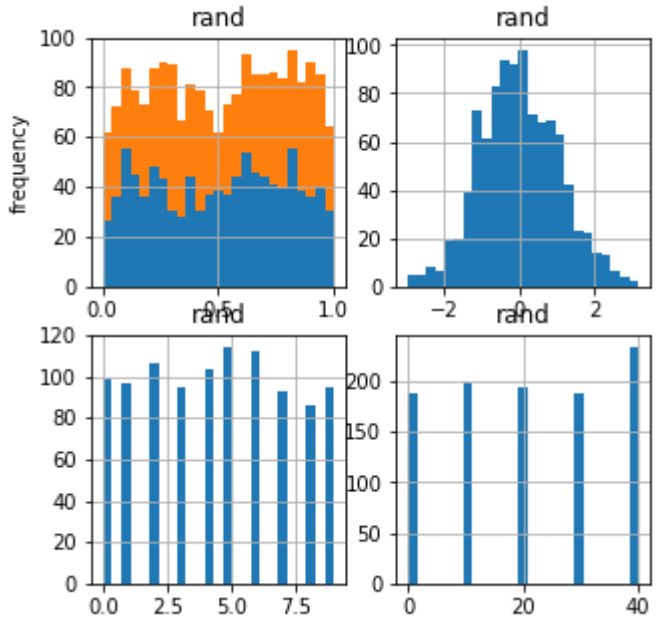
In [91]:

```
size=1000
rn1=npr.rand(size,2)
rn2=npr.randn(size) #标准正态分布
rn3=npr.randint(0,10,size) #随机整数
rang=[0,10,20,30,40]
#生成给定一维数组的随机样本
rn4=npr.choice(rang,size=size)
```

可视化

In [92]:

```
import matplotlib.pyplot as plt
#设定总体图形样式
fig,((ax1,ax2),(ax3,ax4))=plt.subplots(nrows=2,ncols=2,figsize=(5,5))
# 每个图形进行操
ax1.hist(rn1,bins=25,stacked=True)
#stacked 是指重叠
ax1.set_title("rand")
ax1.set_ylabel("frequency")
ax1.grid(True) #显示网格
ax2.hist(rn2,bins=25)
ax2.set_title("rand")
ax2.grid(True) #显示网格
ax3.hist(rn3,bins=25)
ax3.set_title("rand")
ax3.grid(True) #显示网格
ax4.hist(rn4,bins=25)
ax4.set_title("rand")
ax4.grid(True) #显示网格
```



分布的随机数

In [93]:

```
rn5=npr.binomial(100,0.3,size)  #二项分布
rn6=npr.normal(10,36,size) #正态分布
rn7=npr.chisquare(0.5,size) #卡方分布 自由度
rn8=npr.poisson(2.0,size)  #泊松分布
```

In [94]:

```
fig,((ax1,ax2),(ax3,ax4))=plt.subplots(nrows=2,ncols=2,figsize=(5,5))
# 每个图形进行操
ax1.hist(rn5,bins=25)
#stacked 是指重叠
ax1.set_title("rand")
ax1.set_ylabel("frequency")
ax1.grid(True) #显示网格
ax2.hist(rn6,bins=25)
ax2.set_title("rand")
ax2.grid(True) #显示网格
ax3.hist(rn7,bins=25)
ax3.set_title("rand")
ax3.grid(True) #显示网格
ax4.hist(rn8,bins=25)
ax4.set_title("rand")
ax4.grid(True) #显示网格
```