```
import numpy as np
```

随机函数

NumPy中也有自己的随机函数，包含在random模块中。它能产生特定分布的随机数，如正态分布等。接下来介绍一些常用的随机数。

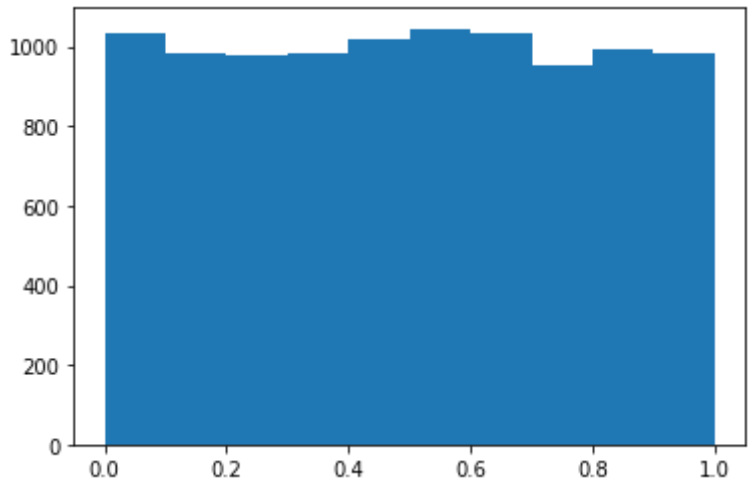
函数名	功能	参数使用(int a,b,c,d)
rand(int1,[int2,[int3,]])	生成(0,1)均匀分布随机数	(a)，(a,b)，(a,b,c)
randn(int1,[int2,[int3,]])	生成标准正态分布随机数	(a)，(a,b)，(a,b,c)
randint(low[,hight,size,dtype])	生成随机整数	(a,b)，(a,b,c)，(a,b,(c,d))
sample(size)	生成[0,1)随机数	(a)，((a,b))，((a,b,c))

numpy.random.rand(d0,d1,…,dn)

- rand函数根据给定维度生成[0,1)之间的数据，包含0，不包含1
- dn表示每个维度
- 返回值为指定维度的array

```
from matplotlib import pyplot as plt
# 创建4行2列的随机数据
a = np.random.rand(10000)
plt.hist(a)
```

```
(array([1035.,  985.,  977.,  985., 1016., 1045., 1031.,  951.,
992.,
        983.]),
 array([2.20297135e-05, 1.00016931e-01, 2.00011832e-01,
3.00006733e-01,
       4.00001634e-01, 4.99996536e-01, 5.99991437e-01,
6.99986338e-01,
       7.99981239e-01, 8.99976140e-01, 9.99971042e-01]),
 <a list of 10 Patch objects>)
```




```
# 创建2块2行3列的随机数据
np.random.rand(2,2,3)
```

```
array([[[0.20791139, 0.3859659 , 0.09974949],
        [0.97522703, 0.86420763, 0.5905462 ]],

       [[0.07021044, 0.32345555, 0.24879223],
        [0.040974  , 0.69146699, 0.8334221 ]]])
```

numpy.random.randn(d0,d1,…,dn)

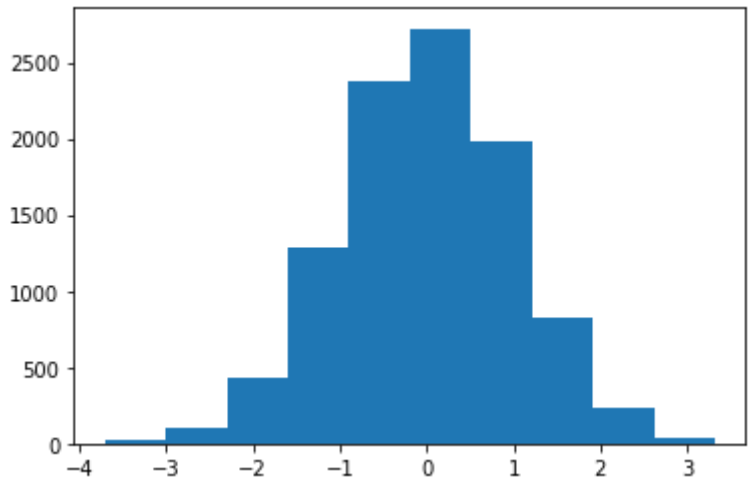
- randn函数返回一个或一组样本，具有标准正态分布。
- dn表示每个维度
- 返回值为指定维度的array

标准正态分布又称为u分布，是以0为均值、以1为标准差的正态分布，记为N（0，1）。

```
from matplotlib import pyplot as plt
a = np.random.randn(10000)
print(a)
# 直方图
plt.hist(a)
```

```
[-0.16890737  0.37986348  1.84458246 ...  0.25121365  1.10885655
 -0.92354789]
```

```
(array([  21.,  102.,  427., 1286., 2371., 2721., 1979.,  826.,  235.,
         32.]),
 array([-3.70755966, -3.00492333, -2.30228701, -1.59965068, -0.89701436,
       -0.19437803,  0.50825829,  1.21089462,  1.91353094,  2.61616727,
        3.31880359]),
 <a list of 10 Patch objects>)
```



numpy.random.randint()

`numpy.random.randint(low, high=None, size=None, dtype='l')`

- 返回随机整数，范围区间为[low,high），包含low，不包含high

- 参数：low为最小值，high为最大值，size为数组维度大小，dtype为数据类型，默认的数据类型是np.int
- high没有填写时，默认生成随机数的范围是[0，low)

```
# 返回[0,1)之间的整数，所以只有0
np.random.randint(2,size=5)
```

```
array([1, 1, 0, 1, 0])
```

```
# 返回1个[1,5)时间的随机整数
np.random.randint(1,5)
```

```
3
```

```
# 返回 -5到5之间不包含5的 2行2列数据
np.random.randint(-5,5,size=(2,2))

```

```
array([[ 3, -1],
       [ 2, -2]])
```

```
#返回 -5到5之间不包含5的 2块3行4列的随机整数
```

numpy.random.sample

`numpy.random.sample(size=None)`

返回半开区间内的随机浮点数[0.0，1.0]。

```
np.random.sample((2,3))
```

```
array([[0.83353533, 0.36751274, 0.27298122],
       [0.76358457, 0.96365424, 0.36947203]])
```

随机种子np.random.seed()

使用相同的seed()值，则每次生成的随机数都相同,使得随机数可以预测

但是，只在调用的时候seed()一下并不能使生成的随机数相同，需要每次调用都seed()一下，表示种子相同，从而生成的随机数相同。

```

np.random.seed(1)
L1 = np.random.randn(3, 3)

L2 = np.random.randn(3, 3)
print(L1)
print("-"*10)
print(L2)
```

```
[[ 1.62434536 -0.61175641 -0.52817175]
 [-1.07296862  0.86540763 -2.3015387 ]
 [ 1.74481176 -0.7612069   0.3190391 ]]
----------
[[-0.24937038  1.46210794 -2.06014071]
 [-0.3224172  -0.38405435  1.13376944]
 [-1.09989127 -0.17242821 -0.87785842]]
```

```
np.random.seed(1)

L1 = np.random.randn(3, 3)

np.random.seed(1)
L2 = np.random.randn(3, 3)*1000
print(L1)
print("-"*10)
print(L2)
```

```
[[ 1.62434536 -0.61175641 -0.52817175]
 [-1.07296862  0.86540763 -2.3015387 ]
 [ 1.74481176 -0.7612069   0.3190391 ]]
----------
[[ 1624.34536366  -611.75641365  -528.17175226]
 [-1072.96862216   865.40762932 -2301.53869688]
 [ 1744.81176422  -761.2069009    319.03909606]]
```

正态分布 numpy.random.normal

`numpy.random.normal(loc=0.0, scale=1.0, size=None)`

作用：返回一个由size指定形状的数组，数组中的值服从 μ=loc,σ=scale 的正态分布。

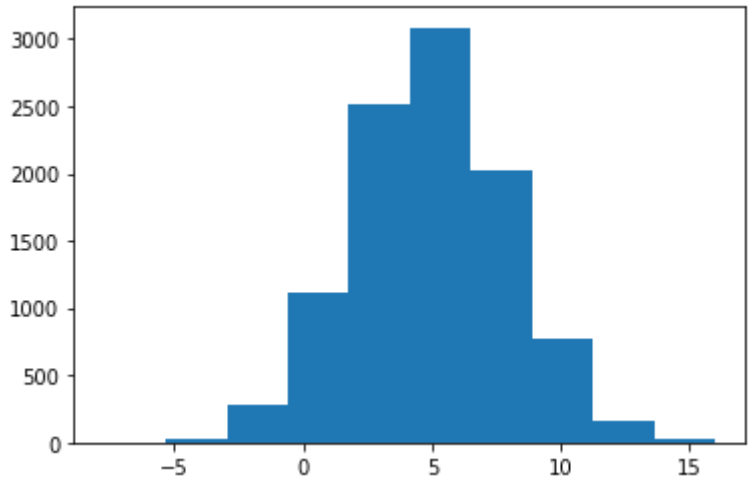
参数：

- loc : float型或者float型的类数组对象，指定均值 μ
- scale : float型或者float型的类数组对象，指定标准差 σ
- size : int型或者int型的元组，指定了数组的形状。如果不提供size，且loc和scale为标量（不是类数组对象），则返回一个服从该分布的随机数。

```
# 标准正态分布，3行2列
a = np.random.normal(0, 1, (3, 2))
print(a)
print('-'*20)
# 均值为1，标准差为3
import matplotlib.pyplot as plt
b = np.random.normal(5, 3, (10000,))
print(b)
plt.hist(b)
```

```
[[ 0.23072323 -2.14424542]
 [-1.00790645  0.40333881]
 [ 0.69803359 -0.07887092]]
--------------------
[5.30067732 8.85101023 6.41674918 ... 2.62397308 3.06163177
5.93090075]
```

```
(array([   4.,   28.,  276., 1118., 2513., 3084., 2019.,  778.,
159.,
         21.]),
 array([-7.69949439, -5.3316152 , -2.96373601, -0.59585682,
1.77202237,
        4.13990157,  6.50778076,  8.87565995, 11.24353914,
13.61141833,
       15.97929753]),
 <a list of 10 Patch objects>)
```



数组的其他函数

主要有以下方法:

函数名称	描述说明
resize	返回指定形状的新数组。
append	将元素值添加到数组的末尾。
insert	沿规定的轴将元素值插入到指定的元素前。
delete	删掉某个轴上的子数组，并返回删除后的新数组。
argwhere	返回数组内符合条件的元素的索引值。
unique	用于删除数组中重复的元素，并按元素值由大到小返回一个新数组。
sort()	对输入数组执行排序，并返回一个数组副本
argsort	沿着指定的轴，对输入数组的元素值进行排序，并返回排序后的元素索引数组

numpy.resize()

`numpy.resize(arr, shape)`

numpy.resize() 返回指定形状的新数组。

numpy.resize(arr,shape) 和ndarray.resize(shape, refcheck=False)区别:

- numpy.resize(arr,shape),有返回值,返回复制内容.如果维度不够,会使用原数组数据补齐
- ndarray.resize(shape, refcheck=False),修改原数组,不会返回数据,如果维度不够,会使用0补齐

```
a = np.array([[1,2,3],[4,5,6]])
print('a数组:',a)
#a数组的形状
print('a数组形状:',a.shape)

```

```
a数组: [[1 2 3]
 [4 5 6]]
a数组形状: (2, 3)
```

```
b = np.resize(a,(3,3))
b
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [1, 2, 3]])
```

```
a
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
a.resize((3,3),refcheck=False)
a
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [0, 0, 0]])
```

numpy.append()

在数组的末尾添加值，默认返回一个一维数组。

`numpy.append(arr, values, axis=None)`

参数说明：

- arr：输入的数组；
- values：向 arr 数组中添加的值，需要和 arr 数组的形状保持一致；
- axis：默认为 None，返回的是一维数组；当 axis =0 时，追加的值会被添加到行，而列数保持不变，若 axis=1 则与其恰好相反。

```
a = np.array([[1,2,3],[4,5,6]])
#向数组a添加元素
print (np.append(a, [7,8,9]))

```

```
[1 2 3 4 5 6 7 8 9]
```

```
print(a)
#沿轴 0 添加元素
print (np.append(a, [[7,8,9]],axis = 0))

```

```
[[1 2 3]
 [4 5 6]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
#沿轴 1 添加元素
print (np.append(a, [[5,5,5,5],[7,8,9,6]],axis = 1))
```

```
---------------------------------------------------------------------
---------

ValueError                                Traceback (most recent
call last)

<ipython-input-41-81c5fd935fa9> in <module>
      1 #沿轴 1 添加元素
----> 2 print (np.append(a, [[5,5,5,5],[7,8,9,6],[7,8,9,6]],axis =
1))
```

```
<__array_function__ internals> in append(*args, **kwargs)
```

```
D:\Anaconda3\lib\site-packages\numpy\lib\function_base.py in
append(arr, values, axis)
   4691         values = ravel(values)
   4692         axis = arr.ndim-1
→ 4693     return concatenate((arr, values), axis=axis)
   4694
   4695
```

```
<__array_function__ internals> in concatenate(*args, **kwargs)
```

```
ValueError: all the input array dimensions for the concatenation
axis must match exactly, but along dimension 0, the array at index
0 has size 2 and the array at index 1 has size 3
```

numpy.insert()

表示沿指定的轴，在给定索引值的前一个位置插入相应的值，如果没有提供轴，则输入数组被展开为一维数组。

`numpy.insert(arr, obj, values, axis)`

参数说明：

- arr：要输入的数组
- obj：表示索引值，在该索引值之前插入 values 值；
- values：要插入的值；

- axis：指定的轴，如果未提供，则输入数组会被展开为一维数组。

```
a = np.array([[1,2],[3,4],[5,6]])
#不提供axis的情况，会将数组展开
print (np.insert(a,3,[11,12]))
#沿轴 0 垂直方向
print (np.insert(a,1,[11],axis = 0))
#沿轴 1 水平方向
print (np.insert(a,1,11,axis = 1))
```

```
[ 1  2  3 11 12  4  5  6]
[[ 1  2]
 [11 11]
 [ 3  4]
 [ 5  6]]
[[ 1 11  2]
 [ 3 11  4]
 [ 5 11  6]]
```

numpy.delete()

该方法表示从输入数组中删除指定的子数组，并返回一个新数组。它与 insert() 函数相似，若不提供 axis 参数，则输入数组被展开为一维数组。

`numpy.delete(arr, obj, axis)`

参数说明：

- arr：要输入的数组；
- obj：整数或者整数数组，表示要被删除数组元素或者子数组；
- axis：沿着哪条轴删除子数组。

```
a = np.arange(12).reshape(3,4)
#a数组
print(a)
#不提供axis参数情况
print(np.delete(a,5))
#删除第二列
print(np.delete(a,1,axis = 1))


```

```
a = np.array([1,2,3,4,5,6,7,8,9,10])
#删除多行
print(np.delete(a,[1,2,3]))

# 注意不能使用切片的形式
#print(np.delete(a,[1:4]))
```

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

```
np.s_[1:4]
```

```
slice(1, 4, None)
```

```
np.r_[1:5]
```

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

```
np.c_[1:4]
```

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

numpy.argwhere()

该函数返回数组中非 0 元素的索引，若是多维数组则返回行、列索引组成的索引坐标。

```
x = np.arange(6).reshape(2,3)

```

```
y = np.random.rand(2,3)

```

```
x[x>3]
```

```
array([4, 5])
```

```
x_gt3 = np.argwhere(x>2)

#y[x_gt3[...,0],[...,1]]
```

```
---------------------------------------------------------------------
---------

IndexError                                Traceback (most recent
call last)

<ipython-input-6-24ce64314e19> in <module>
      1 x_gt3 = np.argwhere(x>2)
----> 2 y[x_gt3[...,0],[...,1]]
```

```
IndexError: only integers, slices (`:`), ellipsis (`...`),
numpy.newaxis (`None`) and integer or boolean arrays are valid
indices
```

```
x = np.arange(6).reshape(2,3)
print(x)
#返回所有大于1的元素索引
y=np.argwhere(x>1)
print(y,y.shape)
```

```
[[0 1 2]
 [3 4 5]]
[[0 2]
 [1 0]
 [1 1]
 [1 2]] (4, 2)
```

numpy.unique()

用于删除数组中重复的元素，其语法格式如下：

`numpy.unique(arr, return_index, return_inverse, return_counts)`

参数说明：

- arr：输入数组，若是多维数组则以一维数组形式展开；
- return_index：如果为 True，则返回新数组元素在原数组中的位置（索引）；
- return_inverse：如果为 True，则返回原数组元素在新数组中的位置（索引）；
- return_counts：如果为 True，则返回去重后的数组元素在原数组中出现的次数。

```
a = np.array([5,2,6,2,7,5,6,8,2,9])
print (a)
# 对a数组的去重
uq = np.unique(a)
print (uq)
```

```
[5 2 6 2 7 5 6 8 2 9]
[2 5 6 7 8 9]
```

```
# 数组去重后的索引数组
u,indices = np.unique(a, return_index = True)
# 打印去重后数组的索引
print(u)

print('-'*20)

print(indices)
```

```
[2 5 6 7 8 9]
--------------------
[1 0 2 4 7 9]
```

```python
# 去重数组的下标:
ui,indices = np.unique(a,return_inverse = True)
print (ui)
print('-'*20)
# 打印下标
print (indices)
```

```python
# 返回去重元素的重复数量
uc,indices = np.unique(a,return_counts = True)
print (uc)
# 元素出现次数:
print (indices)
uc[np.argmax(indices)]
# argmin
uc[np.argmin(indices)]
```

```
[2 5 6 7 8 9]
[3 2 2 1 1 1]
```

```
7
```

```python
# 取出出现次数最多或最小的值和索引
```

numpy.sort()

对输入数组执行排序，并返回一个数组副本。

`numpy.sort(a, axis, kind, order)`

参数说明：

- a：要排序的数组；
- axis：沿着指定轴进行排序，如果没有指定 axis，默认在最后一个轴上排序，若 axis=0 表示按列排序，axis=1 表示按行排序；
- kind：默认为 quicksort（快速排序）；
- order：若数组设置了字段，则 order 表示要排序的字段。

```
a = np.array([[3,7,5],[6,1,4]])
print('a数组是: ', a)

#调用sort()函数
b = np.sort(a)
print('排序后的内容:',b)
a
```

```
a数组是:  [[3 7 5]
 [6 1 4]]
排序后的内容: [[3 5 7]
 [1 4 6]]
```

```
array([[3, 7, 5],
       [6, 1, 4]])
```

```
#以行为参照,列上面的数据排序:
print(np.sort(a, axis = 0))
```

```
[[3 1 4]
 [6 7 5]]
```

```
#以列为参照,行上面的数据排序:
print(np.sort(a, axis = 1))
```

```
[[3 5 7]
 [1 4 6]]
```

```
#设置在sort函数中排序字段
dt = np.dtype([('name',  'S10'),('age',  int)])
a = np.array([("raju",21),("anil",25),("ravi",  17),
("amar",27)], dtype = dt)
#再次打印a数组
print(a)
print('----'*10)
#按name字段排序
print(np.sort(a, order = 'age'))
```

```
[(b'raju', 21) (b'anil', 25) (b'ravi', 17) (b'amar', 27)]
---------------------------------------
[(b'ravi', 17) (b'raju', 21) (b'anil', 25) (b'amar', 27)]
```

numpy.argsort()

argsort() 沿着指定的轴，对输入数组的元素值进行排序，并返回排序后的元素索引数组。示例如下：

```
a = np.array([90, 29, 89, 12])
print("原数组:",a)
sort_ind = np.argsort(a)
print("打印排序元素索引值:",sort_ind)
a[sort_ind]
#使用索引数组对原数组排序
#sort_a = a[sort_ind]
#print("打印排序数组")
#for i in sort_ind:
#    print(a[i],end = " ")
```

```
原数组: [90 29 89 12]
打印排序元素索引值: [3 1 2 0]
```

```
array([12, 29, 89, 90])
```

练习题：

```
# 1. 创建长度为10的零向量
```

```
# 2.获取数组np.zeros((10, 10))所占内存大小
```

```
# 3.创建一个长度为10的零向量，并把第五个值赋值为1
```

```
# 4.创建一个值域为10到49的向量
```

```
# 5.将一个向量进行反转（第一个元素变为最后一个元素）
```

```
# 6.创建一个3x3的矩阵，值域为0到8
```

```
# 7.从数组[1, 2, 0, 0, 4, 0]中找出非0元素的位置索引
```

```
# 8.创建一个3x3x3的随机数组
```

```
# 9.创建一个10x10的随机数组，并找出该数组中的最大值与最小值
```

```
# 10.创建一个长度为30的随机向量，并求它的平均值
```

```
# 11.创建一个10*10的2维数组，该数组边界值为1，内部的值为0
```

```
# 12.对5x5的随机矩阵进行归一化 提示:归一化（x - min）/（max - min))
```

```
# 13一个5x3的数组和一个3x2的数组是否可以相乘
```

```
# 14. 创建一个大小为10的随机向量，并把它排序
```

```
# 15. 创建一个numpy数组元素值全为True（真）的数组
```

```

```