1、通过python的list列表创建数组

In [3]:
```
import numpy as np
```

In [3]:
```
a = np.array([1,2,3,4])
b = np.array([[1,2],[3,4],[5,6]])
#查看array的属性，包括数据的维度和类型
print(b.ndim)
print(b.shape)
print(b.dtype)
```

```
2
(3, 2)
int32
```

2、通过Numpy的函数创建数组

In [4]:
```
# 创建等差一维数组，步长为1
c = np.arange(10)
print("c=",c)

# 创建等差一维数组，[0，2]分成11等分后的数组
d = np.linspace(0,2,11)
print(" d=",d)

#创建3×3的全1数组
e = np.ones((3,3 ))
print("e=",e)

#创建3×6的全零数组
f = np.zeros((3,6))
print("f=",f)

#创建4×4的对角数组
g = np.eye(4)
print("g=",g)

#创建6×4的随机数组
h = np.random.randn(6,4)
print("h=",h)
```

```
c= [0 1 2 3 4 5 6 7 8 9]
 d= [0.  0.2 0.4 0.6 0.8 1.  1.2 1.4 1.6 1.8 2. ]
e= [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
f= [[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
g= [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
h= [[-0.3394676  -1.07332358 -0.73602057  0.67964438]
 [ 0.27419171  1.09401895  0.22734163 -0.95773142]
 [ 0.40146577  0.41600182 -1.12522247 -0.23261734]
 [-0.18317688 -0.47427887 -1.57004558 -0.84941357]
 [-1.31766873  0.76340462 -0.86068608 -1.0901683 ]
 [-0.89254091 -1.4136243   0.49203729 -0.49056419]]
```

In [5]:
```
#一维数组
a= np.arange(10)
print(a)

print(a[0], a[3] , a[-1])
print(a[:4])
print(a[3:7])
print(a[6:])
print(a[2:8:2])# 3个参数表示起始、结束和步长，不包含结束位置print(a[2:2])#结束位置可以
print(a[:3])#开始和结束都省略
```

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

In [7]:
```
# 二维数组
a= np.arange(0 , 51,10).reshape(6,1) + np.arange(6)

#一维向量，1×6数组矩阵
print(np.arange(0, 51,10))

#转变成6×1数组矩阵
print(np.arange(0 , 51,10).reshape(6,1))

#一维向量，1×6数组矩阵
print(np.arange(6))

#两个矩阵相加运算
print(a)

#以下为对二维数组矩阵进行索引切片操作
print(a[0 , 0], a[2,-1])
print(a[0,2 : 5])
print(a[:3,3:])
print(a[2,: ])
print(a[ : , 3])#结果应该是列向量，Numpy自动转换行向量
print(a[ : , : : 2])
print(a[ :: 2 ,:: 3])
```

```
[ 0 10 20 30 40 50]
[[ 0]
 [10]
 [20]
 [30]
 [40]
 [50]]
[0 1 2 3 4 5]
[[ 0  1  2  3  4  5]
 [10 11 12 13 14 15]
 [20 21 22 23 24 25]
 [30 31 32 33 34 35]
 [40 41 42 43 44 45]
 [50 51 52 53 54 55]]
0 25
[2 3 4]
[[ 3  4  5]
 [13 14 15]
 [23 24 25]]
[20 21 22 23 24 25]
[ 3 13 23 33 43 53]
[[ 0  2  4]
 [10 12 14]
```

```
 [20 22 24]
 [30 32 34]
 [40 42 44]
 [50 52 54]]
[[ 0  3]
 [20 23]
 [40 43]]
```

In [8]:

```
#数组和标量的运算
#最简单的数值计算是数组和标量进行计算，计算过程是直接把数组里的元素和标量逐个进行计算
a = np.arange(6)
print(a)
print(a + 5)#数组和标量加法
# 创建1~5之间的20个随机整数，形成一维向量，即1×20的数组矩阵
b = np.random.randint(1 ,5,20)
print(b)
# 将1×20的数组矩阵构建出4×5的数组矩阵
c = b.reshape(4 ,5)
print(c)
print(c * 3)#数组和标量乘法
```

```
[0 1 2 3 4 5]
[ 5  6  7  8  9 10]
[4 2 4 1 4 1 3 1 4 1 3 2 3 1 2 4 2 2 1 4]
[[4 2 4 1 4]
 [1 3 1 4 1]
 [3 2 3 1 2]
 [4 2 2 1 4]]
[[12  6 12  3 12]
 [ 3  9  3 12  3]
 [ 9  6  9  3  6]
 [12  6  6  3 12]]
```

In [9]:

```
#数组和数组的运算，如果数组的维度相同，那么在组里对应位置进行逐个元素的数学运算。
# 创建5×4数组矩阵，元素是1~5随机整数
a = np.random.random_integers(1 , 5,(5,4 ))
print(a)

# 创建一个5×4的数组矩阵，每个元素都是1
b = np.ones((5,4), dtype=int)
print(b)
#数组加法
a + b
print(a + b)

#创建3×4数组矩阵，元素是1~5的随机整数
c = np.random.random_integers(1 ,5, ( 3,4 ))
print(c)

# 创建3×4的数组矩阵，元素是1~5随机整数
d = np.random.random_integers(1 ,5,( 3,4 ))
print(d)

#数组相乘，逐元素相乘，不是矩阵内积运算
c* d
print(c * d)
```

```
[[1 3 4 1]
 [2 2 2 4]
 [2 1 4 4]
 [3 5 4 2]
 [1 5 4 2]]
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]
```

```
 [1 1 1 1]
 [1 1 1 1]]
[[2 4 5 2]
 [3 3 3 5]
 [3 2 5 5]
 [4 6 5 3]
 [2 6 5 3]]
[[1 3 1 2]
 [2 1 2 2]
 [4 4 5 4]]
[[3 5 3 1]
 [3 1 5 1]
 [1 2 1 5]]
[[ 3 15  3  2]
 [ 6  1 10  2]
 [ 4  8  5 20]]
<ipython-input-9-852546b0ce65>:3: DeprecationWarning: This function is deprecated. Ple
ase call randint(1, 5 + 1) instead
  a = np.random.random_integers(1 , 5,(5,4 ))
<ipython-input-9-852546b0ce65>:14: DeprecationWarning: This function is deprecated. Pl
ease call randint(1, 5 + 1) instead
  c = np.random.random_integers(1 ,5, ( 3,4 ))
<ipython-input-9-852546b0ce65>:18: DeprecationWarning: This function is deprecated. Pl
ease call randint(1, 5 + 1) instead
  d = np.random.random_integers(1 ,5,( 3,4 ))
```

In [10]:

```
#数组的乘法是对应元素相乘，不是矩阵内积，矩阵内积使用的是np.dot()函数
#创建一个3×2的数组矩阵，每个元素是1~5的随机整数，random_integers函数包含末尾数5
a = np.random.random_integers(1 ,5,(3,2))
print(a)
#创建一个2×3的数组矩阵，每个元素是1~5的随机整数
b = np.random.random_integers(1 , 5,(2,3))
print(b)
#矩阵内积
np.dot(a, b)
print(np.dot(a, b))
```

```
[[1 1]
 [1 5]
 [5 4]]
[[1 2 2]
 [4 1 2]]
[[ 5  3  4]
 [21  7 12]
 [21 14 18]]
<ipython-input-10-72f50d59ff01>:3: DeprecationWarning: This function is deprecated. Pl
ease call randint(1, 5 + 1) instead
  a = np.random.random_integers(1 ,5,(3,2))
<ipython-input-10-72f50d59ff01>:6: DeprecationWarning: This function is deprecated. Pl
ease call randint(1, 5 + 1) instead
  b = np.random.random_integers(1 , 5,(2,3))
```

In [11]:

```
#数组可以直接进行比较，返回一个同维度的布尔数组。针对布尔数组，可以使用all)/any()函数
a= np.array([1,2,3,4])
b = np.array([4 ,2,2,4])
print(a == b)
print(a > b)
print((a==b).all())#必须布尔矩阵中所有元素均为True时结果才为True
print((a== b).any())#只要布尔矩阵中有任一元素为True时结果就为True
```

```
[False  True False  True]
[False False  True False]
False
True
```

In [12]:

```
# 内置函数
a = np.arange(6)
print(a)
#求余弦
print(np.cos(a))
#求指数
print(np.exp(a))
#求平方根
print(np.sqrt(a))
```

```
[0 1 2 3 4 5]
[ 1.          0.54030231 -0.41614684 -0.9899925  -0.65364362  0.28366219]
[  1.           2.71828183   7.3890561   20.08553692  54.59815003
 148.4131591 ]
[0.         1.         1.41421356 1.73205081 2.         2.23606798]
```

In [14]:

```
#创建6个元素的一维向量，元素为1~5之间随机整数
#求最小值
a= np.random.random_integers(1,5,6)
print(a.min())
print(a)
#求最大值
#求和
print(a.max())
print(a.sum())
#最小值元素所在的索引
#求平均值
print(a.argmin())
print(a.mean())
#最大值元素所在的索引
# 求标准差
print(a.argmax())
print(a.std())
```

```
1
[2 3 1 3 5 4]
5
18
2
3.0
4
1.2909944487358056
<ipython-input-14-d26141310b07>:3: DeprecationWarning: This function is deprecated. Pl
ease call randint(1, 5 + 1) instead
  a= np.random.random_integers(1,5,6)
```

In [16]:

```
# 二维数组统计运算
#创建一个6×4二维数组矩阵，每个元素的值为1~5之间的随机整数
b = np.random.random_integers(1 , 5,(6,4))
print(b)

#求矩阵中所有元素的和
print(b.sum())

#按照矩阵的列求和,axis=0表示按列
print(b.sum(axis=0))
#按照矩阵的行求和，axis=1表示按行
print(b.sum(axis=1))

#先按照行求和，得到一个数组，再将数组的所有元素求和
print(b.sum(axis =1).sum())

#按照矩阵的行求每一行的最小值
print(b.min(axis=1))
```

```
#按照矩阵的行求每一行的最小值所对应的索引值(指的是在特定行的索引)
print(b.argmin(axis=1))

#按照矩阵的行求每一行的标准差
print(b.std(axis=1))
```

```
[[3 3 5 1]
 [2 5 4 1]
 [1 2 3 3]
 [4 3 2 5]
 [4 5 5 3]
 [1 2 2 2]]
71
[15 20 21 15]
[12 12  9 14 17  7]
71
[1 1 1 2 3 1]
[3 3 0 2 3 0]
[1.41421356 1.58113883 0.8291562  1.11803399 0.8291562  0.4330127 ]
<ipython-input-16-be82c1ec2f09>:3: DeprecationWarning: This function is deprecated. Pl
ease call randint(1, 5 + 1) instead
  b = np.random.random_integers(1 , 5,(6,4))
```

In [17]:

```
#reshape():转换维度，变成多维数组
# ravel():将多维数组“摊平”，变成一维向量

# 创建一维向量，元素为1~11的12个整数
a= np.arange(12)
print(a)
#转换为4×3的二维数组
b = a.reshape(4 , 3)
print(b)
# 变为一维向量，转换时按照行的顺序进行排列，第2行接在第1行的末尾
print(b.ravel())

#创建一维向量，元素为1~3的4个整数
a= np.arange(4)
print(a)

#打印纬度信息，为(4,)，即只有第1维具有4个元素，没有第2维
print(a.shape)

#在列上添加一个维度，纬度信息为(4,1)，变成4×1数组，其自身位于第1列
b = a[: np.newaxis]
print(b)
#打印纬度信息，为(4,1)，即4×1的数组
print(b.shape)

#在行上添加一个维度，纬度信息为(1,4)，变成1×4数组，其自身位于第1行
c = a[np.newaxis,:]
print(c)

#打印纬度信息，为(1,4)，即1×4的数组
print(c.shape)
```

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

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

In [18]:

```
#数组的排序和索引
# 创建6×4的二维数组，元素为1~10的10个随机整数
a = np.random.random_integers(1,10,(6,4))
print(a)

#按行独立排序，返回一个备份b，即a不变
b = np.sort(a, axis=1)
print(b)
#按列排序，直接把结果保存到当前数组
a.sort(axis=0)
print(a)
#创建由6个1~10的随机整数构成的一维向量
a = np.random.random_integers(1 ,10,6)
#打印排序前的a
print(a)

# 此时的a仍然是排序前的数组
idx = a.argsort()
#打印排序前的a
print(a)

#假如将a排序后，a中各个元素在排序前的索引值
print(idx)

#a仍然是原始的排序前的a，而a[idx]是一个虚拟的排序后的a
print(a[idx])

#将虚拟的排序后的数组，赋值给b，使得b成为物理上排序后的数组
b = a[idx]
# a仍然是排序前的数组，而b是排序后的数组
print(b)

#由于a并没有物理上的排序操作，因此a仍然是排序前的a，下面打印排序前的a

print(a)
```

```
[[ 5  8  4 10]
 [10  7  6  7]
 [ 7  9  2  6]
 [ 8  2  1  7]
 [ 3  4  4 10]
 [ 7  9  7 10]]
[[ 4  5  8 10]
 [ 6  7  7 10]
 [ 2  6  7  9]
 [ 1  2  7  8]
 [ 3  4  4 10]
 [ 7  7  9 10]]
[[ 3  2  1  6]
 [ 5  4  2  7]
 [ 7  7  4  7]
 [ 7  8  4 10]
 [ 8  9  6 10]
 [10  9  7 10]]
[ 6 10 10  9  2  9]
[ 6 10 10  9  2  9]
[4 0 3 5 1 2]
[ 2  6  9  9 10 10]
[ 2  6  9  9 10 10]
[ 6 10 10  9  2  9]
<ipython-input-18-1e617981d847>:3: DeprecationWarning: This function is deprecated. Pl
ease call randint(1, 10 + 1) instead
```

```
  a = np.random.random_integers(1,10,(6,4))
<ipython-input-18-1e617981d847>:13: DeprecationWarning: This function is deprecated. P
lease call randint(1, 10 + 1) instead
  a = np.random.random_integers(1 ,10,6)
```

test文件存储与导入

In [19]:
```
#.Numpy数组作为文本文件，可以直接保存到文件系统里，也可以从文件系统里读取出数据，也可以

a = np.arange(15).reshape(3 ,5)
#将数组保存文本文件
np.savetxt("test.txt" , a)#保存时将整型转换成浮点型
```

In [20]:
```
#读取文本文件，恢复成数组
b = np.loadtxt("test.txt")
print(b)
```

```
[[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]
 [10. 11. 12. 13. 14.]]
```

In [21]:
```
#将数组保存为二进制文件
np.save("test.npy", a)
```

In [22]:
```
# 读取二进制文件，恢复成数组
c = np.load("test.npy")
print(c)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
```

In [ ]:
```

```

Pandas

In [1]:
```
import pandas as pd
```

In [6]:
```
s = pd.Series([4 , 2,5 ,0 , 6,3])
print(s)
# DataFrame是二维数组对象
df = pd.DataFrame(np.random.randn(6,4), columns = list(['name','sex','age','add']))
print(df)
```

```
0    4
1    2
2    5
3    0
4    6
5    3
dtype: int64
       name       sex       age       add
0  0.362121 -0.999879  0.312608  0.423049
1  0.758857  1.017790  0.658011  0.710428
2 -0.329908 -1.131401 -0.206563 -0.240963
3 -0.624279 -0.517627 -0.059643 -0.854407
```

```
4  1.034407  0.485744 -1.999249  0.306186
5  0.594606 -1.181732 -0.205827  0.827692
```

In [19]:

```
# Series 对象可以理解为一维数组
s = pd.Series([4 , 2,5 ,0 , 6,3])
print(s)

# DataFrame是二维数组对象
# df = pd.DataFrame(np.random.randn(6,4), columns = list('ABCD'))
df = pd.DataFrame(np.random.randn(6,4), columns = list(['name','sex','age','add']))
#print(df)

#打印某一行数据
print(df.iloc[0])

#打印某一列数据
#print(df.A)
print(df.name)

#查看数据的维度信息
print(df.shape)

#查看前几行数据
df.head(3)

#查看后几行数据
df.tail(2)

#查看行索引信息
print('行索引：',df.index)

#查看列索引信息
print('列索引：',df.columns)

#查看数据的统计信息,算出每列的元素个数、平均值、标准差、最小值、最大值及几种中位数的值
print('统计：')
print(df.describe())
```

```
0    4
1    2
2    5
3    0
4    6
5    3
dtype: int64
name   -0.336072
sex    -0.683600
age     1.142856
add    -1.167471
Name: 0, dtype: float64
0   -0.336072
1   -0.771726
2    0.980503
3   -1.365613
4   -0.253273
5   -0.387299
Name: name, dtype: float64
(6, 4)
行索引： RangeIndex(start=0, stop=6, step=1)
列索引： Index(['name', 'sex', 'age', 'add'], dtype='object')
统计：
           name       sex       age       add
count  6.000000  6.000000  6.000000  6.000000
mean  -0.355580 -0.151837 -0.399207 -0.130146
std    0.773345  1.097139  1.279506  1.477125
min   -1.365613 -1.884972 -2.308821 -1.554394
```

```
25%   -0.675619 -0.511535 -1.152870 -1.356978
50%   -0.361685  0.027974 -0.186009 -0.363444
75%   -0.273973  0.124125  0.416922  0.744165
max    0.980503  1.453202  1.142856  2.075194
```

In [22]:

```
# 数据映射
#通过Series.map()函数对数据进行转换映射

# 创建一个Series对象a
a = pd.Series(['Java','C','C++'])
print('a组：')
print(a)

print()
#对a中的数据进行映射转换
b= a.map({'Java': '11','C': '22','C++': '33'})
print('b组：')
print(b)
```

```
a组：
0    Java
1       C
2     C++
dtype: object

b组：
0    11
1    22
2    33
dtype: object
```

In [25]:

```
# 数据排序
#通过DataFrame.sort_index()函数对索引进行排序，通过DataFrame.soti_values()对数值进行排

# DataFrame是二维数组对象
df = pd.DataFrame(np.random.randn(6,4), columns = list('ABCD'))
print(df)
# 根据列名称进行逆序排列，axis=1表示按列
print(df.sort_index(axis=1, ascending=False))
#根据B这一列的数据从小到大进行排序
print(df.sort_values(by ='B'))
```

```
          A         B         C         D
0  0.329717 -0.059489  0.901134 -0.859686
1 -0.301952  0.001746  3.621898  1.172631
2 -1.241323 -0.162597 -1.105483  0.024316
3  1.196289 -1.040562 -1.001477 -1.553270
4  1.747654 -0.778663  0.730327 -0.427326
5 -1.475357  0.044950 -0.387159 -1.998673
          D         C         B         A
0 -0.859686  0.901134 -0.059489  0.329717
1  1.172631  3.621898  0.001746 -0.301952
2  0.024316 -1.105483 -0.162597 -1.241323
3 -1.553270 -1.001477 -1.040562  1.196289
4 -0.427326  0.730327 -0.778663  1.747654
5 -1.998673 -0.387159  0.044950 -1.475357
          A         B         C         D
3  1.196289 -1.040562 -1.001477 -1.553270
4  1.747654 -0.778663  0.730327 -0.427326
2 -1.241323 -0.162597 -1.105483  0.024316
0  0.329717 -0.059489  0.901134 -0.859686
1 -0.301952  0.001746  3.621898  1.172631
5 -1.475357  0.044950 -0.387159 -1.998673
```

In [8]:

```
import pandas as pd
```

```
import numpy as np
# 数据访问
# DataFrame是二维数组对象
df = pd.DataFrame(np.random.randn(6, 4), columns = list('ABCD'))
print('df=',df)

#获取原始数据，不包含行号、列名称，只包含纯数据
print('纯数据=',df.values)

#通过行索引范围来访问特定几行的数据
print('切片索引：',df[3 : 5])

#选择A、B、D这3列数据:
print('ABD列数据=',df[['A','B','D']])

#数通过标签来选择某个元素
print(df.loc[3 ,'A'])

#通过数组索引来访问某个元素
print('3,0=',df.iloc[3, 0])
```

```
df=           A         B         C         D
0  0.151843  0.758752  0.460027  0.184973
1  0.350904 -0.641328  2.169870  1.951034
2 -0.211794  0.384549 -0.766406 -0.564402
3 -0.020339  0.861019  0.644914 -0.776380
4 -0.263050 -0.524833  0.400063  1.219772
5 -0.614960 -1.711552 -0.352444  0.179036
纯数据= [[ 0.15184311  0.75875239  0.4600271   0.18497316]
 [ 0.35090394 -0.64132833  2.16987046  1.95103441]
 [-0.21179383  0.38454908 -0.76640573 -0.56440155]
 [-0.02033868  0.86101882  0.64491407 -0.77638023]
 [-0.26304988 -0.52483273  0.40006323  1.21977205]
 [-0.61496019 -1.711552   -0.35244421  0.17903562]]
切片索引：          A         B         C         D
3 -0.020339  0.861019  0.644914 -0.776380
4 -0.263050 -0.524833  0.400063  1.219772
ABD列数据=          A         B         D
0  0.151843  0.758752  0.184973
1  0.350904 -0.641328  1.951034
2 -0.211794  0.384549 -0.564402
3 -0.020339  0.861019 -0.776380
4 -0.263050 -0.524833  1.219772
5 -0.614960 -1.711552  0.179036
-0.020338677115586352
3,0= -0.020338677115586352
```

In [12]:

```
print('[2 : 5,0: 2]结果如下：')
print(df.iloc[2 : 5,0: 2])

#选择C列里所有大于0的数据所在的行
print('C列里所有大于0的数据=')
print(df[df.C > 0])
#添加一列，列名为TAG
df['TAG']=['cat' , 'dog', 'cat', 'cat', 'cat ', 'dog']
print(df)

#根据TAG列做分组统计。
print(df.groupby("TAG").sum())
```

```
[2 : 5,0: 2]结果如下：
          A         B
2 -0.211794  0.384549
3 -0.020339  0.861019
4 -0.263050 -0.524833
C列里所有大于0的数据=
```

```
          A         B         C         D   TAG
0  0.151843  0.758752  0.460027  0.184973   cat
1  0.350904 -0.641328  2.169870  1.951034   dog
3 -0.020339  0.861019  0.644914 -0.776380   cat
4 -0.263050 -0.524833  0.400063  1.219772   cat
          A         B         C         D   TAG
0  0.151843  0.758752  0.460027  0.184973   cat
1  0.350904 -0.641328  2.169870  1.951034   dog
2 -0.211794  0.384549 -0.766406 -0.564402   cat
3 -0.020339  0.861019  0.644914 -0.776380   cat
4 -0.263050 -0.524833  0.400063  1.219772   cat
5 -0.614960 -1.711552 -0.352444  0.179036   dog
            A         B         C         D
TAG
cat -0.080289  2.004320  0.338535 -1.155809
cat -0.263050 -0.524833  0.400063  1.219772
dog -0.264056 -2.352880  1.817426  2.130070
```

Pandas:文件操作:数据文件保存、导入

·使用DataFrame.to csv()函数把数据保存到文件中，使用DataFrame. read csv()函数从文件中读取数据

In [28]:

```
# ppt 16页
```

In [ ]:

```
#读取文件，并且从第0列开始读取
df = pd.read_csv('data.csv',index_col=0)

#数据的形状，即(行数，列数)，如(100，4)
print(df.shape)
#数据的前5行
print(df.head(5))
#保存数据到csv文件
df.to_csv('csv_data_save.csv')
#保存数据到excel文件
df.to_excel('excel_data_save.xls')
```