NumPy 学习指南总结

第 2 章：Numpy 基础

创建多维数组

```
# coding:utf-8

import numpy as np

m=np.array([np.arange(2),np.arange(2)])

print m

print m.shape
```

```
[[0 1]
 [0 1]]
(2L, 2L)
```

一维数组切片和索引

```
# coding:utf-8

import numpy as np

a=np.arange(9)

print a

print a[3:7]

print a[:7:2] #用下标 0-7，以 2 为步长选取元素
```

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

多维数组切片和索引

```
# coding:utf-8

import numpy as np

b=np.arange(24).reshape(2,3,4)

print b
```

```
print '-------------'
print 'b[0,0,0]',b[0,0,0]
print '-------------'
print 'b[:,0,0]',b[:,0,0]
print '-------------'
print 'b[0]',b[0]
print '-------------'
print 'b[0,1]',b[0,1]
print '-------------'
print 'b[0,1,::2]',b[0,1,::2] #上面数组间隔选取元素
print '-------------'
print 'b[:,1]',b[:,1]
print '-------------'
print 'b[0,:,1]',b[0,:,1]
print '-------------'
print 'b[0,:,-1]',b[0,:,-1] #第一层楼最后一列
print '-------------'
print 'b[0,::-1,-1]',b[0,::-1,-1] #反向选取第一层楼的最后一列的所有房间
print '-------------'
print 'b[0,::2,-1]',b[0,::2,-1] #数组切片中间隔的选定元素
```

```
print '--------------'

print 'b[::-1]',b[::-1] #第一层和第二层交换位置
```

```
C:\Users\LIN\Anaconda2\python.exe D:/untitled4/1.py
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
--------------
b[0,0,0] 0
--------------
b[:,0,0] [ 0 12]
--------------
b[0] [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
--------------
b[0,1] [4 5 6 7]
--------------
b[0,1,::2] [4 6]
--------------
b[:,1] [[ 4  5  6  7]
 [16 17 18 19]]
--------------
b[0,:,1] [1 5 9]
--------------
b[0,:,-1] [ 3  7 11]
--------------
b[0,::-1,-1] [11  7  3]
--------------
b[0,::2,-1] [ 3 11]
--------------
```

```
b[::-1] [[[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]

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

改变数组的维度

ravel 函数可以完成展平操作 shape 改变维度

```
# coding:utf-8

import numpy as np

b=np.arange(24).reshape(2,3,4)

print b

a= b.ravel()

print a

a.shape=(6,4) #设置数组维度

print a
```

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
```

数组的组合 np.hstack 水平组合 np.vstack 垂直组合

```
# coding:utf-8

import numpy as np

a=np.arange(9).reshape(3,3)

print a
```

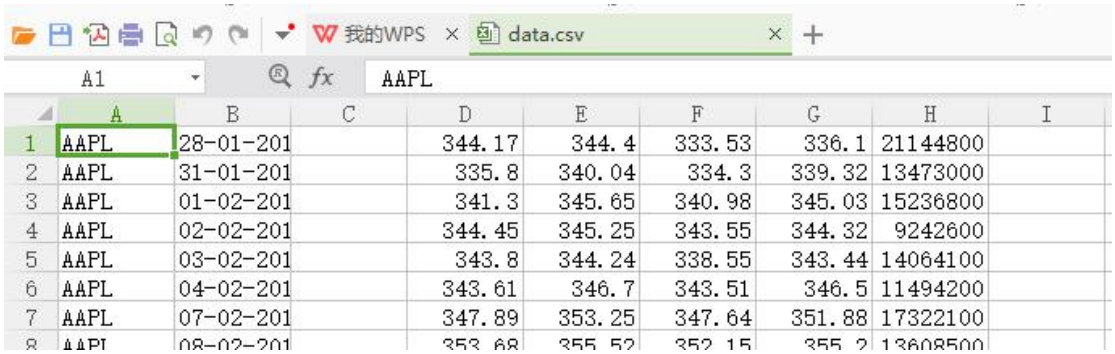
```
b=2*a
print b
print '--------------------'
print np.hstack((a,b)) #数组水平组合
print '--------------------'
print np.vstack((a,b)) #垂直组合
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
[[ 0  2  4]
 [ 6  8 10]
 [12 14 16]]
--------------------
[[ 0  1  2  0  2  4]
 [ 3  4  5  6  8 10]
 [ 6  7  8 12 14 16]]
--------------------
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 0  2  4]
 [ 6  8 10]
 [12 14 16]]
```

第 3 章：常用函数

import numpy as np
i2=np.eye(2)　　#2*2 数组
np.savetxt('eye.txt',i2) #存储文件

AAPL，28-01-2011，　，344.17,344.4,333.53,336.1,21144800 #很多数据其中一行
…



我的WPS × data.csv ×

A1 fx AAPL

	A	B	C	D	E	F	G	H	I
1	AAPL	28-01-201		344.17	344.4	333.53	336.1	21144800	
2	AAPL	31-01-201		335.8	340.04	334.3	339.32	13473000	
3	AAPL	01-02-201		341.3	345.65	340.98	345.03	15236800	
4	AAPL	02-02-201		344.45	345.25	343.55	344.32	9242600	
5	AAPL	03-02-201		343.8	344.24	338.55	343.44	14064100	
6	AAPL	04-02-201		343.61	346.7	343.51	346.5	11494200	
7	AAPL	07-02-201		347.89	353.25	347.64	351.88	17322100	
8	AAPL	08-02-201		353.68	355.52	352.15	355.2	13608500	

c,v=np.loadtxt('data.csv',delimiter=',',usecols=(6,7),unpack=True) #设置分隔符号为，usecols 的参数为一个元组，以获取第 7 字段和第 8 字段的数据。upack 参数设置为 True,意思是拆分存储不同列的数据

vwap=np.average(c,weights=v)　#以 v 列的数据作为权重计算 c 的平均权重值

np.mean(c)　#计算 c 的平均值

时间加权平均价格

t=np.arange(len(c)) #求出行数

np.average(c,weights=t)

h,l=np.loadtxt('data.csv',delimiter=',',usecols=(4,5),unpack=True)　　#将每日最高价和最低价的数据载入数组

np.max(h) #获取该行最大值

np.min(l) #获取该行最小值

ptp 计算数组的取值范围=max(array)-min(array)

np.ptp(h)　np.ptp(l)

np.median(c)　找到中位数

np.msort(c)　将数组从小到大排序

np.var(c)　计算数组的方差

np.diff(c)　返回由相邻数组元素的插值构成的数组

np.std()　返回数组的标准差

np.where(数组>0) where 函数可以根据指定的条件返回所有满足条件的数组元素的索引值

strptime()　函数根据指定的格式把一个时间字符串解析为时间元组。

converters:数据列和转换函数之间进行映射的字典

np.take(数组，索引)　获取数组索引值的元素值

x = np.array([[1, 2], [3, 4]])

>>> x.ravel()

　array([1, 2, 3, 4])

p52 没有完

第 3 章：便捷函数

A1 BHP

	A	B	C	D	E	F	G	H	I
1	BHP	11-02-201		93.11	94.26	92.9	93.72	1741900	
2	BHP	14-02-201		94.57	96.23	94.39	95.64	2620800	
3	BHP	15-02-201		94.45	95.47	93.91	94.56	2461300	
4	BHP	16-02-201		92.67	93.58	92.56	93.3	3270900	
5	BHP	17-02-201		92.65	93.98	92.58	93.93	2650200	
6	BHP	18-02-201		92.34	93	92	92.39	4667300	
7	BHP	22-02-201		93.14	93.98	91.75	92.11	5359800	
8	BHP	23-02-201		91.93	92.46	91.05	92.36	7768400	
9	BHP	24-02-201		92.42	92.71	90.93	91.76	4799100	
10	BHP	25-02-201		93.48	94.04	92.44	93.91	3448300	
11	BHP	28-02-201		94.81	95.11	94.1	94.6	4719800	

我的WPS × BHP.csv × VALE.csv ×

A1 VALE

	A	B	C	D	E	F	G	H	I
1	VALE	11-02-201		33.88	34.54	33.63	34.37	18433500	
2	VALE	14-02-201		34.53	35.29	34.52	35.13	20780700	
3	VALE	15-02-201		34.89	35.31	34.82	35.14	17756700	
4	VALE	16-02-201		35.16	35.4	34.81	35.31	16792800	
5	VALE	17-02-201		35.18	35.6	35.04	35.57	24088300	
6	VALE	18-02-201		35.31	35.37	34.89	35.03	21286600	
7	VALE	22-02-201		33.94	34.57	33.36	33.44	28364700	
8	VALE	23-02-201		33.43	34.12	33.1	33.94	22559300	
9	VALE	24-02-201		34.3	34.3	33.56	34.21	20591900	
10	VALE	25-02-201		34.67	34.95	34.05	34.27	20151500	
11	VALE	28-02-201		34.34	34.51	33.7	34.23	16126000	

```
bhp = np.loadtxt('BHP.csv', delimiter=',', usecols=(6,), unpack=True)

bhp_returns = np.diff(bhp) / bhp[ : -1]

vale = np.loadtxt('VALE.csv', delimiter=',', usecols=(6,), unpack=True)

vale_returns = np.diff(vale) / vale[ : -1]
```

np.corrcoef(bhp_returns,vale_returns)　计算两个矩阵的相关系数
[[1.　　0.67841747
0.67841747　　1.]] 右对角线是相关系数

poly=np.polyfi(长度，差值，　)拟合一系列数据点 实际就是一个函数
np.polyval(poly,长度+1)　推断下一个值
vals=np.ployval(poly,t)
np.argmax(vals)　函数最大值
np.argmin(vals)　函数最小值
np.sign(change)　change 是数据列表　返回对应数据正负号对应列表
hanning 函数是一个加权余弦的窗函数

第 5 章：矩阵和通用函数

```
a=np.mat('1 2 3;4 5 6;7 8 9') #创建矩阵 有空格

a.T 矩阵转置
```

a.I 矩阵求逆

```
A = np.mat(np.arange(9).reshape(3,3))
```

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

```
A = np.eye(2)
```

```
[[ 1.  0.]
 [ 0.  1.]]
```

```
B=2*A
```

```
[[ 2.  0.]
 [ 0.  2.]]
```

```
np.bmat('A B;A B')
```

```
[[ 1.  0.  2.  0.]
 [ 0.  1.  0.  2.]
 [ 1.  0.  2.  0.]
 [ 0.  1.  0.  2.]]
```

```
a=np.arange(9)

print np.add.reduce(a)   求和结果 36
```

```
a=np.array([2,6,5])

b=np.array([1,2,3])

print np.true_divide(a,b)
```

```
[ 2.          3.          1.66666667]
```
数组相除

```
a=np.arange(-4,4)
print a%2
```

```
[0 1 0 1 0 1 0 1]
```

[1,1,2,3,5,8,13,21]

```
a=np.matrix([[1,1],[1,0]]) 创建斐波那契数列矩阵
print (a**4)[0,0]  为5 该数列第5个数
```

第 6 章：深入学习 NumPy 模块

numpy.linalg 模块包含线性代数的函数，使用这个模块可以计算逆矩阵，求特征值，解线性方程组以及求解行列式。

求逆矩阵

```
import numpy as np
A=np.mat('0 1 2;1 0 3;4 -3 8')
print A
inverse=np.linalg.inv(A)
print inverse
```

```
[[ 0  1  2]
 [ 1  0  3]
 [ 4 -3  8]]
[[-4.5  7.  -1.5]
 [-2.   4.  -1. ]
 [ 1.5 -2.   0.5]]
```

求解线性方程组的解

```
# coding:utf-8
import numpy as np
A=np.mat('1 -2 1;0 2 -8;-4 5 9')
```

```
print A

b=np.array([0,8,-9]) #数组 y

print b

x=np.linalg.solve(A,b)

print x
```

```
[[ 1 -2  1]
 [ 0  2 -8]
 [-4  5  9]]
[ 0  8 -9]
[ 29.  16.   3.]
```

numpy.linalg 模块中，eigvals 函数可以计算矩阵的特征值

```
# coding:utf-8

import numpy as np

A=np.mat('3 -2;1 0')

print A

B=np.linalg.eigvals(A)

print B
```

```
[[ 3 -2]
 [ 1  0]]
[ 2.  1.]
```

eig 函数求解特征值和特征向量

```
# coding:utf-8

import numpy as np

A=np.mat('3 -2;1 0')

print A
```

```
B=np.linalg.eig(A)

print B
```

```
[[ 3 -2]
 [ 1  0]]
(array([ 2.,  1.]), matrix([[ 0.89442719,  0.70710678],
        [ 0.4472136 ,  0.70710678]]))
```

奇异值分解

```
# coding:utf-8

import numpy as np

A=np.mat('4 11 14;8 7 -2')

print A

U,Sigma,V=np.linalg.svd(A,full_matrices=False)

print'U'

print U

print 'Sigma'

print Sigma

print 'V'

print V
```

```
[[ 4 11 14]
 [ 8  7 -2]]
U
[[-0.9486833  -0.31622777]
 [-0.31622777  0.9486833 ]]
Sigma
[ 18.97366596   9.48683298]
V
[[-0.33333333 -0.66666667 -0.66666667]
 [ 0.66666667  0.33333333 -0.66666667]]
```

并非得到中间的奇异值矩阵，得到的是对角线上的值

矩阵行列式

```
# coding:utf-8

import numpy as np

A=np.mat('3 4;5 6')

print A

B=np.linalg.det(A)

print B
```

```
[[3 4]
 [5 6]]
-2.0
```

第七章：专用函数

sort 函数返回排序后的数组
lexsort 函数根据键值的字典序进行排序
argsort 函数返回输入数组排序后的下标
ndarray 类的 sort 方法可对数组进行原地排序
msort 函数沿着第一个轴排序
sort_complex 函数对复数按照先实部后虚部的顺序进行排序

argmax 函数返回数组中最大值对应下标 argmin 类似

```
# coding:utf-8

import numpy as np

a = np.array([2,4,8])

print np.argmax(a)
```

```
2
```

searchsorted 函数为指定的插入值返回一个在有序数组中的索引位置

```
# coding:utf-8

import numpy as np
```

```
a = np.arange(5)

print a

indices=np.searchsorted(a,[-2,7])

print indices
```

```
[0 1 2 3 4]
[0 5]
```

numpy 的 extract 函数可以根据某个条件从数组中抽取元素。
使用 nonzero 函数抽取数组中的非零元素

```
# coding:utf-8

import numpy as np

a = np.arange(7)

print a

condition=(a%2)==0

print 'even numbers',np.extract(condition,a)

print 'Non zero',np.nonzero(a)
```

```
[0 1 2 3 4 5 6]
even numbers [0 2 4 6]
Non zero (array([1, 2, 3, 4, 5, 6], dtype=int64),)
```

第 9 章：使用 Matplotlib 绘图

matplotlib.pyplot 包中包含了简单绘图功能
使用 show 函数显示

绘制多项式函数

```
# coding:utf-8

import numpy as np

import matplotlib.pyplot as plt
```

```
func=np.poly1d(np.array([1,2,3,4]).astype(float)) #创建多项式
print func
x=np.linspace(-10,10,30) #在-10 和 10 之间产生 30 个均匀分布的值
y=func(x) #创建多项式的值
plt.plot(x,y) #调用 plot 函数
plt.xlabel('x') #使用 xlabel 函数添加 x 轴标签
plt.ylabel('y(x)')
plt.show()
```