python学习笔记：Numpy

- python复习：Numpy
  - 等差数组

3.2.2 np.linspace

```
In [48]: np.linspace(0,10,21)          #等差数组 开始，结束，元素的数量。两边闭合。
Out[48]: array([ 0. ,  0.5,  1. ,  1.5,  2. ,  2.5,  3. ,  3.5,  4. ,  4.5,  5. ,
                 5.5,  6. ,  6.5,  7. ,  7.5,  8. ,  8.5,  9. ,  9.5, 10. ])

In [90]: np.linspace(0,10,6)           #步长=（10-0）/（5-1）
Out[90]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

  - 等比数组

3.2.3 np.logspace

```
In [73]: np.logspace(0,100,5)    #等比数组，默认以10为底数
Out[73]: array([1.e+000, 1.e+025, 1.e+050, 1.e+075, 1.e+100])
```

```
In [55]: a=1.8
         int(a)
Out[55]: 1
```

```
In [72]: np.logspace(0,6,7,base=2)     #指定以2为底数，
Out[72]: array([ 1.,  2.,  4.,  8., 16., 32., 64.])
```

  - 生成特殊数组

3.2.4 np.zeros,np.ones

```
In [80]: np.zeros(5)              #全零的一维数组
         np.zeros(5).dtype
         np.ones(5)               #全一的一维数组

Out[80]: array([0., 0., 0., 0., 0.])

Out[80]: dtype('float64')

Out[80]: array([1., 1., 1., 1., 1.])

Out[80]: array([[1., 1., 1., 1., 1.],
                [1., 1., 1., 1., 1.]])
```

3.2.5 np.zeros_like

```
In [82]: a=np.array([[1,2,3,4],[6,7,8,9]])
         a
         np.zeros_like(a)          #生成和a的shape一样的全零的数组
         np.ones_like(a)           #生成和a的shape一样的全1的数组

Out[82]: array([[1, 2, 3, 4],
                [6, 7, 8, 9]])

Out[82]: array([[0, 0, 0, 0],
                [0, 0, 0, 0]])

Out[82]: array([[1, 1, 1, 1],
                [1, 1, 1, 1]])
```

3.2.6 np.full

```
In [92]:
1 np.full((4,),3)
2 np.full((4,5),3)
```

```
Out[92]: array([3, 3, 3, 3])
```

```
Out[92]: array([[3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3]])
```

3.2.7 np.identity ; np.eye

```
In [114]:
1 np.identity(4)   #单位数组:行数=列数
```

```
Out[114]: array([[1., 0., 0., 0.],
                 [0., 1., 0., 0.],
                 [0., 0., 1., 0.],
                 [0., 0., 0., 1.]])
```

```
In [115]:
1 np.eye(4)  #单位数组
```

```
Out[115]: array([[1., 0., 0., 0.],
                 [0., 1., 0., 0.],
                 [0., 0., 1., 0.],
                 [0., 0., 0., 1.]])
```

3.2.8 np.diag

```
In [93]:
1 np.diag([1,2,3,4,5]) #对角数组
```

```
Out[93]: array([[1, 0, 0, 0, 0],
                [0, 2, 0, 0, 0],
                [0, 0, 3, 0, 0],
                [0, 0, 0, 4, 0],
                [0, 0, 0, 0, 5]])
```

```
In [103]:
1 np.diag(range(6))
```

```
Out[103]: array([[0, 0, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0, 0],
                 [0, 0, 2, 0, 0, 0],
                 [0, 0, 0, 3, 0, 0],
                 [0, 0, 0, 0, 4, 0],
                 [0, 0, 0, 0, 0, 5]])
```

- 随机数的产生：random，normal模块

3.2.10 随机数的产生：random模块

In [193]:
```
np.random.randint(2,10,5)     #随机整数，不包括10
np.random.rand(5)             #0到1之间随机浮点数
```

Out[193]: array([4, 9, 8, 7, 9])

Out[193]: array([0.32917232, 0.39660367, 0.3306584 , 0.47870501, 0.25053192])

In [203]:
```
np.random.seed(1)     #固定种子，生成随机数，保证每次生成的随机数都一样
np.random.rand(5)     #一般在对参数进行随机初始化时会用到种子
```

Out[203]:
```
array([4.17022005e-01, 7.20324493e-01, 1.14374817e-04, 3.02332573e-01,
       1.46755891e-01])
```

In [216]:
```
np.random.normal(size=5)#一维数组，标准正态分布。从均值为0，标准差1为的标准正态分布中随机抽取4个数字

np.random.normal(size=(3,2))    #二维数组，标准正态分布。生成的数组中，每行和每列的数字是标准正态分布

np.random.normal(5,2,size=(3,4))    #均值为5，标准差2，shape为(3,4)，正态分布
```

Out[216]: array([2.04202875, 0.44752069, 0.68338423, 0.02288597, 0.85723427])

Out[216]:
```
array([[ 0.18393058, -0.41611158],
       [ 1.25005005,  1.24829979],
       [-0.75767414,  0.58829416]])
```

Out[216]:
```
array([[5.69371867, 7.73406541, 6.34743215, 2.41687459],
       [3.30351217, 4.66680086, 6.83439204, 5.16050118],
       [5.45647755, 3.23904641, 5.55625769, 4.85968645]])
```

- 数据类型的转换

4.1 数据类型的转换

In [269]:
```
np.float64(43)  #浮点数转换
np.int32(78.2)

a=np.random.rand(1)*3
a
np.int(a)
```

Out[269]: 43.0

Out[269]: 78

Out[269]: array([1.92469863])

Out[269]: 1

In [280]:
```
a=np.random.randint(0,10,5)
a
a.dtype

b=a.astype(np.float64) #转化为浮点型数
b
b.dtype
```

Out[280]: array([4, 0, 0, 3, 4])

Out[280]: dtype('int32')

Out[280]: array([4., 0., 0., 3., 4.])

Out[280]: dtype('float64')

- 缺失值：特殊的浮点型数据

4.2 缺失值:特殊的浮点型

In [320]:
```
#缺失值/空值
a=np.nan
a
type(a)          #特殊的浮点型
np.isnan(a)      #空值的判断
2*np.nan         #与空值进行的任何运算,结果均为空值
```

Out[320]: nan

Out[320]: float

Out[320]: True

Out[320]: nan

- 数组的索引

5.1 一维数组的索引

In [325]:
```
a=np.random.randint(0,10,5)#一维数组索引
a
a[0]
a[-1]
a[1:5:2]
a[4:1:-1]
```

Out[325]: array([2, 2, 8, 5, 0])

Out[325]: 2

Out[325]: 0

Out[325]: array([2, 5])

Out[325]: array([0, 5, 8])

5.2 多维数组的索引

In [341]:
```
a = np.array([[1, 2, 3, 4, 5],[6, 7, 8,9,10], [17, 18, 19, 20, 21]])
a

a[1]                    #索引第0行
a[:,(1,2,3)]          #索引第1、第2、第3列
a[:,1:4]              #索引第1、第2、第3列
a[0,(3,4)]          #索引第0行中第3和第4列的元素
```

Out[341]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [17, 18, 19, 20, 21]])
```

Out[341]: array([6, 7, 8, 9, 10])

Out[341]:
```
array([[ 2,  3,  4],
       [ 7,  8,  9],
       [18, 19, 20]])
```

Out[341]:
```
array([[ 2,  3,  4],
       [ 7,  8,  9],
       [18, 19, 20]])
```

Out[341]: array([4, 5])

In [377]:
```
a
a[1:3,3:5]
a[0:2,(0,2,3)]              #没有规律时指定行/列数
a[(0,1,2),(1,2,3)]    #花式索引--从两个序列的对应位置取出两个整数组成下标
```

Out[377]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [17, 18, 19, 20, 21]])
```

Out[377]:
```
array([[ 9, 10],
       [20, 21]])
```

Out[377]:
```
array([[1, 3, 4],
       [6, 8, 9]])
```

Out[377]: array([2, 8, 20])

- 变换数组的形态：reshape新生成；resize原地修改

6.1 reshape

In [383]:
```
a=np.arange(12)
a

a.reshape(3,4)                #元素个数要一致。新生成：原来的数组a形状不变
a
```

Out[383]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])

Out[383]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Out[383]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])

In [386]:
```
a
a.reshape(3,-1)                    #-1表示自动计算，前提是能够整除
a.reshape(-1,6)
```

Out[386]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])

Out[386]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Out[386]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11]])
```

6.2 resize

```
In [395]:
a=np.arange(12)
a
a.resize(3,4)                #原地修改。返回值是空值。不能在赋值给其他变量
a
```

```
Out[395]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

Out[395]: array([[ 0,  1,  2,  3],
                 [ 4,  5,  6,  7],
                 [ 8,  9, 10, 11]])
```

6.3 flatten

```
In [413]:
b=np.arange(12)
b.resize(3,4)
b
b.flatten()             #展平，降维，横着拼
b.reshape(1,-1)
b.reshape(3,-1)
b.flatten(order='F') #展平，降维，竖着拼
```

```
Out[413]: array([[ 0,  1,  2,  3],
                 [ 4,  5,  6,  7],
                 [ 8,  9, 10, 11]])

Out[413]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

Out[413]: array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]])

Out[413]: array([[ 0,  1,  2,  3],
                 [ 4,  5,  6,  7],
                 [ 8,  9, 10, 11]])

Out[413]: array([ 0,  4,  8,  1,  5,  9,  2,  6, 10,  3,  7, 11])
```

6.4 ravel

```
In [396]:
a
```

```
Out[396]: array([[ 0,  1,  2,  3],
                 [ 4,  5,  6,  7],
                 [ 8,  9, 10, 11]])
```

```
In [398]:
a.ravel()    #展平，横着拼
```

```
Out[398]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

- swapaxes：0轴和1轴交换

6.5 swapaxes

```
In [414]:
a=np.arange(12).reshape(3,4)
a
```

```
Out[414]: array([[ 0,  1,  2,  3],
                 [ 4,  5,  6,  7],
                 [ 8,  9, 10, 11]])
```

```
In [415]:
a.swapaxes(0,1)    #0轴和1轴交换
```

```
Out[415]: array([[ 0,  4,  8],
                 [ 1,  5,  9],
                 [ 2,  6, 10],
                 [ 3,  7, 11]])
```

In [416]:
```
a
a.T          #转置
```

Out[416]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Out[416]:
```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

- 三维数组

In [425]:
```
a=np.arange(24).reshape(2,3,4)
a
```

Out[425]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

In [405]:
```
a.swapaxes(1,2)  #三维数组轴交换
```

Out[405]:
```
array([[[ 0,  4,  8],
        [ 1,  5,  9],
        [ 2,  6, 10],
        [ 3,  7, 11]],

       [[12, 16, 20],
        [13, 17, 21],
        [14, 18, 22],
        [15, 19, 23]]])
```

- 横向/纵向堆叠stack

6.6 stack

In [428]:
```
a=np.arange(6).reshape(2,3)
a
b=a*3
b
```

Out[428]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```

Out[428]:
```
array([[ 0,  3,  6],
       [ 9, 12, 15]])
```

In [429]:
```
np.hstack([a,b])  #横向堆叠，水平方向拼接
```

Out[429]:
```
array([[ 0,  1,  2,  0,  3,  6],
       [ 3,  4,  5,  9, 12, 15]])
```

In [430]:
```
np.vstack([a,b])  #纵向堆叠
```

Out[430]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 0,  3,  6],
       [ 9, 12, 15]])
```

- 横向/纵向分割split

6.8 split

In [434]:
```
#分割
a=np.arange(16).reshape(4,4)
a
```

Out[434]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [435]:
```
np.hsplit(a,2)#等分，返回值为列表
```

Out[435]:
```
[array([[ 0,  1],
        [ 4,  5],
        [ 8,  9],
        [12, 13]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11],
        [14, 15]])]
```

In [436]:
```
np.vsplit(a,2)#纵向分割
```

Out[436]:
```
[array([[0, 1, 2, 3],
        [4, 5, 6, 7]]),
 array([[ 8,  9, 10, 11],
        [12, 13, 14, 15]])]
```

- 数组的保存与读取

8 numpy数据的保存和读取

8.1 一个数组情况

In [446]:
```
a1 = np.arange(9).reshape(3,3)      #创建一个数组
np.save('a1.npy',a1)                #保存的是单个数组
```

In [445]:
```
np.save(r'D:\CDA数据分析培训\课表\a1.npy',a1)          #保存到某个文件夹下
```

In [172]:
```
np.load('a1.npy')                   #读取文件
```

Out[172]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

8.2 多个数组情况

In [447]:
```
a1 = np.arange(12).reshape(3,4)
a2 = np.arange(10)
np.savez('arr12.npz',a1=a1,a2=a2)    #保存的是两个数组到一个文件中
```

In [175]:
```
a12=np.load('arr12.npz')  #读取
a12
```

Out[175]:
```
<numpy.lib.npyio.NpzFile at 0x207a02975b0>
```

In [176]:
```
a12['a1']
```

Out[176]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [177]:
```
a12['a2']
```

Out[177]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

- 使用numpy进行统计分析

9.1 排序

In [458]:
```
np.random.seed(43)                          #设置随机种子，固定种子
a = np.random.randint(1,10,size=5)          #生成随机数1~10
a

a.sort()                                    #原地排序，没有返回值
a
```

Out[458]:
```
array([5, 1, 2, 6, 1])
```

Out[458]:
```
array([1, 1, 2, 5, 6])
```

In [467]:
```
a1 = np.random.randint(1,10,size = 12).reshape(3,4)
a1

a1.sort(axis=0)  #二维数组排序
a1
```

Out[467]:
```
array([[4, 2, 5, 3],
       [3, 2, 5, 6],
       [6, 1, 4, 5]])
```

Out[467]:
```
array([[3, 1, 4, 3],
       [4, 2, 5, 5],
       [6, 2, 5, 6]])
```

In [471]:
```
a1 = np.random.randint(1,10,size = 12).reshape(3,4)
a1
a1.sort(axis=1)  #二维数组排序
a1
```

Out[471]:
```
array([[8, 2, 3, 8],
       [4, 2, 2, 4],
       [7, 6, 9, 1]])
```

Out[471]:
```
array([[2, 3, 8, 8],
       [2, 2, 4, 4],
       [1, 6, 7, 9]])
```

In [474]:
```
a = np.random.randint(1,10,size=5)
a

a.argsort()          #从小到大排序，返回其相对应的索引值
a.argmax()           #返回最大值的索引
```

Out[474]: array([2, 7, 4, 1, 2])

Out[474]: array([3, 0, 4, 2, 1], dtype=int64)

Out[474]: 1

9.2 去重复和重复

In [475]:
```
a = np.random.randint(1,10,size = 10)  #生成随机数1~10，shape为10
a

np.unique(a)#去重复
```

Out[475]: array([8, 9, 4, 3, 1, 4, 3, 1, 3, 7])

Out[475]: array([1, 3, 4, 7, 8, 9])

In [482]:
```
a=np.arange(3)
a

np.tile(a,3)    #对整个数组进行重复
a.repeat(3)     #对单个的元素进行重复
```

Out[482]: array([0, 1, 2])

Out[482]: array([0, 1, 2, 0, 1, 2, 0, 1, 2])

Out[482]: array([0, 0, 0, 1, 1, 1, 2, 2, 2])

In [483]:
```
a=np.array([[0,1],[1,0]])
a

np.tile(a,(4,4))                #棋盘样式
```

Out[483]:
```
array([[0, 1],
       [1, 0]])
```

Out[483]:
```
array([[0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0]])
```

9.3 填充

In [484]:
```
#np.pad(数组，边界宽度，常量，常量值)，用常量值填充边界
a=np.ones((2,2))
a
```

Out[484]:
```
array([[1., 1.],
       [1., 1.]])
```

In [110]:
```
np.pad(a,pad_width=2,mode='constant',constant_values=0)
```

Out[110]:
```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 1., 0., 0.],
       [0., 0., 1., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

- 常用的统计函数

In [491]:
```
a=np.arange(6).reshape(2,3)
a
```

Out[491]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```

In [492]:
```
np.sum(a)              #函数形式，通用。a是作为一个参数
                       #通用的，参数可能也是其他类型的数据，效率不如专用的方式

a.sum()              #方法，专用，优先考虑              a.sum()调用方法
a.sum(axis=0)       #0轴和
a.sum(axis=1)       #1轴和
```

Out[492]: 15

Out[492]: 15

Out[492]: array([3, 5, 7])

Out[492]: array([3, 12])

In [504]:
```
a.mean()            #求平均
15/6
a.mean(axis=0)    #求平均，纵轴
a.mean(axis=1)   #求平均，横轴
a.var()           #方差
a.std()          #标准差
a.max()         #最大值
a.min()       #最小值
```

Out[504]: 2.5

Out[504]: 2.5

Out[504]: array([1.5, 2.5, 3.5])

Out[504]: array([1., 4.])

Out[504]: 2.9166666666666665

Out[504]: 1.707825127659933

Out[504]: 5

Out[504]: 0

In [510]:
```
a=np.random.randint(1,10,5)
a

a.argmin()              #求最小元素的索引
a.argmax()              #求最大元素的索引
a.cumsum()              #累计和
a.cumprod()             #累计积
```

Out[510]: array([2, 9, 2, 7, 1])

Out[510]: 4

Out[510]: 1

Out[510]: array([2, 11, 13, 20, 21], dtype=int32)

Out[510]: array([2, 18, 36, 252, 252], dtype=int32)

以上内容整理于 幕布文档