numpy

1. transpose转置函数

```
import numpy as np
x = np.arange(4).reshape((2,2))
print(x)
>> [[0 1]
 [2 3]]
y=x.transpose()
print(y)
>> [[0 2]
 [1 3]]
print(x.transpose(0,1))  #相当于不变轴顺序
>> [[0 2]
 [1 3]]
print(x.transpose(1,0))  #相当于0轴和1轴互换
>> [[0 2]
 [1 3]]


z.transpose(0,1,2) #相当于不变轴顺序
z.transpose(1,0,2) #相当于0轴和1轴互换
```

2. sum,mean,std,var

```
mean,sum,std,var函数形式一样，只不过意义不一样
mean求均值，sum求和，std求标准差，var求方差

sum：对数组中全部或某轴向的元素求和
numpy.sum(a,axis=None,dtype=None,out=None,keepdims=False)
a: 待求和数组
axis: 可选择None，int，整型的tuple类型(如果不指定，会对所有函数求和)
dtype： 返回值类型，默认是参数a的类型 out: 输出存放位置，默认会创建一个新数组
keepdims: 如果设置为True，会保留大小为1的维度。方便后续广播机制实现。如果为False时，维度为1会被省掉

import numpy as np
np.random.seed(1234)
```

```
data = np.random.randint(10, size=(3, 5))
print(data)

>>[[3 6 5 4 8]
 [9 1 7 9 6]
 [8 0 5 0 9]]

# 按行相加
# 不使用keepdims
res_row_nokeep=np.sum(data,axis=1)
print(res_row_nokeep.shape,res_row_nokeep)
# 使用keepdims
res_row_keep=np.sum(data,axis=1,keepdims=True)
print(res_row_keep.shape,res_row_keep)

>>(3,) [26 32 22]
(3, 1) [[26]
 [32]
 [22]]

# 按列相加
# 不使用keepdims
res_col_nokeep=np.sum(data,axis=0)
print(res_col_nokeep.shape,res_col_nokeep)
# 使用keepdims
res_col_keep=np.sum(data,axis=0,keepdims=True)
print(res_col_keep.shape,res_col_keep)

>>(5,) [20  7 17 13 23]
(1, 5) [[20  7 17 13 23]]
```

3. max, argmax, maximum

```
找最大值
np.max(a,axis=None,out=None,keepdims=False) 无参数时，所有中的最大值

np.maximum(X,Y,out=None)
最少两个参数，X与Y逐个比较取较大者，

np.argmax(a,axis=None,out=None)
返回沿轴axis最大值的索引

import numpy as np

# 生成固定的随机矩阵
np.random.seed(1234)
data = np.random.randint(10, size=(3, 5))
print(data)

>>[[3 6 5 4 8]
 [9 1 7 9 6]
 [8 0 5 0 9]]

# 所有值的最大值
max_data = data.max()
print(max_data)
```

```
>>9

# 每一列最大值
data_max_col = data.max(axis=0)
print(data_max_col)

>>[9 6 7 9 9]

# 每一行最大值
data_max_row = data.max(axis=1)
print(data_max_row)

>>[8 9 9]

# 比较两个矩阵大小
data_max_two = np.maximum(data, 5)
print(data_max_two)

>>[[5 6 5 5 8]
 [9 5 7 9 6]
 [8 5 5 5 9]]

# 每一列最大值对应的位置
data_max_col_index = np.argmax(data, axis=0)
print(data_max_col_index)

>>[1 0 1 1 2]

# 每一行最大值对应的位置
data_max_row_index = np.argmax(data, axis=1)
print(data_max_row_index)

>>[4 0 4]

# 整体最大值对应的位置
# 有两种方式
#       1.argmax会找到第一个最大值位置
#       2.where会找到所有等于最大值位置,
data_max_index = np.argmax(data)
print("一维索引：", data_max_index)
row_ind, col_ind = divmod(data_max_index, data.shape[1])
print("二维索引argmax拆分：", row_ind, col_ind)
row_ind_where, col_ind_where = np.where(data == np.max(data))
print("二维索引where寻找：", row_ind_where, col_ind_where)

>>一维索引： 5
二维索引argmax拆分： 1 0
二维索引where寻找： [1 1 2] [0 3 4]
```

4. cumsum, cumprob

```
cumsum累计求和，cumprob累积求乘积
np.cumsum(a,axis=None,dtype=None,out=None)

data = np.array([[1, 2, 3], [4, 5, 6]])
print(data)
>>[[1 2 3]
```

```
 [4 5 6]]

# 按照行累加
data_rowcum = np.cumsum(data, axis=1)
print(data_rowcum)

>>[[ 1  3  6]
 [ 4  9 15]]

# 按照列累加
data_colcum = np.cumsum(data, axis=0)
print(data_colcum)

>>[[1 2 3]
 [5 7 9]]
```

5. sort, argsort, lexsort

```
numpy.sort(a,axis=-1,kind='quicksort',order=None) 返回从小到大排序
numpy.argsort(a,axis=-1,kind='quicksort',order=None) 返回从小到大索引
numpy.argsort(a,axis=-1,kind='quicksort',order=None) 返回多个序列进行排序的索引
a: 要排序的数组
axis： 沿着它排序数组的轴，默认为-1，沿着最后的轴排序，axis=0按列排序，axis=1按行排序
kind: 默认为'quicksort',还有'mergesort','heapsort'
order: 在结构化数组中，可以指定按某个字段排序
注意：是在原始数组上修改

#---------------------------------------sort函数---------------------------------
-----
#  ----------np.sort()和array.sort()区别----------
# 排序算法：np.sort()会返回一个新数组，不改变原来数组，
#           data.sort()会在原来数组上排序，不会返回
data = np.array([2, 1, 4, 8, 5, 0])
res = np.sort(data)
print(data)
print(res)

>>[2 1 4 8 5 0]
[0 1 2 4 5 8]

res=data.sort()
print(data)
print(res)
>>[0 1 2 4 5 8]
None

#  ----------参数axis----------
list1 = [[4, 3, 2], [2, 1, 4]]

# 按照行排序
array = np.array(list1)
array.sort(axis=1)
print(array)

>>[[2 3 4]
 [1 2 4]]
```

```
# 按照列排序
array = np.array(list1)
array.sort(axis=0)
print(array)

>>[[2 1 2]
 [4 3 4]]

#  ----------参数order-----------
# 排序列表是 order=['Height', 'Age'] 先按照'Height'排序。如果Height相等，然后按照'Age'排序
typeName = [('Name', 'S10'), ('Height', float), ('Age', int)]
values = [('Li', 1.8, 41), ('Wang', 1.8, 38), ('Duan', 1.7, 38)]
data = np.array(values, dtype=typeName)
data.sort(order=['Height', 'Age'])
print(data)

>>[(b'Duan', 1.7, 38) (b'Wang', 1.8, 38) (b'Li', 1.8, 41)]
```

```
#---------------------------------------argsort函数---------------------------------
--------
data = np.array([4, 2, 5, 7, 3])
b = np.argsort(data)
print(b)

>>[1 4 0 2 3]

c = np.array([[3, 2], [5, 7]])
index = np.argsort(c, axis=1)
print(index)

>>[[1 0]
 [0 1]]

index = np.argsort(c, axis=0)
print(index)
>>[[0 0]
 [1 1]]
```

```
#---------------------------------------lexsort函数---------------------------------
--------
# np.lexsort((dv,nm)) 先按nm比较，然后按dv比较，最后输出从小到大的索引
nm = ('raju', 'anil', 'ravi', 'amar')
dv = ('f.y.', 's.y.', 's.y.', 'f.y.')
ind = np.lexsort((dv, nm))
print(ind)

>>[3 1 0 2]

# 录入了四位同学的成绩，按照总分排序，总分相同时语文高的优先
math    = (10, 20, 50, 10)
chinese = (30, 50, 40, 60)
total   = (40, 70, 90, 70)
# 将优先级高的项放在后面
ind = np.lexsort((math, chinese, total))
for i in ind:
```

```
    print(total[i],chinese[i],math[i])

>>40 30 10
70 50 20
70 60 10
90 40 50
```

6. range 和 np.arange

```
range([start,]stop[,step]) 根据start与stop指定的范围以及step生成一个序列，
        从start开始默认是0，计数到end结束，但不包括end，step默认是1，且必须是整数
np.arange([start,]stop[,step],dtype=None) 根据start与stop指定的范围以及step生成一个
ndarray
```

7. random

```
1. np.random.rand(d0,d1,...,dn)  rand函数根据维度生成[0,1)之间的数据，包含0，但不包括1

import numpy as np
print(np.random.rand(4,2))
>>[[0.85148854 0.79736542]
 [0.84927757 0.67330582]
 [0.7267554  0.86763532]
 [0.40993618 0.74718662]]

2. np.random.rand(d0,d1,...,dn)  rand函数根据维度生成符合标准正态分布的随机数
3. numpy.random.randint(low, high=None, size=None, dtype='l')
    返回随机整数，范围是 [low,high) size是数组维度，默认数据类型是np.int
4.numpy.random.choice(a, size=None, replace=True, p=None) 从1维array里随机选取
    a是一维数组或者整数(np.arange(a)),size是数组维度，p为数组数据出现的概率，replace代表放
回，True代表放回。
5. numpy.randowm.seed() 随机种子，使得随机数固定
```

8. concatenate

```
数组拼接 np.concatenate((a1,a2),axis=0)
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
b=np.array([[11,12,13],[14,15,16]])
c=np.concatenate((a,b),axis=0)
d=np.concatenate((a,b),axis=1)
print(c)
print(d)
>>[[ 1  2  3]
 [ 4  5  6]
 [11 12 13]
 [14 15 16]]
[[ 1  2  3 11 12 13]
 [ 4  5  6 14 15 16]]
```