成像算法基础

Numpy

高思成像科技
Goss Imaging Technology

NumPy 是什么?

NumPy是在1995年诞生的Python库Numeric的基础上建立起来的。但真正促使NumPy的发行的是Python的SciPy库。

SciPy是2001年发行的一个类似于Matlab，Maple，Mathematica等数学计算软件的Python库，它实现里面的大多数功能。

但SciPy中并没有合适的类似于Numeric中的对于基础的数据对象处理的功能。于是，SciPy的开发者将SciPy中的一部分和Numeric的设计思想结合，在2005年发行了NumPy。

高思成像科技
Goss Imaging Technology

NumPy 有什么?

NumPy是Python的一种开源的数值计算扩展库。它包含很多功能：

· 创建n维数组（矩阵）
· 对数组进行函数运算
· 数值积分
· 线性代数运算
· 傅里叶变换
· 随机数产生
……

NumPy诞生为了弥补List的缺陷。它提供了两种基本的对象：

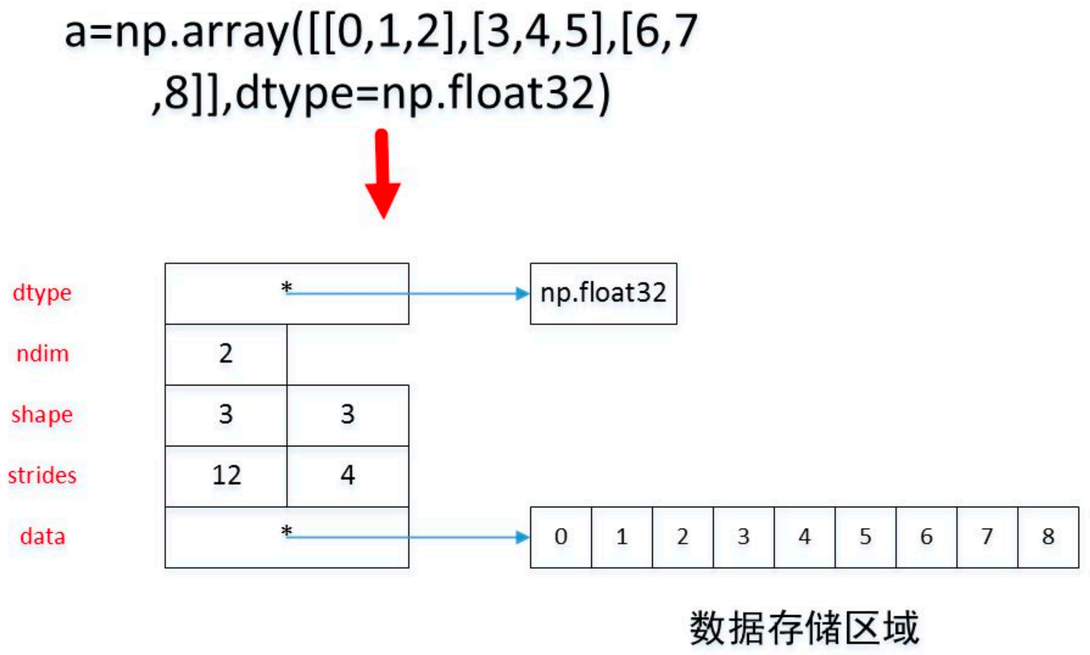
ndarray：全称（n-dimensional array object）是储存单一数据类型的多维数组。

ufunc：全称（universal function object）它是一种能够对数组进行处理的函数。

高思成像科技
Goss Imaging Technology

NumPy Ndarray 对象

- 一个指向数据（内存或内存映射文件中的一块数据）的指针。
- 数据类型或 dtype，描述在数组中的固定大小值的格子。
- 一个表示数组形状（shape）的元组，表示各维度大小的元组。
- 一个跨度元组（stride），其中的整数指的是为了前进到当前维度下一个元素需要"跨过"的字节数。



高思成像科技
Goss Imaging Technology

ndarray的创建

- 来自list和数据
- 特殊的创建

高思成像科技
Goss Imaging Technology

从列表创建

NumPy中的核心对象是**ndarray**。

ndarray可以看成数组，类似于matlab的向量或者矩阵。

NumPy里面所有的函数都是围绕ndarray展开的。

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

	0	1	2	3
0	1	2	3	4

b = np.array((5, 6, 7, 8))

	0	1	2	3
0	5	6	7	8

c = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

	0	1	2	3
0	1	2	3	4
1	5	6	7	8

ndarray对维数没有限制。

[ ]从内到外分别为第0轴，第1轴，第2轴。c第0轴长度为2，第1轴长度为4。

高思成像科技
Goss Imaging Technology

特殊生成

一维初始化

根据步长创建

```
a = np.arange(0, 1, 0.1)
```

根据点数创建

```
b = np.linspace(0, 1, 10)
c = np.linspace(0, 1, 10, endpoint=False)
```

在Log域根据点数创建

```
d = np.logspace(0, 2, 5)
```

多维初始化

空数组

```
a = np.empty((2,3), np.int)
```

全零

```
b = np.zeros((2,4), np.int)
```

全一

```
c = np.ones((6,3), np.int)
```

其它值

```
d = np.full((6,3), np.pi)
```

自定义

```
def func(i,j):
    return i % 4 + 1
e = np.fromfunction(func, (10,4))
print("e=",e)
```

高思成像科技
Goss Imaging Technology

数组属性

方法	描述
ndarray.flags	有关数组内存布局的信息。
ndarray.shape	数组维度的元组。
ndarray.strides	遍历数组时每个维度中的字节元组。
ndarray.ndim	数组维数。
ndarray.data	Python缓冲区对象指向数组的数据的开头。
ndarray.size	数组中的元素数。
ndarray.itemsize	一个数组元素的长度，以字节为单位
ndarray.nbytes	数组元素消耗的总字节数。
ndarray.base	如果内存来自其他对象，则为基础对象
ndarray.dtype	数组元素的数据类型。

高思成像科技
Goss Imaging Technology

常用的属性

ndarray的元素具有相同的元素类型。常用的有int（整型），float（浮点型），complex（复数型）。

```
a = np.array([1, 2, 3, 4], dtype=float)
print(a.dtype)
a = np.array([1, 2, 3, 4])
print(a.dtype)
```

ndarray的shape属性用来获得它的形状，也可以自己指定。

```
c = np.array([[1, 2, 3, 4], [4, 5, 6, 7], [7, 8, 9, 10]])
print(c.shape)
a = np.array([1, 2, 3, 4])
d = a.reshape((2,2))
```

高思成像科技
Goss Imaging Technology

维度属性

shape=3

1维

0 1 2

0维

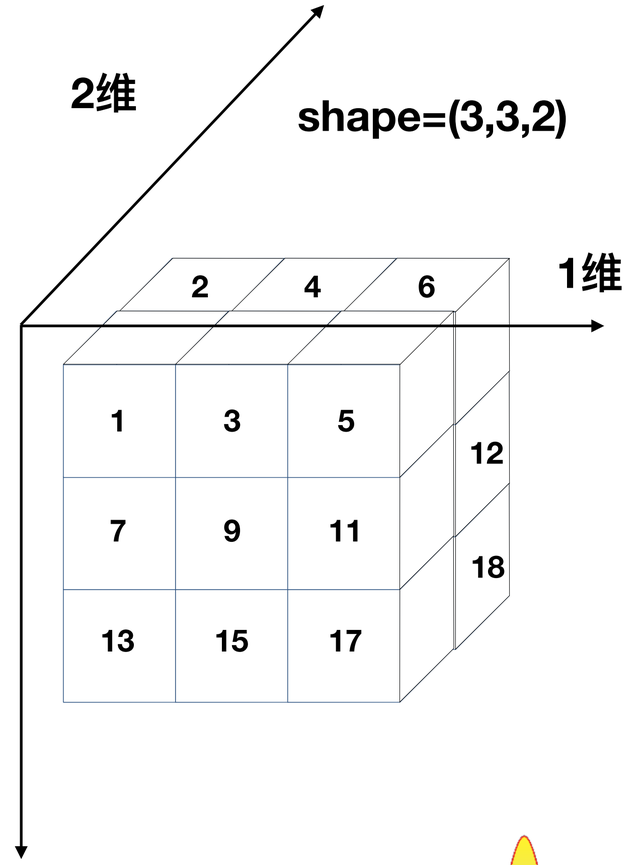
0	1	2

shape=(7,6)

1维

0维

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35
36	37	38	39	40	41

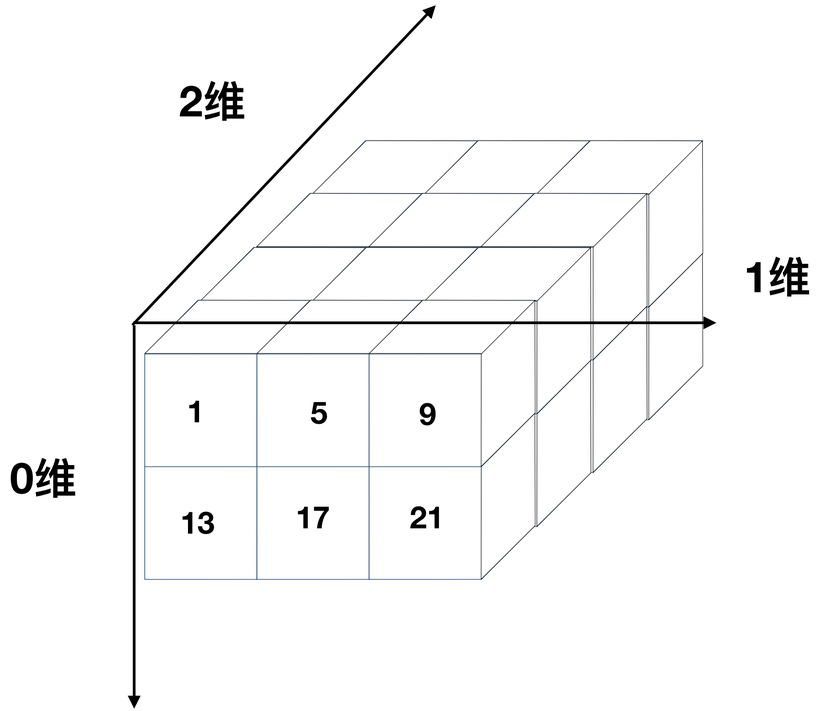


高思成像科技
Goss Imaging Technology

查看存储

- 从最后的维度存储

```
ls=[[[1,2,3,4],[5,6,7,8],[9,10,11,12]],[[13,14,15,16],[17,18,19,20],[21,22,23,24]]]
a=np.array(ls,dtype=int)
print(a)
print(a.strides)
```



高思成像科技
Goss Imaging Technology

a[开始:结束:步长]

ndarray的切片

ndarray通过切片产生一个新的数组b，**b和a共享同一块数据存储空间。**如果想改变这种情况，我们可以**用列表对数组元素切片。或者使用copy函数.**

```
import numpy as np
a = np.arange(10)
print("a=",a)
a= [0 1 2 3 4 5 6 7 8 9]
a[2:4] = 100, 101
b = a[3:7]
b[2] = -10
print("a=",a)
print("b=",b)
a= [  0   1 100 101   4 -10   6   7   8   9]
b= [101   4 -10   6]
c = a[[3, 3, -3, 8]]
print("c=",c)
c= [101 101   7   8]
```

```
c[2] = 100
print("a=",a)
print("c=",c)
a= [  0   1 100 101   4 -10   6   7   8   9]
c= [101 101 100   8]
d= a[3:7].copy()
d[2] = -99
print("a=",a)
print("b=",b)
print("d=",d)
a= [  0   1 100 101   4 -10   6   7   8   9]
b= [101   4 -10   6]
d= [101   4 -99   6]
```

test6.py

负索引

当做切片的操作的时候-1代表最后的元素,-2代表倒数第二个.负索引还可以用作其它的时候如reshape.

```
import numpy as np
b = np.arange(0, 60, 10)
```

	0	1	2	3	4	5
0	0	10	20	30	40	50

```
c = b.reshape(-1,1)
```

	0
0	0
1	10
2	20
3	30
4	40
5	50

```
print(b[-1])
print(c[-2])
print(c[-2,0])
```

50
[40]
40

test7.py

结构数组

C语言中可以通过struct关键字定义结构类型。NumPy中也有类似的**结构数组**。

```
import numpy as np
#import math

persontype = np.dtype({ 'names':['name', 'age', 'weight'],'formats':['S30','i', 'f']})
a = np.array([("Zhang", 32, 75.5), ("Wang", 24, 65.2)],dtype=persontype)
```

	0	1
0	(b'Zhang', 32, 75.5)	(b'Wang', 24, 65.2)

```
#z = math.sin(x) # 错误
print(y)
```

ufunc简介

ufunc是universal function的简称，它是一种能对**数组每个元素进行运算**的函数。NumPy的许多ufunc函数都是用C语言实现的，因此它们的运算速度非常快。

```
import numpy as np
#import math
x = np.linspace(0, 2*np.pi, 10)
y = np.sin(x)
#z = math.sin(x) # 错误
print(y)
```

	0	1	2	3	4	5	6
0	0.00000	0.64279	0.98481	0.86603	0.34202	-0.34202	-0.86603

四则运算

NumPy提供了许多ufunc函数，它们和相应的运算符运算结果相同。。

```
> np.subtract(a, b) # 减法
> np.multiply(a, b) # 乘法
> np.divide(a, b) # 如果两个数字都为整数，则为整数除法
> np.power(a, b) # 乘方
```

比较运算和布尔运算

使用==，>对两个数组进行比较，会返回一个**布尔数组，每一个元素都是对应元素的比较结果。**

```
> np.array([1, 2, 3]) < np.array([3, 2, 1])
array([ True, False, False], dtype=bool)
```

布尔运算在NumPy中也有对应的ufunc函数。

表达式	ufunc函数
y=x1==x2	equal(x1,x2[,y])
y=x1!=x2	not_equal(x1,x2[,y])
y=x1<x2	less(x1,x2[,y])
y=x1<=x2	not_equak(x1,x2[,y])
y=x1>x2	greater(x1,x2[,y])
y=x1>=x2	gerater_equal(x1,x2[,y])

自定义ufunc函数

NumPy提供的标准ufunc函数可以组合出复合的表达式，但是有些情况下，自己编写的则更为方便。我们可以**把自己编写的函数用frompyfunc()转化成ufunc函数。**

frompyfunc(func, nin, nout)

func：计算函数

nin：func()输入参数的个数

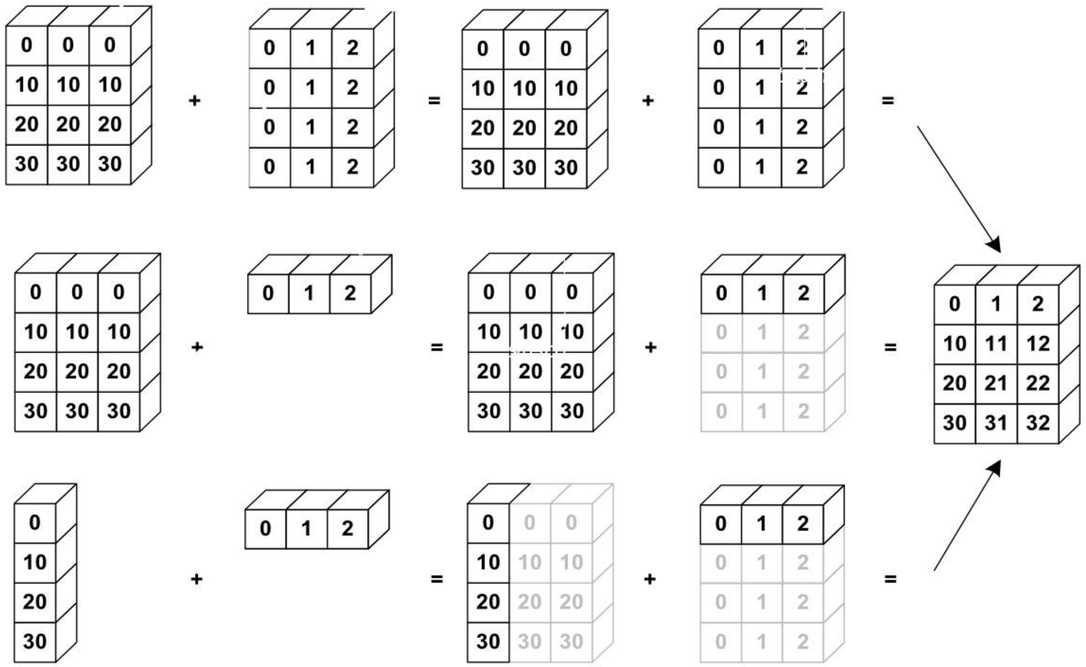
nout：func()输出参数的个数

```
import numpy as np
def num_judge(x, a):  # 对于一个数字如果是3或5的倍数就
    if x % 3 == 0:
        r = 0
    elif x % 5 == 0:
        r = 0
    else:
        r = a
    return r
x = np.linspace(0, 10, 11)
y = np.array([num_judge(t, 2) for t in x])#列表生成式
print("y")
numb_judge = np.frompyfunc(num_judge, 2, 1)
y = numb_judge(x,2)
print("y")
```

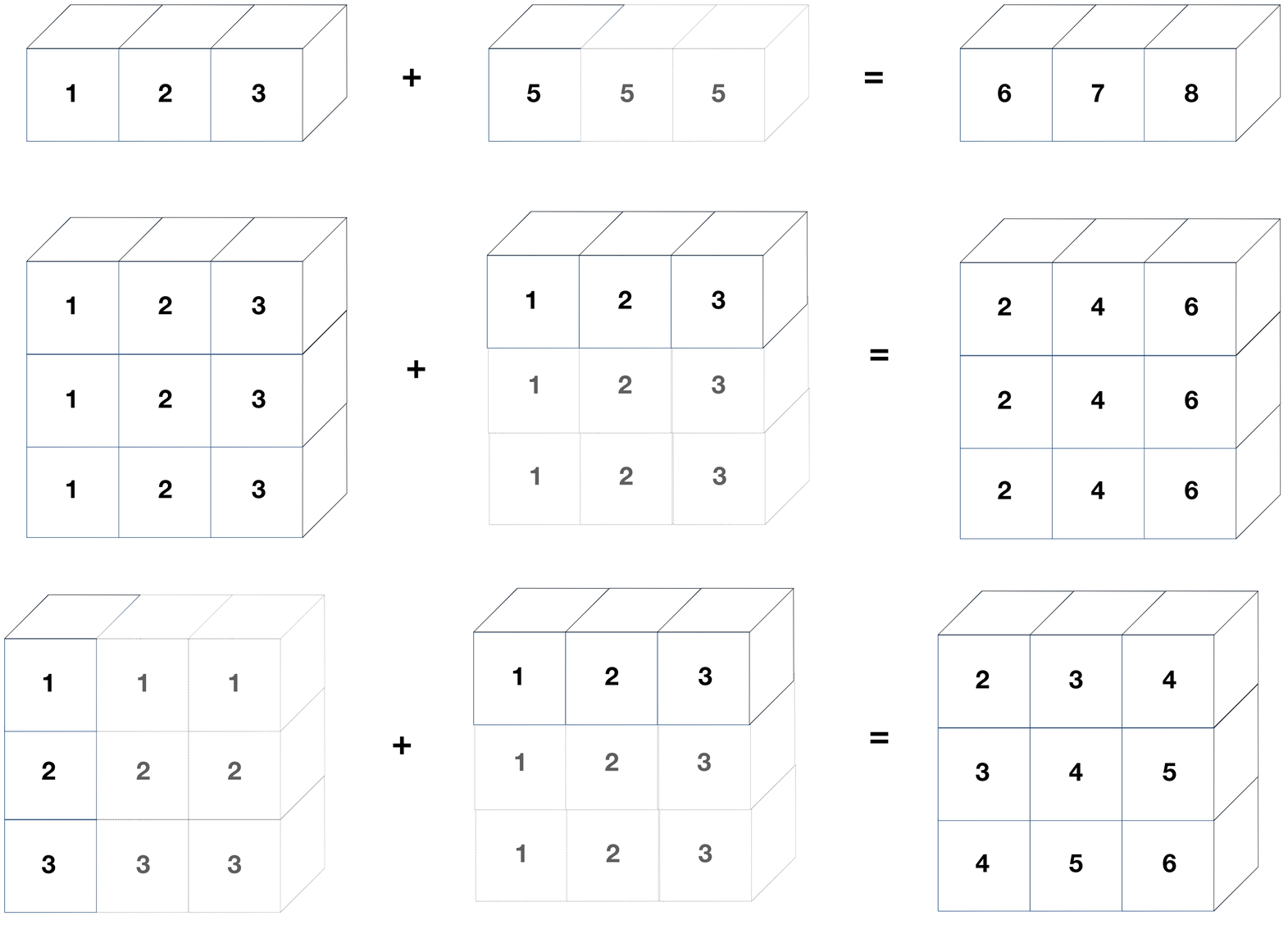
广播(broadcasting)

使用ufunc对两个数组进行运算时，ufunc函数会对两个数组的对应元素进行运算。如果数组的形状不相同，就会进行**下广播**处理。

简而言之，就是向**两个数组每一维度上的最大值靠齐**。



广播(broadcasting)



```
import numpy as np
a = np.arange(0, 60, 10).reshape(-1, 1) + np.arange(0, 6)
```

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

```
b = np.arange(0, 60, 10)
```

	0	1	2	3	4	5
0	0	10	20	30	40	50

```
c = b.reshape(-1,1)
d = np.arange(0, 6)
```

	0	1	2	3	4	5
0	0	1	2	3	4	5

```
e = c+d
```

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

0	0
1	10
2	20
3	30
4	40
5	50

随机数

NumPy产生随机数的模块在**random里面，其中有大量的分布方式。**

rand	0到1之间的随机数	normal	正太分布的随机数
randint	制定范围内的随机整数	uniform	均匀分布
randn	标准正太的随机数	poisson	泊松分布
choice	随机抽取样本	shuffle	随机打乱顺序

求和，平均值，方差

NumPy在**均值等方面常用的函数**如下：

函数名	功能
sum	求和
average	加权平均数
var	方差
mean	期望
std	标准差
product	连乘积

大小与排序

NumPy在**排序等方面常用的函数**如下：

函数名	功能	函数名	功能
min	最小值	max	最大值
ptp	极差	argmin	最小值的下标
mininum	二元最小值	maxinum	二元最大值
sort	数组排序	argsort	数组排序下标
percentile	分位数	median	中位数

特殊的参数和广播

axis,out,keepdims等特殊参数

```
np.random.seed(42)
a = np.random.randint(0,10,size=(4,5))
print("a=",a)
a= [[6 3 7 4 6]
 [9 2 6 7 4]
 [3 7 7 2 5]
 [4 1 7 5 1]]
print("np.sum(a, axis=1)=",np.sum(a, axis=1))
np.sum(a, axis=1)= [26 28 24 18]
print("np.sum(a, axis=0)=",np.sum(a, axis=0))
np.sum(a, axis=0)= [22 13 27 18 16]
print("np.sum(a,1,keepdims=True)=",np.sum(a,1,keepdims=True))
np.sum(a,1,keepdims=True)= [[26]
 [28]
 [24]
 [18]]
print("np.sum(a,0,keepdims=True)=",np.sum(a,0,keepdims=True))
np.sum(a,0,keepdims=True)= [[22 13 27 18 16]]
```

Maximum是逐点比较

```
a = np.array([1, 3, 5, 7])
b = np.array([2, 4, 6])
print(np.maximum(a[None, :], b[:, None]))#maxinum返回两组矩阵广播计算后的结果
[[2 3 5 7]
 [4 4 5 7]
 [6 6 6 7]]
```

高思成像科技
Goss Imaging Technology

大小,排序,统计

函数名	功能
sort	对数组进行排序会改变数组的内容，返回一个新的数组
percentile	**percentile**计算处于p%上的值。
unique	unique有两个参数,**return_index=True**同时返回原始数组中的下标,**return_inverse=True**表示原始数据在新数组的下标。
bicount	**bincount()**对**非负整数数组**中的各个元素出现的次数进行统计，返回数组中的第i个元素是整数i出现的次数。
histogram	对以为数组进行直方图统计，其参数为： **histogram(a, bins=10, range=None, weights=None)** 函数返回两个一维数组，**hist**是每个区间的统计结果, **bin_edges**返回区间的边界

高思成像科技
Goss Imaging Technology

多维数组拼接

vstack(),hstack(),column_stack()拼接函数。

```
import numpy as np
a = np.arange(3)
```

	0	1	2
0	0	1	2

```
b = np.arange(10, 13)
```

	0	1	2
0	10	11	12

```
v = np.vstack((a, b)) # 按第1轴连接数组
```

	0	1	2
0	0	1	2
1	10	11	12

```
h = np.hstack((a, b)) # 按第0轴连接数组
```

	0	1	2	3	4	5
0	0	1	2	10	11	12

```
c = np.column_stack((a, b)) # 按列连接多个一维数组
```

0	0	10
1	1	11
2	2	12

操作多维数组

```
import numpy as np

a = np.array([6, 3, 7, 4, 6, 9, 2, 6, 7, 4, 3, 7])
b = np.array([ 1,  3,  6,  9, 10])
print(np.split(a, b))# 按元素位置进行分段
[array([6]), array([3, 7]), array([4, 6, 9]), array([2, 6, 7]), array([4]), array([3, 7])]
print(np.split(a, 2))
[array([6, 3, 7, 4, 6, 9]), array([2, 6, 7, 4, 3, 7])]
```

array_split
Split an array into multiple sub-arrays of equal or near-equal size. Does not raise an exception if an equal division cannot be made.
hsplit
Split array into multiple sub-arrays horizontally (column-wise).
vsplit
Split array into multiple sub-arrays vertically (row wise).
dsplit
Split array into multiple sub-arrays along the 3rd axis (depth).
concatenate
Join a sequence of arrays along an existing axis.
stack
Join a sequence of arrays along a new axis.
hstack
Stack arrays in sequence horizontally (column wise).
vstack
Stack arrays in sequence vertically (row wise).
dstack
Stack arrays in sequence depth wise (along third dimension).

其它的库

- scipy
- matplotlib
- opencv

高思成像科技
Goss Imaging Technology