

Numpy.

1. Numpy 用于大型多维数组上执行数值运算。
在 Python 中没有数组这一类型，数组可以理解为矩阵。一般就是列表，多个列表嵌套的形式。

2. 用 Numpy 创建数组(矩阵)。

由于 numpy 这个词在程序后面经常要用，所以导入时用 np 作为名字。

```
import numpy as np.
```

Numpy 产生的数组都是 numpy.ndarray 类型的。
可以用 np.array() 或 np.arange() 等方法来创建。
Numpy 没有方法。

```
如: t1 = np.array([1, 2, 3])
```

t1 就是 [1 2 3]，不过↑的数据类型是列表，即 t1 类型是 numpy.ndarray，而且它会把列表中的逗号都去掉。

np.arange(k) 和 np.array(range(k)) 一样，都是形成一个 0 到 k-1 的数组
(也叫 np.arange(a, a2, n)，即从 a 到 a2，步长为 n)

1)

3. 关于数组还有一个类型叫 dtype，dtype 是指数组内存放数据的类型。

dtype 是一个属性，直接在数组后加上。

```
如 print(t3.dtype)
```

numpy 中有比 python 中常见的更多数据类型

由于数据多，尽量多用 int8 等小而类型存储。
dtype 类型默默认时看电脑是多少位就创多少位。
若要改变则可在创建数组时指定。

```
t0t = np.array(range(1, 4), dtype=m).
```

m 可以为 int, float 等，也可字符串 "float32"。
还可以是 bool 类型(会将所有数据变成只有 True or False)。

若要调整数据类型，用 astype 方法，

```
如 t2 = t1.astype("int8")
```

t1 变成随机的小数。

```
import random.
```

```
t = np.array([random.random() for i in range(10)])
```

生成 10 个随机小数。

5. 若要改变数组中小数的位数，用 round 方法。

```
如 t2 = np.round(t1, 2) ← 取 2 位小数。
```

6. 创建多维数组，要写成列表嵌套的形式
如 3 维数组。

```
t3 = np.array([[ [1, 2, 3], [4, 5, 6] ], [[ 7, 8, 9 ],  
[ 10, 11, 12 ] ]]).
```

```
t3: array([[[1, 2, 3],  
[4, 5, 6],  
[7, 8, 9],  
[10, 11, 12]]]).
```

```
t3.shape → (2, 2, 3)
```

$\rightarrow t.shape$ 里有几个数， t 就是几维数组。

7. 修改数组形状：reshape 方法

如 $t_1 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])$

$t_2 = t_1.reshape(3, 4)$ 小括号 reshape 内部无视
它是填一个数组，但也可不填。
 $t_2 = np.array([[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]])$

reshape 成 3 维？

可直接写在 `arrange` 后

$t_3 = np.arange(0, 12).reshape(2, 3, 4)$

3 维

```
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]]])
```

8. `reshape` 方法是有返回值的，有返回值的一

般不会影响数据本身。如对 t `reshape` 后， t 不会变。

会把值往右值接收。

9. 变为一维数组，共 k 个数。

$t.reshape(k)$

若不清楚总个数。

行数 ↓ 列数 ↓

用 $t_2 = t_1.reshape((t_1.shape[0] * t_1.shape[1],))$ }= 二维时。

或 $t_2 = t_1.flatten()$

↑ 按行展开。

10. 数组和数的运算。

$t + i \leftarrow$ 将数组 t 中所有数字加 i (广播机制)

加减乘除都是这样

numpy 里可以除 0，会自动把 0 替成 eps 。

$1 / 0 = nan$ (not a number), 其他数 / 0 = inf.

11. 数组和数组的运算

两个数组形状一致时：对应位置相加。

加减乘除都是这样。

形状不一样时，在相同维度上分别运算（某维度上一样）

```
In [45]: t7 = np.arange(0, 6)
In [46]: t7
Out[46]: array([0, 1, 2, 3, 4, 5])
In [47]: t5
Out[47]:
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 [48]: t5 - t7
Out[48]:
array([[0, 0, 0, 0, 0, 0],
       [6, 6, 6, 6, 6, 6],
       [12, 12, 12, 12, 12, 12],
       [18, 18, 18, 18, 18, 18]])
In [49]: t8 = np.arange(4).reshape((4, 1))
In [50]: t8
Out[50]:
array([[0],
       [1],
       [2],
       [3]])
In [51]: t5 - t8
Out[51]:
array([[0, 1, 2, 3, 4, 5],
       [5, 6, 7, 8, 9, 10],
       [10, 11, 12, 13, 14, 15],
       [15, 16, 17, 18, 19, 20]])
```

5 和 7 每行相减。

5 和 8 每列相减

若两数组每个维度都不同，无法运算，报错

广播原则

如果两个数组的后端维度 (trailing dimension, 即从末尾开始算起的维度) 的轴长长度相符或其中一方的长度为 1，则认为它们是广播兼容的。广播会在缺失和 (或) 长度为 1 的维度上进行。

可是任一维。

怎么理解呢？

可以把维度指的是 `shape` 所对应的数字个数

那么问题来了：

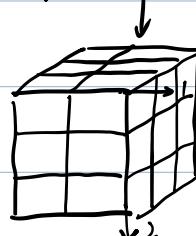
`shape` 为 (3, 3, 3) 的数组能够和 (3, 2) 的数组进行计算么？

`shape` 为 (3, 3, 2) 的数组能够和 (3, 2) 的数组进行计算么？

有什么好处呢？

举个例子：每列的数据减去列的平均值的结果

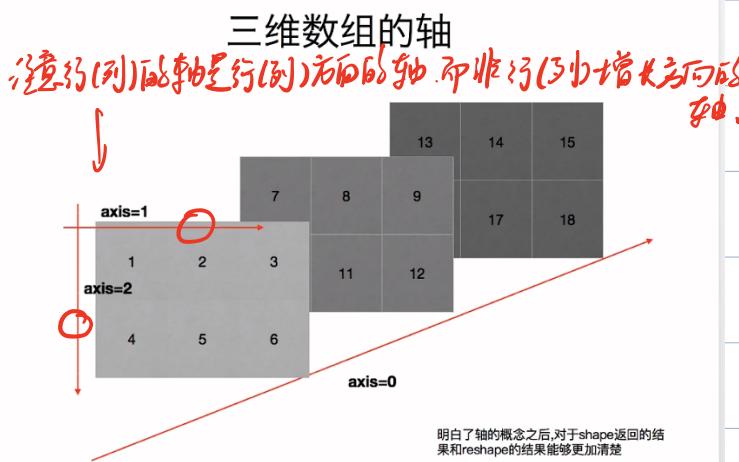
`shape` 为 (3, 3, 3) 和 (3, 3) 也可运算。



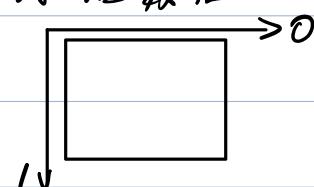
可加在左图的顶上。

12. axis 轴.

可以理解为方向. n维数组有n个轴. 从0轴到n-1轴.



对二维数组



在二维数组中. 对0轴上进行操作就是对行轴作操作. 行数改变. 对1轴上进行操作就是对列轴作操作. 列数改变. (这里行(列)数改造成求sum中一般直接降维. 然后行(列)数直接在shape中消失).
→推广到n维. axis=0就是对shape括号内从左到右第0个数+1个数进行改变.

实例.

①二维. 对shape为(6,4)的a. 进行a.sum(axis=0)

后行数消失. shape变为(4,)

②三维. 对shape为(2,3,4)的a. 进行a.sum(axis=2)

后列数消失. shape为(2,3). (经轴)

记忆：只需求之. 二维上axis=0是对每列上所有数进行运算操作 axis=1是对每行上所有数运算操作.

3维上axis=0是对0轴方向各块上对应数操作

axis=1 ... axis=2 ...

13. numpy 读取数据.

一般用 pandas 来读取. Pandas 功能更加强

numpy 读数据方法.

数据文件名隔开
数据地址

np.loadtxt (frame, dtype=np.float, delimiter

=None, skiprows=0, usecols=None, unpack=False)
↑ ↑ ↑
跳过哪一行 (第几行为标题) 读文件中某几列 韩置.

14. CSV文件：通常分隔值文件. 各数据间

用逗号间隔. 一行为一个记录.

15. 用np.loadtxt读CSV文件不会自动把txt中逗号作为数据分隔. 要用delimiter参数. delimiter=","

16. 若读出数据自行转成科学计数法.

要用dtype参数. dtype="int".

17. unpack参数默认为False. 若改为True. 则读出的数据转置.

18. 数据的转置.

3种方法 { t.transpose

t.T 改换轴.
t.swapaxes(1,0)

19. numpy 的数据索引.

取 - 行. t[2] 取连续多行 t[2:]

取不连续多行. t[[2,8,10]]

取1列 $t[:, 0]$ 取连续多列 $t[:, 2:]$

取不连续多列 $t[:, [0, 2]]$

万能法：在切片中括号加一个逗号，逗号两边若取连续则加冒号，若不取连续则加一个列表 $[x_1, x_2, \dots]$

只取行也可使用万能法，如 $t[2:, :]$

也可用万能法同时取行和列

(取行和列交叉点 $t[2:5, 1:4]$)

取多个不相邻的点、 $t[[0, 2], [0, 1]]$

↑ 第1维从左到右第2维从第2列的左

(tips: 所有的列都是从0开始编号的)

20. 数组的拼接

方法： $\text{np.vstack}((t_1, t_2))$ ← t₁在上, t₂在下。

$\text{np.hstack}((t_1, t_2))$ ← t₁在左, t₂在右

垂直分割是竖直拼接的逆过程 ∵ 是水平切一刀。

水平分割同理。

21. 行列交换

由于在拼接时每行(列)意义不一定相同，故要交换后再拼

行交换：2,3行交换 $\rightarrow t[[1, 2], :] = t[[2, 1], :]$

列交换：1,3列交换 $\rightarrow t[:, [0, 2]] = t[:, [2, 0]]$

22. 求t的行数 $t.shape[0]$.

...列... $t.shape[1]$

23. 创建一个全0数组。 $\text{np.zeros}(13, 4)$

全1数组 $\text{np.ones}(13, 4)$

对角线1数组 $\text{np.eye}(3) \leftarrow$ 方阵

24. 求最小值/最大值位置

每行最大值

最大值 $\text{np.argmax}(t, axis=0)$

最小值 $\text{np.argmin}(t, axis=1)$ 每列最小值

25. numpy生成随机数。

numpy生成随机数

对称或非对称

参数	解释
<code>.rand(d0, d1, ..., dn)</code>	创建d0-dn维度的均匀分布的随机数组，浮点数，范围从0-1
<code>.randn(d0, d1, ..., dn)</code>	创建d0-dn维度的標準正态分布随机数，浮点数，平均数0，标准差1
<code>. randint(low, high, shape)</code>	从给定上下限范围选取随机数整数，范围是low,high，形状是shape
<code>.uniform(low, high, size)</code>	产生具有均匀分布的数组，low起始值，high结束值，size形状
<code>.normal(loc, scale, size)</code>	从指定正态分布中随机抽取样本，分布中心是loc (概率分布的均值)，标准差是scale，形状是size
<code>.seed(s)</code>	随机数种子，s是给定的种子值。因为计算机生成的是伪随机数，所以通过设定相同的随机数种子，可以每次生成相同的随机数

若要多次得的随机值一样，在前面加 $\text{np.random.seed}(10)$

26. copy.

$a=b$ 或 $a=b[:]$ 這種種 copy 在后续程序中

若改变一个，另一个也会跟着变(浅拷贝)。

a, b 间是相互影响。

只能用 $a=b.\text{copy}()$ 。相当于重新开辟一个空间。将b的值 copy 进去后用a指向这一片空间(深拷贝)

或者更简单的方法。对b操作之后重新赋值给b。

27. nan 和 inf 都是 double 类型。

特征 $\text{np.nan} = \text{np.nan}$ 。

用 $\text{np.count_nonzero}()$ 和 $\text{np.isnan}()$ 可判

断某数组中 nan 的个数

例见后

```

In [89]: t2 =
Out[89]:
array([[ 0.,   3.,   3.,   3.,   3.,   3.],
       [ 0.,   3.,   3.,   3.,  10.,  11.],
       [ 0.,  13.,  14.,  15.,  16.,  17.],
       [ 0.,  19.,  20.,  nan,  20.,  20.]])

```

In [90]: np.count_nonzero(t2) 1.两个nan是不相等的
Out[90]: 20

```

In [91]: t2!=t2
Out[91]:
array([[False, False, False, False, False, False],
       [False, False, False, False, False, False],
       [False, False, False, False, False, False],
       [False, False, False, True, False, False]], dtype=bool)

```

```

In [92]: np.count_nonzero(t2!=t2)
Out[92]: 1

```

```

In [93]: np.isnan(t2)
Out[93]:
array([[False, False, False, False, False, False],
       [False, False, False, False, False, False],
       [False, False, False, False, False, False],
       [False, False, False, True, False, False]], dtype=bool)

```

```

In [94]: np.count_nonzero(np.isnan(t2))
Out[94]: 1

```

特性2: nan和任何值计算结果都是nan.

将nan处的值一般会替换成功数值或中值.

```

# print(t1)
def fill_ndarray(t1):
    for i in range(t1.shape[1]): #遍历每一列
        temp_col = t1[:,i] #当前的一列
        nan_num = np.count_nonzero(temp_col!=temp_col)
        if nan_num !=0: #不为0, 说明当前这一列中有nan
            temp_not_nan_col = temp_col[temp_col==temp_col] #当前一列不为nan的array
            # 选中当前为nan的位置, 把值赋值为不为nan的均值
            temp_col[np.isnan(temp_col)] = temp_not_nan_col.mean()
    return t1

```

If __name__ == ...

28. numpy中常用统计函数

numpy中常用统计函数

求和: `t.sum(axis=None)`

均值: `t.mean(axis=None)` 受离群点的影响较大

中值: `np.median(t,axis=None)`

最大值: `t.max(axis=None)`

最小值: `t.min(axis=None)`

极差: `np.ptp(t,axis=None)` 即最大值和最小值之差

标准差: `t.std(axis=None)`

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$

标准差是一组数据平均值分散程度的一种度量。一个较大的标准差，代表大部分数值和其平均值之间差异较大；一个较小的标准差，代表这些数值较接近平均值反映出数据的波动稳定性情况，越大表示波动越大，约不稳定

默认返回多维数组的全部的统计结果,如果指定axis

则返回一个当前轴上的结果

29. 计算

