编 程 学 习 笔 记

Numpy

Code

作 者 姓 名： 颜佳

目 录

[1 前言 1](#_Toc101102919)

[2 Numpy数组对象 1](#_Toc101102920)

[3 Numpy数组的创建 2](#_Toc101102921)

[3.1 array=np.array() 基于list或tuple 2](#_Toc101102922)

[3.2 array=np.arange(1,10,2) 3](#_Toc101102923)

[3.3 array=np.linspace(0,10,5) 3](#_Toc101102924)

[3.4 array=array.reshape(3,4) 3](#_Toc101102925)

[3.5 array=np.random.random((3,3)) 3](#_Toc101102926)

[3.5.1 numpy.random.rand(d0,d1,…,dn) 4](#_Toc101102927)

[3.5.2 numpy.random.randn(d0,d1,…,dn) 4](#_Toc101102928)

[3.5.3 numpy.random.randint(low, high=None, size=None, dtype=’l’) 4](#_Toc101102929)

[3.5.4 numpy.random.random\_integers(low, high=None, size=None) 4](#_Toc101102930)

[3.5.5 numpy.random.random\_sample(size=None) 4](#_Toc101102931)

[3.5.6 numpy.random.random(size=None) 4](#_Toc101102932)

[3.5.7 numpy.random.ranf(size=None) 4](#_Toc101102933)

[3.5.8 numpy.random.sample(size=None) 4](#_Toc101102934)

[3.5.9 numpy.random.choice(a, size=None, replace=True, p=None) 4](#_Toc101102935)

[3.5.10 numpy.random.seed() 4](#_Toc101102936)

[3.6 array=np.zeros((3,3)) 5](#_Toc101102937)

[3.7 array=np.ones((3,3)) 5](#_Toc101102938)

[3.8 array=np.eye(5) 5](#_Toc101102939)

[4 Numpy数组的数值类型 5](#_Toc101102940)

[5 Numpy数组的属性 7](#_Toc101102941)

[5.1 dtype属性 7](#_Toc101102942)

[5.2 ndim属性 8](#_Toc101102943)

[5.3 shape属性 8](#_Toc101102944)

[5.4 size属性 8](#_Toc101102945)

[5.5 itemsize属性 8](#_Toc101102946)

[5.6 nbytes属性 8](#_Toc101102947)

[5.7 T属性 8](#_Toc101102948)

[5.8 复数的实部和虚部属性，real和imag属性 9](#_Toc101102949)

[5.9 flat属性，返回一个numpy.flatiter对象，即可迭代的对象。 10](#_Toc101102950)

[6 Numpy数组的切片和索引 11](#_Toc101102951)

[6.1 一维数组 11](#_Toc101102952)

[6.2 二维数组 12](#_Toc101102953)

[7 Numpy数组的变换 12](#_Toc101102954)

[7.1 6.1 形状转换 12](#_Toc101102955)

[7.1.1 reshape()和resize() 12](#_Toc101102956)

[7.1.2 ravel()和flatten(), 将多维数组转换成一维数组 13](#_Toc101102957)

[7.1.3 transpose()转置 14](#_Toc101102958)

[7.1.4 用tuple指定数组的形状 14](#_Toc101102959)

[7.2 堆叠数组 14](#_Toc101102960)

[7.2.1 np.hstack()与np.column\_stack()，水平叠加 14](#_Toc101102961)

[7.2.2 np.vstack()与np.row\_stack()，垂直叠加 15](#_Toc101102962)

[7.2.3 np.concatenate()方法，通过设置axis的值来设置叠加方向 15](#_Toc101102963)

[7.2.4 np.dstack()，深度叠加 16](#_Toc101102964)

[7.3 拆分数组 17](#_Toc101102965)

[7.3.1 np.hsplit()，沿横向轴拆分（axis=1） 17](#_Toc101102966)

[7.3.2 np.vsplit()，沿纵向轴拆分（axis=0） 18](#_Toc101102967)

[7.3.3 np.dsplit(arr\_dstack,2)，深度拆分 18](#_Toc101102968)

[8 Numpy数组的类型转换 19](#_Toc101102969)

[8.1 数组转换成list，使用tolist() 20](#_Toc101102970)

[8.2 转换成指定类型，astype()函数 20](#_Toc101102971)

[9 Numpy数组的常用统计函数 20](#_Toc101102972)

[9.1 np.ptp()，返回整个数组的最大值减去最小值，如下： 21](#_Toc101102973)

[9.2 np.cumsum()，沿指定轴方向进行累加 21](#_Toc101102974)

[9.3 np.cumprod()，沿指定轴方向进行累乘积 （Return the cumulative product of the elements along the given axis） 22](#_Toc101102975)

[10 数组的广播 22](#_Toc101102976)

# 前言

Numpy是一个开源的Python科学计算库，它是Python科学计算库的基础库，许多其他著名的科学计算库如Pandas，Scikit-learn等都要用到Numpy库的一些功能。

本文主要内容如下：

* Numpy数组对象
* 创建ndarray数组
* Numpy的数值类型
* ndarray数组的属性
* ndarray数组的切片和索引
* 处理数组形状
* 数组的类型转换
* numpy常用统计函数
* 数组的广播

# Numpy数组对象

Numpy中的多维数组称为ndarray。Numpy数组通常是由**相同种类的元素**组成的，即数组中的数据项的类型一致。这样有一个好处，由于知道数组元素的类型相同，所以能快速确定存储数据所需空间的大小。Numpy数组能够运用向量化运算来处理整个数组，速度较快；而Python的列表则通常需要借助循环语句遍历列表，运行效率相对来说要差。

Numpy使用了优化过的C API，运算速度较快

**关于向量化和标量化运算**，对比下面的参考例子就可以看出差异

使用python的list进行循环遍历运算

def pySum():

a = list(range(10000))

b = list(range(10000))

c = []

for i in range(len(a)):

c.append(a[i]\*\*2 + b[i]\*\*2)

return c

%timeit pySum()

10 loops, best of 3: 49.4 ms per loop

使用numpy进行向量化运算

import numpy as np

def npSum():

a = np.arange(10000)

b = np.arange(10000)

c = a\*\*2 + b\*\*2

return c

%timeit npSum()

The slowest run took 262.56 times longer than the fastest. This could mean that an intermediate result is being cached.

1000 loops, best of 3: 128 µs per loop

从上面的运行结果可以看出，numpy的向量化运算的效率要远远高于python的循环遍历运算**（效率相差好几百倍）。**  
（1ms=1000µs）

# Numpy数组的创建

## array=np.array() 基于list或tuple，参数为列表、嵌套列表、元组、嵌套元组、元组列表、列表元组、混合

# 一维数组 # 基于list

arr1 = np.array([1,2,3,4])

print(arr1)

# 基于tuple

arr\_tuple = np.array((1,2,3,4))

print(arr\_tuple)

# 二维数组 (2\*3)

arr2 = np.array([[1,2,4], [3,4,5]])

arr2

[1 2 3 4]

[1 2 3 4]

array([[1, 2, 4],

[3, 4, 5]])

array = np.array(((1,2,3),(4,5,6)))

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

请注意：

一维数组用print输出的时候为 [1 2 3 4]，跟python的列表是有些差异的，没有“**,**”

在创建二维数组时，在每个子list外面还有一个"[]"，形式为“**[**[list1], [list2]**]**”

## array=np.arange(1,10,2)

# 一维数组

arr1 = np.arange(5)

print(arr1)

[0 1 2 3 4]

# 二维数组

arr2 = np.array([np.arange(3), np.arange(3)])

arr2

array([[0, 1, 2],

[0, 1, 2]])

## array=np.linspace(0,10,5)

## array=np.zeros((3,3))

## array=np.ones((3,3))

## array=np.eye(5)

## array=np.random.random((3,3))

### np.random.rand(d0,d1,…,dn)

rand函数根据给定维度生成[0,1)之间的数据，包含0，不包含1

dn表格每个维度

返回值为指定维度的array

### np.random.randn(d0,d1,…,dn)

randn函数返回一个或一组样本，具有标准正态分布。

dn表格每个维度

返回值为指定维度的array

### np.random.randint(low, high=None, size=None, dtype=’l’)

返回随机整数，范围区间为[low,high），包含low，不包含high

参数：low为最小值，high为最大值，size为数组维度大小，dtype为数据类型，默认的数据类型是np.int

high没有填写时，默认生成随机数的范围是[0，low)

### np.random.random\_integers(low, high=None, size=None)

返回随机整数，范围区间为[low,high]，包含low和high

参数：low为最小值，high为最大值，size为数组维度大小

high没有填写时，默认生成随机数的范围是[1，low]

### np.random.random\_sample(size=None)

### np.random.random(size=None)

### np.random.ranf(size=None)

### np.random.sample(size=None)

### np.random.choice(a, size=None, replace=True, p=None)

从给定的一维数组中生成随机数

参数： a为一维数组类似数据或整数；size为数组维度；p为数组中的数据出现的概率

a为整数时，对应的一维数组为np.arange(a)

### np.random.seed()

np.random.seed()的作用：使得随机数据可预测。

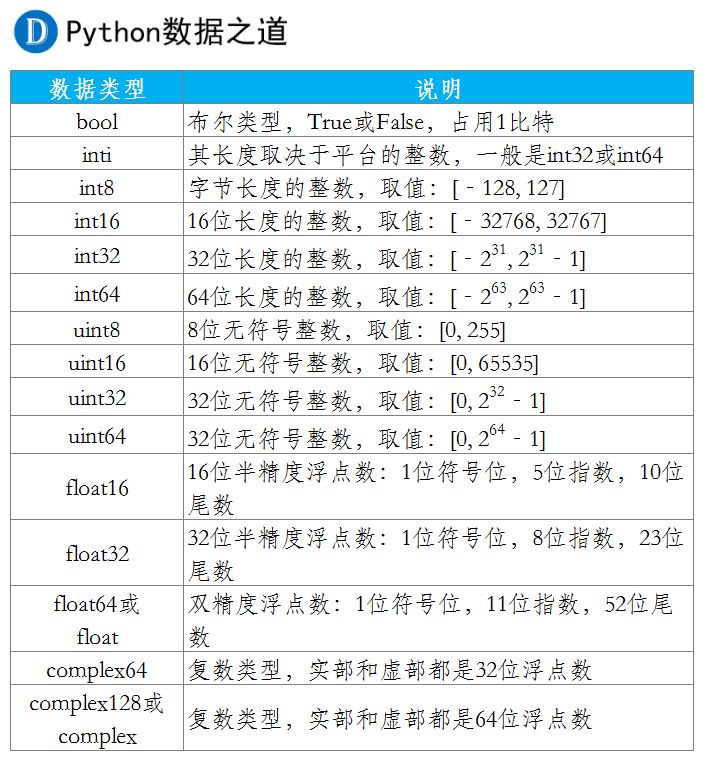
当我们设置相同的seed，每次生成的随机数相同。如果不设置seed，则每次会生成不同的随机数

np.random.seed(0)

np.random.rand(5)

# Numpy数组的数值类型

Numpy的数值类型如下：



# Numpy数组的属性

| **运算** | **结果** |
| --- | --- |
| ndarray.dtype | 数组的数据类型 |
| ndarray.ndim | 数组维度的数量 |
| ndarray.shape | 数组对象的尺度，shape是一个元组（tuple） |
| ndarray.size | 数组元素的数量 |
| ndarray.itemsize | 数组中各个元素所占用的字节数大小 |
| ndarray.nbytes | 整个数组所需的字节数量 |
| ndarray.T | 数组的转置 |
| ndarray.real | 数组中复数元素的实部 |
| ndarray.imag | 数组中复数元素的虚部 |
| ndarray.flat | 返回一个numpy.flatiter对象，即可迭代的对象。 |

# Numpy数据的数学运算

## 乘法

### \*，矩阵对应位置元素相乘

### @，矩阵点乘

### A.dot(B)，矩阵点乘

### np.dot(A, B)

#### 如果a和b都是一维的，那么结果就是普通的內积(inner product)。可以使用np.matmul 或者 a @ b 得到相同的答案。

#### 如果a和b都是二维的，运算等同于矩阵乘法（Dot product）。可以使用np.matmul 或者 a @ b 得到相同的答案。

#### 如果 a 或者 b 中有一个是标量的，效果等价于np.multiply ，可以使用 multiply(a,b) 或者 a \* b 也可以。

#### 如果a是 N-D 数组，b是 1-D的数组，答案是a和b最后一个轴的乘积。

#### 如果a是N-D数组，b是 M-D数组（M>2），答案为a的最后一个轴与a的倒数第二个轴的和。

### np.matnul(A, B)

### np.multiply (A, B)

# Numpy数组的变换

## 形状转换

### ndarray.reshape()和ndarray.resize()方法，

b.reshape(4,3)

array([[ 0, 1, 2],

[ 3, 4, 5],

[ 6, 7, 8],

[ 9, 10, 11]])

b

array([[ 0, 1, 2, 3],

[ 4, 5, 6, 7],

[ 8, 9, 10, 11]])

b.resize(4,3)

b

array([[ 0, 1, 2],

[ 3, 4, 5],

[ 6, 7, 8],

[ 9, 10, 11]])

#### 两者区别：ndarray.resize()方法会改变原数组，相当于ndarray.reshape()方法中 inplace=True

### ndarray.ravel()和ndarray.flatten()方法, 将多维数组转换成一维数组

b.ravel()

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

b.flatten()

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

b

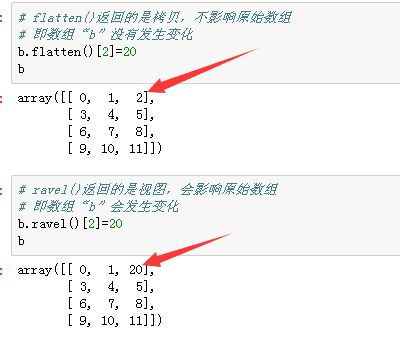
array([[ 0, 1, 2],

[ 3, 4, 5],

[ 6, 7, 8],

[ 9, 10, 11]])

#### 两者区别：ndarray.ravel()方法返回的是视图（view），对视图的修改会影响原始矩阵。ndarray.flatten()方法返回的是一份拷贝，需要分配新的内存空间，对拷贝的修改不会影响原始矩阵。



### ndarray.transpose()发法与ndarray.T属性

### ndarray.shape属性，用tuple指定数组的形状

    arr\_list\_2 = np.array([[0, 1], [1, 2]])

arr\_list\_2.shape = (1, 4)

arr\_list\_2 = [[0 1 1 2]]

## 堆叠数组

### np.hstack()与np.column\_stack()，水平叠加

### np.vstack()与np.row\_stack()，垂直叠加

### np.concatenate()方法，通过设置axis的值来设置叠加方向

axis=1时，沿水平方向叠加

axis=0时，沿垂直方向叠加

np.concatenate((b,c),axis=1)

array([[ 0, 1, 20, 3, 4, 5, 0, 2, 40, 6, 8, 10],

[ 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22]])

np.concatenate((b,c),axis=0)

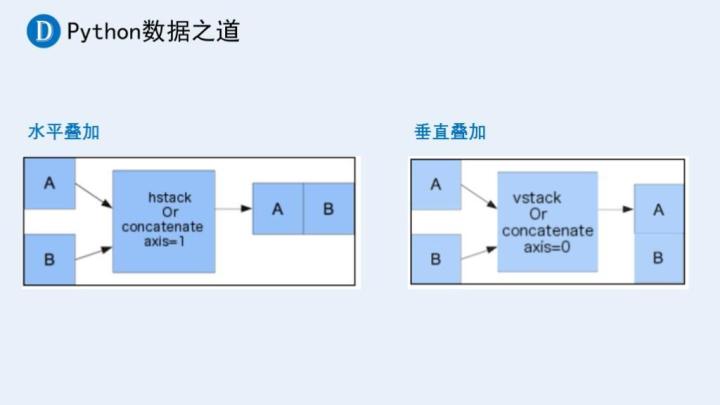
array([[ 0, 1, 20, 3, 4, 5],

[ 6, 7, 8, 9, 10, 11],

[ 0, 2, 40, 6, 8, 10],

[12, 14, 16, 18, 20, 22]])

由于针对数组的轴为0或1的方向经常会混淆，通过示意图，或许可以更好的理解。关于数组的轴方向示意图，以及叠加的示意图，如下：



### np.dstack()，深度叠加

这个有点烧脑，举个例子如下，自己可以体会下：

arr\_dstack = np.dstack((b,c))

print(arr\_dstack.shape)

arr\_dstack

(2, 6, 2)

array([[[ 0, 0],

[ 1, 2],

[20, 40],

[ 3, 6],

[ 4, 8],

[ 5, 10]],

[[ 6, 12],

[ 7, 14],

[ 8, 16],

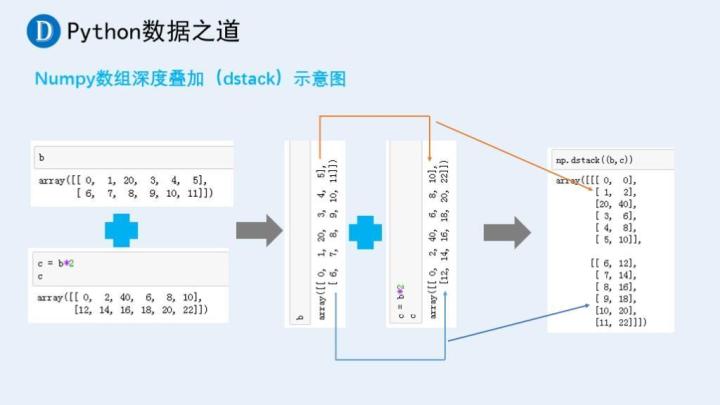
[ 9, 18],

[10, 20],

[11, 22]]])

叠加前，b和c均是shape为（2,6）的二维数组，叠加后，arr\_dstack是shape为（2,6,2）的三维数组。

**深度叠加的示意图如下：**



## 拆分数组

### np.hsplit()，沿横向轴拆分（axis=1）

跟数组的叠加类似，数组的拆分可以分为横向拆分、纵向拆分以及深度拆分。

涉及的函数为 hsplit()、vsplit()、dsplit() 以及split()

b

array([[ 0, 1, 20, 3, 4, 5],

[ 6, 7, 8, 9, 10, 11]])

np.hsplit(b, 2)

[array([[ 0, 1, 20],

[ 6, 7, 8]]), array([[ 3, 4, 5],

[ 9, 10, 11]])]

np.split(b,2, axis=1)

[array([[ 0, 1, 20],

[ 6, 7, 8]]), array([[ 3, 4, 5],

[ 9, 10, 11]])]

### np.vsplit()，沿纵向轴拆分（axis=0）

np.vsplit(b, 2)

[array([[ 0, 1, 20, 3, 4, 5]]), array([[ 6, 7, 8, 9, 10, 11]])]

np.split(b,2,axis=0)

[array([[ 0, 1, 20, 3, 4, 5]]), array([[ 6, 7, 8, 9, 10, 11]])]

### np.dsplit(arr\_dstack,2)，深度拆分

arr\_dstack

array([[[ 0, 0],

[ 1, 2],

[20, 40],

[ 3, 6],

[ 4, 8],

[ 5, 10]],

[[ 6, 12],

[ 7, 14],

[ 8, 16],

[ 9, 18],

[10, 20],

[11, 22]]])

np.dsplit(arr\_dstack,2)

[array([[[ 0],

[ 1],

[20],

[ 3],

[ 4],

[ 5]],

[[ 6],

[ 7],

[ 8],

[ 9],

[10],

[11]]]), array([[[ 0],

[ 2],

[40],

[ 6],

[ 8],

[10]],

[[12],

[14],

[16],

[18],

[20],

[22]]])]

拆分的结果是原来的三维数组拆分成为两个二维数组。

这个烧脑的拆分过程可以自行分析下~~

# Numpy数组的切片和索引

## 一维数组

一维数组的切片和索引与python的list索引类似。

a = np.arange(7)

a

array([0, 1, 2, 3, 4, 5, 6])

a[1:4]

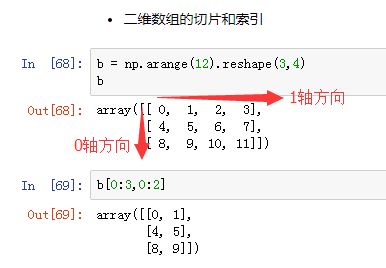
array([1, 2, 3])

# 每间隔2个取一个数

a[ : 6: 2]

array([0, 2, 4])

## 二维数组



# Numpy数组的类型转换

## 数组转换成list，使用tolist()

b

array([[ 0, 1, 20, 3, 4, 5],

[ 6, 7, 8, 9, 10, 11]])

b.tolist()

[[0, 1, 20, 3, 4, 5], [6, 7, 8, 9, 10, 11]]

## 转换成指定类型，astype()函数

b.astype(float)

array([[ 0., 1., 20., 3., 4., 5.],

[ 6., 7., 8., 9., 10., 11.]])

# Numpy数组的常用统计函数

请注意函数在使用时需要指定axis轴的方向，若不指定，默认统计整个数组。

| **运算** | **结果** |
| --- | --- |
| np.sum() | 返回求和 |
| np.mean() | 返回均值 |
| np.max() | 返回最大值 |
| np.min() | 返回最小值 |
| np.ptp() | 返回最大值减去最小值，即（max-min） |
| np.std() | 返回标准偏差（standard deviation） |
| np.var() | 返回方差（variance） |
| np.cumsum() | 返回累加值 |
| np.cumprod() | 返回累乘积值 |

# 数组的广播

当数组跟一个标量进行数学运算时，标量需要根据数组的形状进行扩展，然后执行运算。这个扩展的过程称为“广播（broadcasting）”