第一章 NumPy

目录

NumPy 简介

import **numpy** as np

NumPy

——**Numerical Python**,是高性能科学计算和数据分析的基础包。

它是一个由多维数组对象和相关函数组成的库。

官方网站：http://www.numpy.org/

NumPy 简介

部分功能：

- **ndarray** ，具有矢量算术运算和复杂广播能力的快速且节省空间的多维数组。
- 用于对数组数据进行快速运算的标准数学函数。
- 用于读写磁盘数据的工具以及用于操作内存映射文件的工具。
- 线性代数、随机数生成以及傅里叶变换功能。
- 用于集成由C、C++、Fortran等语言编写的代码的工具。

NumPy 环境

安装：

-pip install numpy

验证NumPy环境：

```
import numpy as np

print(np.array([1, 2, 3]))
```

目录

ndarray

——N维数组对象（整个库的核心对象）

- 相同类型的元素集合。
- 可以使用基于零的索引访问集合中的项目。

Ndarray 对象

- 基本的ndarray是使用 NumPy 中的array函数创建
  - numpy.array(object, dtype=None, copy=True, order=None, subok=False, ndmin = 0)
  - 构造器参数如下：

序号	参数及描述
1.	object 任何暴露数组接口方法的对象都会返回一个数组或任何(嵌套)序列。
2.	dtype 数组的所需数据类型，可选。
3.	copy 可选，默认为true，对象是否被复制。
4.	order C(按行)、F(按列)或A(任意，默认)。
5.	subok 默认情况下，返回的数组被强制为基类数组。 如果为true，则返回子类。
6.	ndmin 指定返回数组的最小维数。

Ndarray 对象示例

- 示例1

```
import numpy as np
a = np.array([1,2,3])
print(a)
```

- 示例2

```
# 多维数组的创建
import numpy as np
a = np.array([[1,  2],  [3,  4]])
print(a)
```

Ndarray 对象示例

- 示例3

```
# 最小维度参数的用法
import numpy as np
a = np.array([1, 2, 3, 4, 5], ndmin=2)
print(a)
```

- 示例4

```
# dtype 参数
import numpy as np

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

Notebook中，如何获得函数的用法？

NumPy - 数据类型

- NumPy 支持比 Python 更多种类的数值类型。 下表显示了 NumPy 中定义的不同标量数据类型。

序号	数据类型及描述
1.	bool_存储为一个字节的布尔值(真或假)
2.	int_默认整数，相当于 C 的long，通常为int32或int64
3.	intc相当于 C 的int，通常为int32或int64
4.	intp用于索引的整数，相当于 C 的size_t，通常为int32或int64
5.	int8字节(-128 ~ 127)
6.	int1616 位整数(-32768 ~ 32767)
7.	int3232 位整数(-2147483648 ~ 2147483647)
8.	int6464 位整数(-9223372036854775808 ~ 9223372036854775807)
9.	uint88 位无符号整数(0 ~ 255)
10.	uint1616 位无符号整数(0 ~ 65535)

序号	数据类型及描述
11.	uint3232 位无符号整数(0 ~ 4294967295)
12.	uint6464 位无符号整数(0 ~ 18446744073709551615)
13.	float_float64的简写
14.	float16半精度浮点：符号位，5 位指数，10 位尾数
15.	float32单精度浮点：符号位，8 位指数，23 位尾数
16.	float64双精度浮点：符号位，11 位指数，52 位尾数
17.	complex_complex128的简写
18.	complex64复数，由两个 32 位浮点表示(实部和虚部)
19.	complex128复数，由两个 64 位浮点表示(实部和虚部)

NumPy - 数组属性

- **ndarray.dtype：**返回一个用于说明数组数据类型的对象。
- **ndarray.shape：**返回一个包含数组维度的元组，也可以用于调整数组大小。
- **ndarray.ndim：**返回数组的维数。
- **ndarray.itemsize：**返回数组中每个元素的单位长度。

NumPy - 数组属性

- 示例 1

```
# 得到数组的结构
import numpy as np

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

- 示例 2

```
# 调整数组大小
import numpy as np

a = np.array([1, 2, 3, 4, 5, 6])
a.shape = (3, 2)  #b=a.reshape(3,2)
print(a)
```

NumPy - 数组属性

- 示例 3

```
# 一维数组
import numpy as np
a = np.arange(8)
print(a.ndim)
# 现在调整其大小
b = a.reshape(2, 4)
print(b.ndim)
```

- 示例 4

```
# 数组的 dtype 为 int8(一个字节)
import numpy as np

x = np.array([1, 2, 3, 4, 5], dtype=np.int32)
print(x.itemsize)
```

NumPy - 数组创建

- **numpy.empty**可以创建指定形状的未初始化数组。

  numpy.empty(shape, dtype = float, order = 'C’)

- **numpy.zeros**返回特定大小，以 0 填充的新数组。
  - numpy.zeros(shape, dtype = float, order = 'C’)
- **numpy.ones**返回特定大小，以 1 填充的新数组。
  - numpy.ones(shape, dtype = None, order = ‘C’)
- **numpy.full**返回特定大小，以指定内容填充的新数组。
  - np.full(shape, fill_value, dtype=None, order='C')

NumPy - 数组创建

- **np.random.randint**返回范围内的随机整数数组。
  - numpy.random.randint(low, high= None, size = None, dtype =int)

- **np.random.random**返回范围内的随机浮点数数组。
  - numpy.random.randint(low, high= None, size = None, dtype =int)

NumPy –固定数值范围的数组

- numpy.arange这个函数返回ndarray对象，包含给定范围内的等间隔值。
  - numpy.arange(start, stop, step, dtype)
  - 构造器接受下列参数：

序号	参数及描述
1.	start 范围的起始值，默认为0
2.	stop 范围的终止值(不包含)
3.	step 两个值的间隔，默认为1
4.	dtype 返回ndarray的数据类型，如果没有提供，则会使用输入数据的类型。

  - 实例

```
# 设置了起始值和终止值参数
import numpy as np

x = np.arange(10, 20, 2)
print(x)
```

NumPy –固定数值范围的数组

- numpy.linspace指定了范围之间的均匀间隔数量，而不是步长。
  - numpy.linspace(start, stop, num, endpoint, retstep, dtype)
  - 构造器接受下列参数：

序号	参数及描述
1.	start 序列的起始值
2.	stop 序列的终止值，如果endpoint为true，该值包含于序列中
3.	num 要生成的等间隔样例数量，默认为50
4.	endpoint 序列中是否包含stop值，默认为ture
5.	retstep 如果为true，返回样例，以及连续数字之间的步长
6.	dtype 输出ndarray的数据类型

NumPy –固定数值范围的数组

- numpy.logspace此函数返回一个ndarray对象，其中包含在对数刻度上均匀分布的数字。 刻度的开始和结束端点是某个底数的幂，通常为 10。
  - numpy.logscale(start, stop, num, endpoint, base, dtype)
  - 构造器接受下列参数：

序号	参数及描述
1.	start 起始值是base ** start
2.	stop 终止值是base ** stop
3.	num 范围内的数值数量，默认为50
4.	endpoint 如果为true，终止值包含在输出数组当中
5.	base 对数空间的底数，默认为10
6.	dtype 输出数组的数据类型，如果没有提供，则取决于其它参数

目录

NumPy - 切片和索引

- 常见的三种索引方法类型：字段访问，基本切片和高级索引。
- **字段访问**，遵从内置Python的索引切片访问方法：

```
import numpy as np

x = np.arange(1, 20, 2)
print(x)
print(x[3])
print(x[:4])
print(x[2:5])
print(x[:-3])
print(x[1:5:2])
```

NumPy - 切片和索引

- **基本切片**， Python 中切片概念到 n 维的扩展。

通过将start，stop和step参数提供给内置的slice函数来构造一个 slice对象。 此slice对象被传递给数组来提取数组的一部分。

```
import numpy as np

x = np.arange(0, 20)
s = slice(2, 7, 2)
print(x[s])
x.shape = (4, 5)
print(x)
s1 = slice(2, 4)
s2 = slice(0, 1)
print(x[..., s2])
```

NumPy - 切片和索引

- 高级索引可以索引非元组序列，元素数据类型为整数或布尔值，或者至少一个元素为序列对象的元组的ndarray。
- 整数索引基于 N 维索引来获取数组中任意元素。

```
import numpy as np

x = np.array([[1, 2], [3, 4], [5, 6]])
y = x[[0, 1, 2], [0, 1, 0]]
print(y)    #返回x[0,0],x[1,1],x[2,0]
```

NumPy - 切片和索引

- 高级索引中的布尔索引。当布尔运算(例如比较运算符)结果为真时，索引对应数据。

```
import numpy as np

x = np.array([[0, 1, 2], [3, 4, 5], [6, 7,
8], [9, 10, 11]])
print(x, '\n')
# 打印出大于 5 的元素
print('大于 5 的元素是: ', x[x > 5])
```

实验一

- 请用两种方法创建一个边界值为1而内部是0的二维数组

```
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

实验一

```
#方法一

import numpy as np
Z = np.ones((10, 10))
Z[1:-1, 1:-1] = 0
print(Z)
```

```
#方法二

import numpy as np
Z = np.ones((10, 10))
Z[1:9, 1:9] = 0
print(Z)
```

实验二

- 创建一个8×8的棋盘格

```
[[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]]
```

```
import numpy as np
Z = np.zeros((8, 8), dtype=int)
Z[1::2, ::2] = 1
Z[::2, 1::2] = 1
print(Z)
```

如何交换数组np.arange(9).reshape(3,3)中的第1列和第2列？

如何交换数组np.arange(9).reshape(3,3)中的第1行和第2行？

如何从数组np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])中提取所有的奇数？

如何从数组np.arange(15)中提取5到10之间的所有数字？

思考

```
a = np.array([1, 2, 3])
b = a * 3
print(b)
```

```
[3 6 9]
```

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

```
[[ 2  2  9]
 [ 2  4 12]]
```

```
a = np.array([[0, 0, 0],[1, 1, 1],[2, 2, 2], [3, 3, 3]])
b = np.array([1, 2, 3])
c = a + b
print(c)
```

```
[[1 2 3]
 [2 3 4]
 [3 4 5]
 [4 5 6]]
```

广播的主要形式

1、数组维度不同，后缘维度的轴长相符

2、数组维度相同，其中有个轴为1

广播

广播：

- 对形状不同的数组，采取的运算操作。
- 当两个数组的形状并不相同的时候，我们可以通过扩展数组的方法来实现相加、相减、相乘等操作。

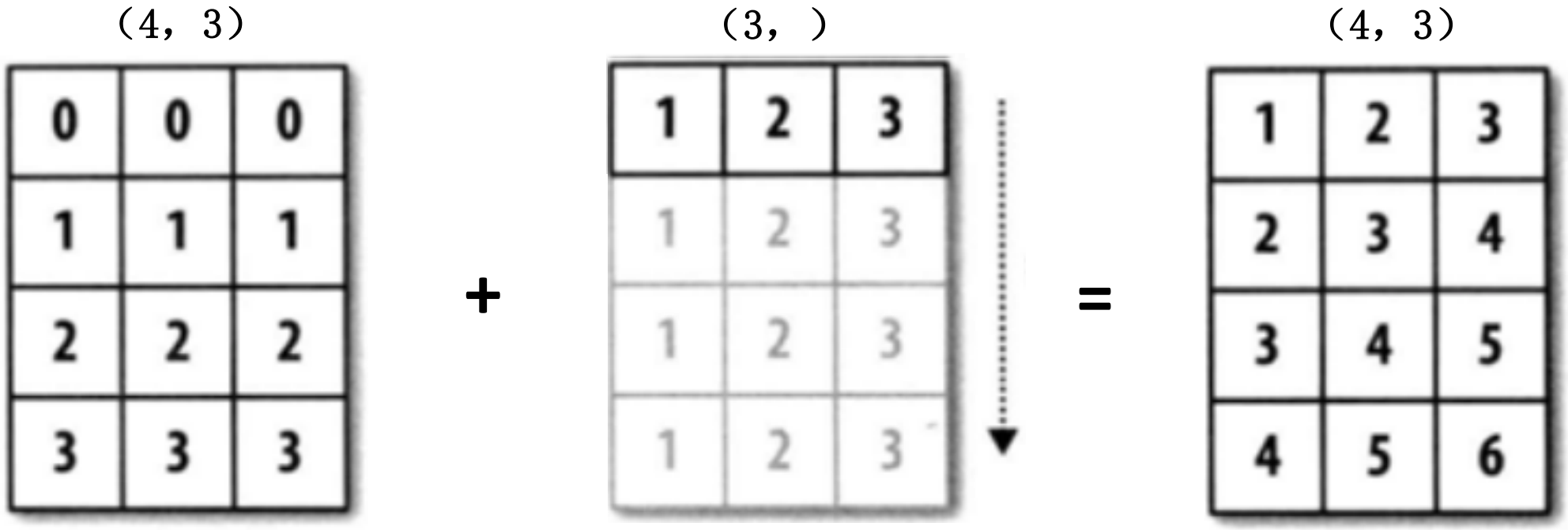
```
arr = np.random.randn(4, 3) #shape(4, 3)
arr_mean = arr.mean(0) #shape(3, )
demeaned = arr -arr_mean
```

广播的主要形式

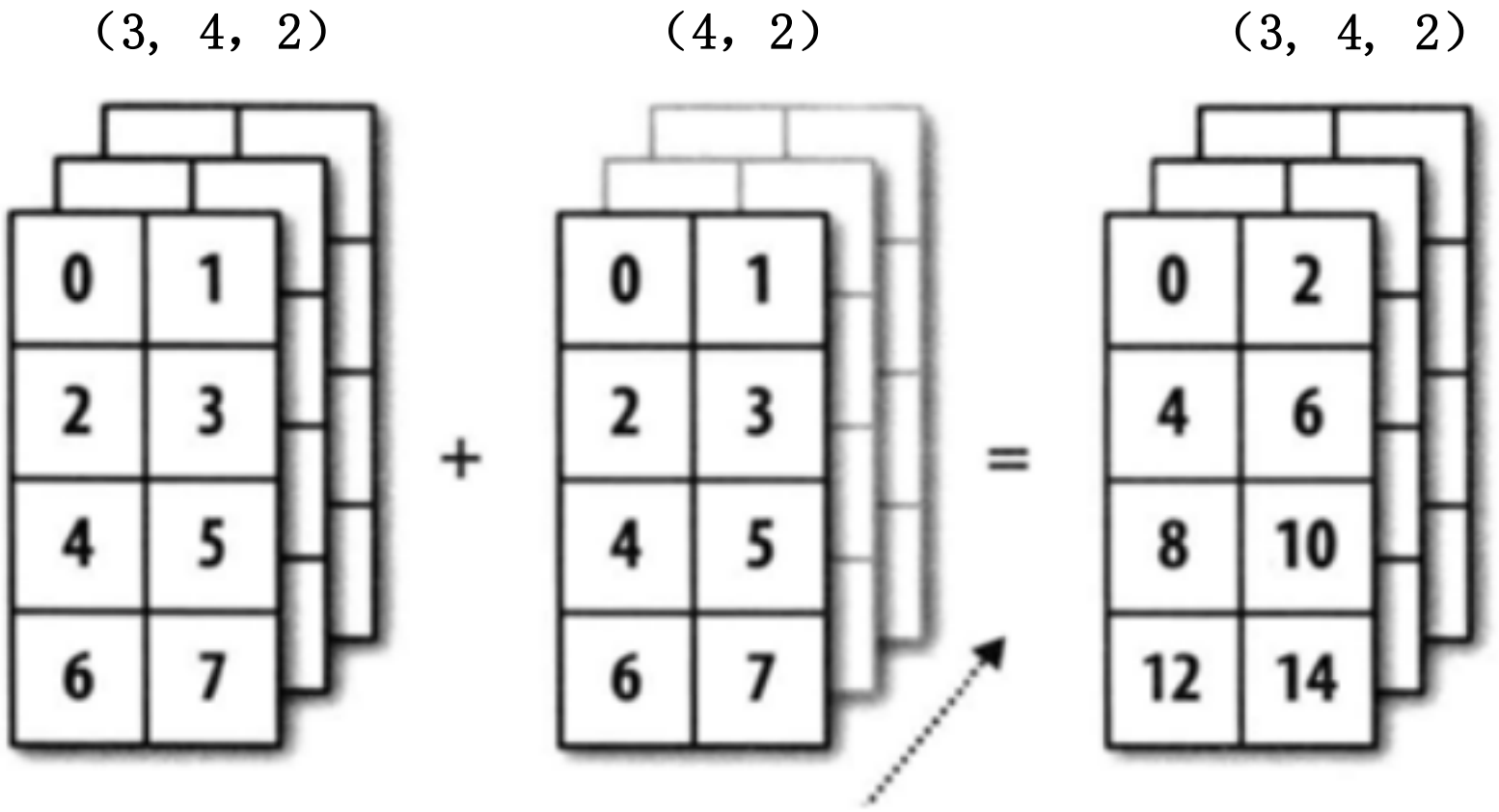
1、数组维度不同，后缘维度的轴长相符

2、数组维度相同，其中有个轴为1

广播的主要形式



广播的主要形式



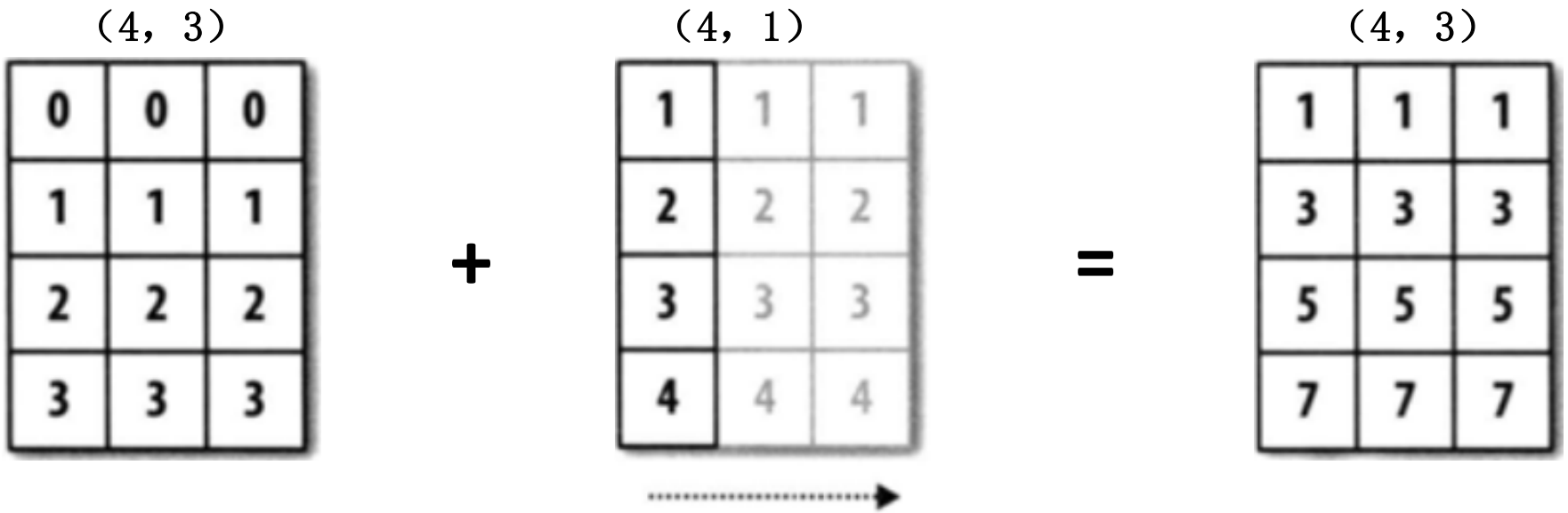
广播的主要形式

1、数组维度不同，后缘维度的轴长相符

2、数组维度相同，其中有个轴为1

广播的主要形式

```
arr1 = np.array([[0, 0, 0],[1, 1, 1],[2, 2, 2], [3, 3, 3]])
arr2 = np.array([[1],[2],[3],[4]])
arr_sum = arr1 + arr2
print(arr_sum)
```



广播的原则

广播的原则：

如果两个数组的后缘维度（trailing dimension，即从末尾开始算起的维度）的轴长度相符，或其中的一方的长度为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。

广播的应用

广播机制的概念与应用，不仅仅适用于Numpy，

在TensorFlow, PyTorch, MxNet中同样适用。

练习

如何创建一个5*5的矩阵，每行的数值都是从0到4？

结果：
```
[[0., 1., 2., 3., 4.],
 [0., 1., 2., 3., 4.],
 [0., 1., 2., 3., 4.],
 [0., 1., 2., 3., 4.],
 [0., 1., 2., 3., 4.]]
```

练习

如何从二维数组a_2d = np.array([[3,3,3],[4,4,4],[5,5,5]])中的每行减去一维数组b_1d = np.array([1,2,3])中相应的值？

结果：
```
[[2, 2, 2],
 [2, 2, 2],
 [2, 2, 2]]
```

NumPy - 数组上的迭代

- NumPy 包含一个迭代器对象numpy.nditer。 它是一个有效的多维迭代器对象，可以用于在数组上进行迭代。 数组的每个元素可使用 Python 的标准Iterator接口来访问。

```
import numpy as np

a = np.arange(0, 60, 5)
a = a.reshape(3, 4)
print(a)

for x in np.nditer(a):
    print(x)
```

NumPy - 数组上的迭代

- 如果两个数组是可广播的，nditer组合对象能够同时迭代它们。 假设数组a具有维度 3X4，并且存在维度为 1X4 的另一个数组b，则使用以下类型的迭代器(数组b被广播到a的大小)。

```
import numpy as np

a = np.arange(0, 60, 5)
a = a.reshape(3, 4)
print(a)
b = np.array([1, 2, 3, 4], dtype=int)
print(b)
for x, y in np.nditer([a, b]):
    print(x, y)
```

目录

Numpy函数

- 为了方便对于NumPy中ndarray的操作，NumPy集成了大量的常量及函数操作。NumPy中的常量如下：
  - numpy.Inf
  - numpy.NAN
  - numpy.Infinity
  - numpy.MAXDIMS
  - numpy.NINF
  - numpy.NZERO
  - …

Numpy函数

- 为了方便对于NumPy中ndarray的操作，NumPy集成了大量的函数操作。主要分为以下几个类别：
  - 数组函数
  - 字符串函数
  - 算数函数
  - 统计函数
  - 排序搜索函数
  - …

数组操作函数

- NumPy中常见的用于处理ndarray对象元素的函数主要分为如下类型：
  - 修改数组形状
  - 反转操作
  - 数组连接
  - 数组分割
  - 元素操作
  - …

数组形状修改函数

- NumPy中数组形状修改的函数主要有如下四个：
  - reshape 不改变数据的条件下修改形状
  - flat 数组上的一维迭代器
  - flatten 返回折叠为一维的数组副本
  - ravel 返回连续的展开数组

数组形状修改函数

- ndarray.reshape 函数在不改变数据的条件下修改形状，参数如下：
  - ndarray.reshape(arr, newshape, order)
  - 其中：
    - arr：要修改形状的数组
    - newshape：整数或者整数数组，新的形状应当兼容原有形状
    - order：'C'为 C 风格顺序，'F'为 F 风格顺序，'A'为保留原顺序。

```
import numpy as np

a = np.arange(8)
print(a)

b = a.reshape(4, 2)
print(b)
```

数组形状修改函数

- ndarray.flat 函数返回数组上的一维迭代器，行为类似 Python 内建的迭代器。

```
import numpy as np

a = np.arange(8).reshape(2, 4)
print(a)

# 返回展开数组中的下标的对应元素
print(list(a.flat))
```

数组形状修改函数

- ndarray.flatten 函数返回折叠为一维的数组副本，函数接受下列参数：
  - ndarray.flatten(order)其中：order：'C' — 按行，'F' — 按列，'A' — 原顺序，'k' — 元素在内存中的出现顺序。

```
import numpy as np

a = np.arange(8).reshape(2, 4)

print(a)
# default is column-major

print(a.flatten())
print(a.flatten(order='F'))
```

翻转操作函数

- NumPy中数组翻转操作的函数主要有如下四个：
  - transpose 翻转数组的维度
  - ndarray.T 和self.transpose()相同
  - rollaxis 向后滚动指定的轴
  - swapaxes 互换数组的两个轴

翻转操作函数

- numpy.transpose 函数翻转给定数组的维度。如果可能的话它会返回一个视图。函数接受下列参数：
  - numpy.transpose(arr, axes)
  - 其中：
    - arr：要转置的数组
    - axes：整数的列表，对应维度，通常所有维度都会翻转。

```
import numpy as np

a = np.arange(12).reshape(3, 4)

print(a)
print(np.transpose(a))
```

翻转操作函数

- numpy.ndarray.T该函数属于ndarray类，行为类似于numpy.transpose.

```
import numpy as np

a = np.arange(12).reshape(3, 4)

print(a)
print(a.T)
```

翻转操作函数

- numpy.swapaxes 函数交换数组的两个轴。这个函数接受下列参数：
  - numpy.swapaxes(arr, axis1, axis2)
  - 参数：
    - arr：要交换其轴的输入数组
    - axis1：对应第一个轴的整数
    - axis2：对应第二个轴的整数

```
import numpy as np

a = np.arange(8).reshape(2, 2, 2)

print(a)
print(np.swapaxes(a, 2, 0))
```

读程序

```
a = np.arange(0,60,5)
a = a.reshape(3,4)
print ('原始数组是: ')
print (a)
print ('\n')
print ('原始数组的转置是: ')
b = a.T
print (b)
print ('\n')
```

读程序

```
print ('以 C 风格顺序排序: ')
c = b.copy(order='C')
print (c)
for x in np.nditer(c):
    print (x, end=", " )
print  ('\n')
print  ('以 F 风格顺序排序: ')
c = b.copy(order='F')
print (c)
for x in np.nditer(c):
    print (x, end=", " )
```

翻转操作函数

- numpy.rollaxis 函数向后滚动特定的轴，直到一个特定位置。这个函数接受三个参数：
  - numpy.rollaxis(arr, axis, start)
  - 其中：
    - arr：输入数组
    - axis：要向后滚动的轴，其它轴的相对位置不会改变
    - start：默认为零，表示完整的滚动。会滚动到特定位置。

```
import numpy as np

a = np.arange(8).reshape(2,2,2)

print(a)
print(np.rollaxis(a,2))
print(np.rollaxis(a,2,1))
```

修改维度函数

- NumPy中修改维度函数主要有如下四个：
  - broadcast_to 将数组广播到新形状
  - expand_dims 扩展数组的形状
  - squeeze 从数组的形状中删除单维条目

修改维度函数

- numpy.broadcast_to 函数将数组广播到新形状。 它在原始数组上返回只读视图。 它通常不连续。 如果新形状不符合 NumPy 的广播规则，该函数可能会抛出ValueError。该函数接受以下参数：
  - numpy.broadcast_to(array, shape, subok)

```
import numpy as np
a = np.arange(4).reshape(1,4)

print(a)
print(np.broadcast_to(a,(4,4)))
```

修改维度函数

- numpy.expand_dims 函数通过在指定位置插入新的轴来扩展数组形状。该函数需要两个参数：
  - numpy.expand_dims(arr, axis)
  - 其中：
    - arr：输入数组
    - axis：新轴插入的位置

```
import numpy as np

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

y = np.expand_dims(x, axis=0)
print(y)
print(x.shape, y.shape)

y = np.expand_dims(x, axis=1)
print(y)
print(x.ndim, y.ndim)
print(x.shape, y.shape)
```

修改维度函数

- numpy.squeeze 函数从给定数组的形状中删除一维条目。 此函数需要两个参数。
  - numpy.squeeze(arr, axis)
  - 其中：
    - arr：输入数组
    - axis：整数或整数元组，用于选择形状中单一维度条目的子集

```
import numpy as np

x = np.arange(9).reshape(1, 3, 3)
print(x)
y = np.squeeze(x)
print(y)
print(x.shape, y.shape)
```

数组的连接操作

- NumPy中数组的连接函数主要有如下四个：
  - concatenate 沿着现存的轴连接数据序列
  - stack 沿着新轴连接数组序列
  - hstack 水平堆叠序列中的数组(列方向)
  - vstack 竖直堆叠序列中的数组(行方向)

数组的连接操作

- numpy.stack 函数沿新轴连接数组序列，需要提供以下参数：
  - numpy.stack(arrays, axis)
  - 其中：
    - arrays：相同形状的数组序列
    - axis：返回数组中的轴，输入数组沿着它来堆叠

```
import numpy as np
a = np.array([[1,2],[3,4]])
print(a)
b = np.array([[5,6],[7,8]])
print(b)
print(np.stack((a,b),0))
print(np.stack((a,b),1))
```

数组的连接操作

- numpy.hstack是numpy.stack函数的变体，通过堆叠来生成水平的单个数组。

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print(a)
b = np.array([[5, 6], [7, 8]])
print(b)
print('水平堆叠: ')
c = np.hstack((a, b))
print(c)
```

数组的连接操作

- numpy.vstack是numpy.stack函数的变体，通过堆叠来生成竖直的单个数组。

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print(a)
b = np.array([[5, 6], [7, 8]])
print(b)
print('竖直堆叠: ')
c = np.vstack((a, b))
print(c)
```

数组的连接操作

- numpy.concatenate 函数用于沿指定轴连接相同形状的两个或多个数组。该函数接受以下参数。
  - numpy.concatenate((a1, a2, ...), axis)
  - 其中：
    - a1, a2, ...：相同类型的数组序列
    - axis：沿着它连接数组的轴，默认为 0

```
import numpy as np
a = np.array([[1,2],[3,4]])
print(a)
b = np.array([[5,6],[7,8]])
print(b)
print(np.concatenate((a,b)))
print(np.concatenate((a,b),axis = 1))
```

数组的分割操作

- NumPy中数组的数组分割函数主要如下：
  - split 将一个数组分割为多个子数组
  - hsplit 将一个数组水平分割为多个子数组(按列)
  - vsplit 将一个数组竖直分割为多个子数组(按行)

数组的分割操作

- numpy.split该函数沿特定的轴将数组分割为子数组。函数接受三个参数：
  - numpy.split(ary, indices_or_sections, axis)
    - ary：被分割的输入数组
    - indices_or_sections：可以是整数，表明要从输入数组创建的，等大小的子数组的数量。 如果此参数是一维数组，则其元素表明要创建新子数组的点。
    - axis：默认为 0

```
import numpy as np
a = np.arange(9)
print(a)
print('将数组分为三个大小相等的子数组：')
b = np.split(a,3)
print(b)
print('将数组在一维数组中表明的位置分割：')
b = np.split(a,[4,7])
print(b)
```

数组的分割操作

- numpy.hsplit是split()函数的特例，其中轴为 1 表示水平分割。

```
import numpy as np

a = np.arange(16).reshape(4,4)
print(a)
print('水平分割: ')
b = np.hsplit(a,2)
print(b)
```

数组的分割操作

- numpy.vsplit是split()函数的特例，其中轴为 0 表示竖直分割，无论输入数组的维度是什么。

```
import numpy as np

a = np.arange(16).reshape(4,4)
print(a)
print('竖直分割：')
b = np.vsplit(a,2)
print(b)
```

数组元素操作

- NumPy中数组操作函数主要如下：
  - resize 返回指定形状的新数组
  - append 将值添加到数组末尾
  - insert 沿指定轴将值插入到指定下标之前
  - delete 返回删掉某个轴的子数组的新数组
  - unique 寻找数组内的唯一元素

数组元素操作

- numpy.resize 函数返回指定大小的新数组。 如果新大小大于原始大小，则包含原始数组中的元素的重复副本。 该函数接受以下参数：
  - numpy.resize(arr, shape)
  - 其中：
    - arr：要修改大小的输入数组
    - shape：返回数组的新形状

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a)
print(a.shape)
b = np.resize(a, (3,2))
print(b)
print(b.shape)
print('修改第二个数组的大小：')
b = np.resize(a,(3,3))
print(b)
```

数组元素操作

- numpy.append 函数在输入数组的末尾添加值。 附加操作不是原地的，而是分配新的数组。 此外，输入数组的维度必须匹配否则将生成 ValueError。函数接受下列函数：
  - numpy.append(arr, values, axis)
  - 其中：
    - arr：输入数组
    - values：要向arr添加的值，比如和arr形状相同(除了要添加的轴)
    - axis：沿着它完成操作的轴。如果没有提供，两个参数都会被展开。

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a)
print(np.append(a, [[7,8,9]],axis = 0))
print(np.append(a, [[5,5,5],[7,8,9]],axis = 1))
```

数组元素操作

- numpy.insert 函数在给定索引之前，沿给定轴在输入数组中插入值。 如果值的类型转换为要插入，则它与输入数组不同。 插入没有原地的，函数会返回一个新数组。 此外，如果未提供轴，则输入数组会被展开。insert()函数接受以下参数：
  - numpy.insert(arr, obj, values, axis)
    - arr：输入数组
    - obj：在其之前插入值的索引
    - values：要插入的值
    - axis：沿着它插入的轴

```
import numpy as np
a = np.array([[1,2],[3,4],[5,6]])
print(a)
print(np.insert(a,3,[11,12]))
print(np.insert(a,1,[11],axis = 0))
print(np.insert(a,1,[11],axis = 1))
```

数组元素操作

- numpy.delete 函数返回从输入数组中删除指定子数组的新数组。 与insert()函数的情况一样，如果未提供轴参数，则输入数组将展开。 该函数接受以下参数：
  - Numpy.delete(arr, obj, axis)
    - arr：输入数组
    - obj：可以被切片，整数或者整数数组，表明要从输入数组删除的子数组
    - axis：沿着它删除给定子数组的轴

```
import numpy as np
a = np.array([[1,2],[3,4],[5,6]])
print(a)
print(np.delete(a,5))
print(np.delete(a,1,axis = 1))
```

数组元素操作

- numpy.unique 函数返回输入数组中的去重元素数组。 该函数能够返回一个元组，包含去重数组和相关索引的数组。 索引的性质取决于函数调用中返回参数的类型。
  - numpy.unique(arr, return_index, return_inverse, return_counts)
    - arr：输入数组，如果不是一维数组则会展开
    - return_index：如果为true，返回输入数组中的元素下标
    - return_inverse：如果为true，返回去重数组的下标，它可以用于重构输入数组
    - return_counts：如果为true，返回去重数组中的元素在原数组中的出现次数

```
import numpy as np
a = np.array([5,2,6,2,7,5,6,8,2,9])
u = np.unique(a)
u,indices = np.unique(a, return_index = True)
u,indices = np.unique(a,return_inverse = True)
u,indices = np.unique(a,return_counts = True)
```

NumPy - 字符串函数

- 以下函数用于对dtype为numpy.string_或numpy.unicode_的数组执行向量化字符串操作。 它们基于 Python 内置库中的标准字符串函数。字符数组类(numpy.char)中定义。

序号	函数及描述
1.	add() 返回两个str或Unicode数组的逐个字符串连接
2.	multiply() 返回按元素多重连接后的字符串
3.	center() 返回给定字符串的副本，其中元素位于特定字符串的中央
4.	capitalize() 返回给定字符串的副本，其中只有第一个字符串大写
5.	title() 返回字符串或 Unicode 的按元素标题转换版本
6.	lower() 返回一个数组，其元素转换为小写
7.	upper() 返回一个数组，其元素转换为大写
8.	split() 返回字符串中的单词列表，并使用分隔符来分割
9.	splitlines() 返回元素中的行列表，以换行符分割
10.	strip() 返回数组副本，其中元素移除了开头或者结尾处的特定字符
11.	join() 返回一个字符串，它是序列中字符串的连接
12.	replace() 返回字符串的副本，其中所有子字符串的出现位置都被新字符串取代
13.	decode() 按元素调用str.decode
14.	encode() 按元素调用str.encode

NumPy - 字符串函数

```
import numpy as np
print(np.char.add(['hello'],[' xyz']))
print(np.char.add(['hello', 'hi'],[' abc', ' xyz']))
print(np.char.multiply('Hello ',3))
print(np.char.center('hello', 20,fillchar = '*'))
print(np.char.capitalize('hello world'))
print(np.char.title('hello how are you?'))
print(np.char.lower(['HELLO','WORLD']))
print(np.char.lower('HELLO'))
print(np.char.upper('hello'))
print(np.char.upper(['hello','world']))
print(np.char.split ('hello how are you?'))
print(np.char.split ('YiibaiPoint,Hyderabad,Telangana', sep = ','))
print(np.char.splitlines('hello\nhow are you?'))
print(np.char.splitlines('hello\rhow are you?'))
print(np.char.strip('ashok arora','a'))
print(np.char.strip(['arora','admin','java'],'a'))
print(np.char.join(':','dmy'))
print(np.char.join([':','-'],['dmy','ymd']))
print(np.char.replace ('He is a good boy', 'is', 'was'))
a = np.char.encode('hello', 'cp500')
print(a)
print(np.char.decode(a,'cp500'))
```

NumPy - 算数函数

- NumPy 包含大量的各种数学运算功能。 NumPy 提供标准的三角函数，算术运算的函数，复数处理函数等。
  - 三角函数
  - 舍入函数
  - 算数函数

NumPy -三角函数

- NumPy 拥有标准的三角函数，它为弧度制单位的给定角度返回三角函数比值。arcsin，arccos，和arctan函数返回给定角度的sin，cos和tan的反三角函数。 这些函数的结果可以通过numpy.degrees()函数通过将弧度制转换为角度制来验证。

```
import numpy as np
a = np.array([0,30,45,60,90])
# 通过乘 pi/180 转化为弧度
print(np.sin(a*np.pi/180))
print(np.cos(a*np.pi/180))
print(np.tan(a*np.pi/180))
```

NumPy -舍入函数

- numpy.around()这个函数返回四舍五入到所需精度的值
  - numpy.around(a,decimals)
  - a 输入数组
  - decimals 要舍入的小数位数。 默认值为0。 如果为负，整数将四舍五入到小数点左侧的位置
- numpy.floor() 函数返回不大于输入参数的最大整数。
- numpy.ceil() 函数返回输入值的上限，大于输入参数的最小整数。

```
import numpy as np
a = np.array([1.0, 5.55, 123, 0.567, 25.532])
print(np.around(a))
print(np.around(a, decimals=1))
print(np.floor(a))
print(np.ceil(a))
```

NumPy - 算数运算

- 用于执行算术运算(如add()，subtract()，multiply()和divide())的输入数组必须具有相同的形状或符合数组广播规则。
  - numpy.reciprocal() 函数返回参数逐元素的倒数。
  - numpy.power() 函数将第一个输入数组中的元素作为底数，计算它与第二个输入数组中相应元素的幂。
  - numpy.mod() 函数返回输入数组中相应元素的除法余数。

```
import numpy as np
a = np.array([0.25,  2,  1,  0.2,  100])
print(np.reciprocal(a))
print(np.power(a,2))
print(np.power(a,2))
a = np.array([10,20,30])
b = np.array([3,5,7])
print(np.mod(a,b))
```

NumPy - 统计函数

- NumPy 有很多有用的统计函数，用于从数组中给定的元素中查找最小，最大，百分标准差和方差等。
  - numpy.amin() , numpy.amax() 从给定数组中的元素沿指定轴返回最小值和最大值。
  - numpy.ptp() 函数返回沿轴的值的范围(最大值 - 最小值)。
  - numpy.percentile() 表示小于这个值得观察值占某个百分比
    - numpy.percentile(a, q, axis)
    - a 输入数组;q 要计算的百分位数，在 0 ~ 100 之间;axis 沿着它计算百分位数的轴
  - numpy.median() 返回数据样本的中位数。
  - numpy.mean() 沿轴返回数组中元素的算术平均值。
  - numpy.average() 返回由每个分量乘以反映其重要性的因子得到的加权平均值

NumPy - 统计函数

```
import numpy as np
a = np.array([[3,7,5],[8,4,3],[2,4,9]])
print(np.amin(a,1))
print(np.amax(a,1))
print(np.ptp(a))
print(np.percentile(a,50))
print(np.median(a))
print(np.mean(a))
print(np.average(a))
print(np.std([1,2,3,4])) #返回数组标准差
print(np.var([1,2,3,4])) #返回数组方差
```

NumPy排序、搜索和计数函数

- NumPy中提供了各种排序相关功能。
  - numpy.sort函数返回输入数组的排序副本。 numpy.sort(a, axis, kind, order)
    - a 要排序的数组;
    - axis 沿着它排序数组的轴，如果没有数组会被展开，沿着最后的轴排序;
    - kind 默认为'quicksort'(快速排序);
    - order 如果数组包含字段，则是要排序的字段
  - numpy.argsort() 函数对输入数组沿给定轴执行间接排序，并使用指定排序类型返回数据的索引数组。 这个索引数组用于构造排序后的数组。
  - numpy.lexsort()函数使用键序列执行间接排序。 键可以看作是电子表格中的一列。该函数返回一个索引数组，使用它可以获得排序数据。 注意，最后一个键恰好是 sort 的主键。
  - numpy.argmax() 和 numpy.argmin()这两个函数分别沿给定轴返回最大和最小元素的索引。

NumPy排序、搜索和计数函数

- NumPy中提供了各种排序相关功能。
  - numpy.nonzero() 函数返回输入数组中非零元素的索引。
  - numpy.where() 函数返回输入数组中满足给定条件的元素的索引。
  - numpy.extract() 函数返回满足任何条件的元素。

```
import numpy as np
a = np.array([[3, 7, 3, 1], [9, 7, 8, 7]])
print(np.sort(a))
print(np.argsort(a))
print(np.argmax(a))
print(np.argmin(a))
print(np.nonzero(a))
print(np.where(a > 3))
nm = ('raju', 'anil', 'ravi', 'amar')
dv = ('f.y.', 's.y.', 's.y.', 'f.y.')
print(np.lexsort((dv, nm)))
```

NumPy IO文件操作

- ndarray对象可以保存到磁盘文件并从磁盘文件加载。 可用的 IO 功能有：
  - numpy.save() 文件将输入数组存储在具有npy扩展名的磁盘文件中。
  - numpy.load() 从npy文件中重建数组。
  - numpy.savetxt()和numpy.loadtxt() 函数以简单文本文件格式存储和获取数组数据。

```
import numpy as np
a = np.array([1,2,3,4,5])
np.save('outfile',a)
b = np.load('outfile.npy')
print(b)

a = np.array([1,2,3,4,5])
np.savetxt('out.txt',a)
b = np.loadtxt('out.txt')
print(b)
```

课堂实验

- 使用循环和向量化两种不同的方法来计算 100 以内的质数之和。

```
# 首先先准备好判断质数的函数
def checkprime(x):
    if x<=1:
        return False
    prime=True
    for i in range(2 , 1+x//2):
        if x%i == 0:
            prime = False
            break
    return prime
```

```
# 用函数方法求和
def sumprimebyiter(n=100):
    primesum=0
    for i in range(1, n+1):
        if( True == checkprime(i)):
            primesum += i
    return primesum
```

```
#用向量化方法
import numpy as np
def sumprimebyarr(n=100):
    a = np.arange(1,n+1)
    # return sum(a[np.array(map(CheckPrime, a))]) # 此处是之前用Python 自带的 map 把函数应用到向量的每个元素
    check_prime_vec = np.vectorize(checkprime) # 此处代码用到了np.vectorize，可以把外置函数应用到向量的每个元素
    return np.sum(a[check_prime_vec(a)])
```

- 随机生成10个坐标，计算两两之间距离

```
import numpy as np
Z = np.random.random((10,2))
X,Y = np.atleast_2d(Z[:,0]), np.atleast_2d(Z[:,1])
D = np.sqrt( (X-X.T)**2 + (Y-Y.T)**2)
print (D)
```

内容回顾

1. NumPy简介及安装
2. NumPy基本数据类型及属性
3. NumPy数据操作
4. NumPy函数

Thank you !