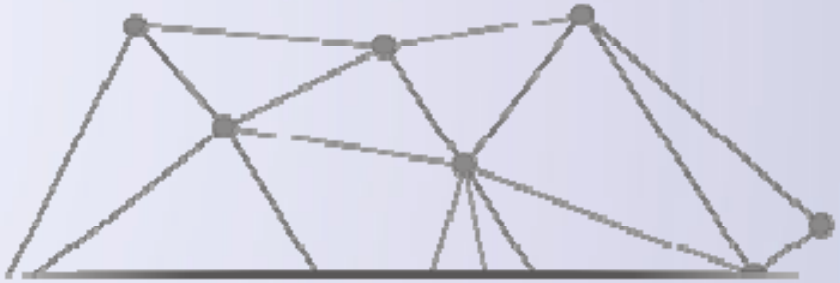
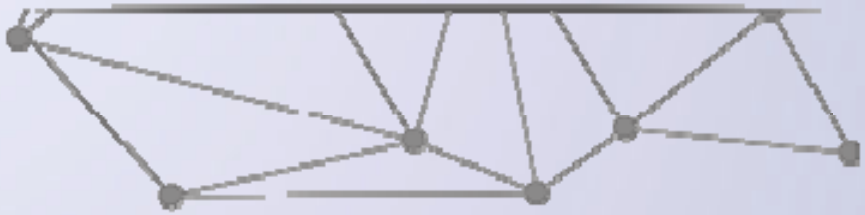


NumPy库介绍



礼 欣

北京理工大学

NumPy的安装

- NumPy系统是Python的一种开源的数值计算扩展
  - 可用来存储和处理大型矩阵
  - 使用前需要安装
    - 可以利用Python自带的pip工具自动安装
    - 或者选择访问下面的网站，下载与Python版本匹配的exe安装文件 http://sourceforge.net/projects/numpy/files/NumPy/
    - 安装完成后，打开Python3.4，运行命令import numpy，若不出现错误则说明安装成功。

NumPy的组成与功能

- Numpy （Numeric Python）可以被理解为一个用python实现的科学计算包，包括：
  - 强大的N维数组对象Array；
  - 成熟的函数库；
  - 实用的线性代数、傅里叶变换和随机数生成函数。
- 提供了许多高级的数值编程工具，如：矩阵数据类型、矢量处理，以及精密的运算库。

基础知识

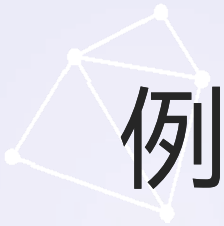
- NumPy的主要对象是同种元素的多维数组
  - 维度(dimensions)叫做轴(axes)
  - 轴的个数叫做秩(rank)。
  - 例如，在3D空间一个点的坐标 [1, 2, 3] 是一个秩为1的数组，因为它只有一个轴。轴长度为3.
  - 例如，在[[1., 0., 0.], [0., 1., 2.]]这个例子中，数组的秩为2(它有两个维 度).第一个维度长度为2,第二个维度长度为3.

基础知识

- NumPy的数组类被称作ndarray，通常被称作数组。
  - 注意numpy.array和标准Python库类array.array并不相同，后者只处理一维数组和提供少量功能。
  - ndarray对象属性主要见下表： 例如其中.shape表示数组的维度， .size表示数组元素的总个数

基础知识

属性	解释
Ndarray.shape	数组的维度，这是一个指示数组在每个维度上大小的整数元组。
ndarray.size	数组元素的总个数，等于shape属性中元组元素的乘积。
ndarray.dtype	一个用来描述数组中元素类型的对象，可以通过使用标准Python类型创造dtype。
ndarray.itemsize	数组中每个元素字节的大小。
ndarry.data	包含实际数组元素的缓冲区，通常我们通过索引引用数组元素，不使用这个属性。



例子

```
>>> from numpy import *
>>> a = arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int32'
>>> a.itemsize
4
```

```
>>> a.size
15
>>> type(a)
<class 'numpy.ndarray'>
>>> b = array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<class 'numpy.ndarray'>
>>>
```

创建数组（方法一）

- 创建数组的方法有多种，比如可以使用array函数利用常规的Python列表和元组创造数组。所创建的数组类型由原序列中的元素类型决定。示例如下：

```
>>> from numpy import *
>>> a = array([2, 3, 4])
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int32')
>>> b = array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
>>> #一个常见的错误包括用多个数值参数调用`array`而不是提供一个由数值组成的列表作为一个参数。
>>> a = array(1, 2, 3, 4) #WRONG
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    a = array(1, 2, 3, 4) #WRONG
ValueError: only 2 non-keyword arguments accepted
>>> a = array([1, 2, 3, 4]) #RIGHT
>>> c = array([[1, 2], [3, 4]], dtype = complex)
>>> c
array([[ 1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]])
```

创建数组（方法二）

- NumPy提供了一些使用占位符创建数组的函数。
  - 例如： 函数zeros创建一个全是0的数组，函数ones创建一个全1的数组，函数empty创建一个内容随机并且依赖与内存状态的数组。默认创建的数组类型(dtype)都是float64。示例如下：

```
>>> zeros((3, 4))
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
>>> ones((2, 3, 4), dtype = int16) # dtype can also be specified
array([[[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]],

       [[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]], dtype=int16)
>>> empty((2, 3))
array([[  7.17065762e-310,   4.62140475e-273,   5.00016999e+173],
       [  7.17065513e-310,   1.98488126e-263,   5.00258533e+173]])
```

创建数组（方法三）

- 此外NumPy提供一个arange的函数返回数组，示例如下：

```
>>> arange(10, 30, 5)
array([10, 15, 20, 25])
>>> arange(0, 2, 0.3) # it accepts float arguments
array([ 0. ,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8])
```

打印数组

- 打印数组时，NumPy以类似嵌套列表的形式显示
  - 示例如下：其中 一维数组被打印成行，二维数组成矩阵，三维数组成矩阵列表。

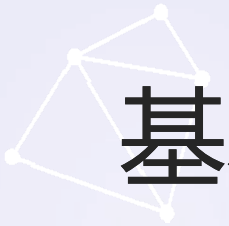
```
>>> a = arange(6) # 1d array
>>> print(a)
[0 1 2 3 4 5]
>>>
>>> b = arange(12).reshape(4, 3) # 2d array
>>> print(b)
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
>>>
>>> c = arange(24).reshape(2, 3, 4) # 3d array
>>> print(c)
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
>>>
```

基本运算

- 数组的算术运算是按元素进行。
  - NumPy中的乘法运算符*指示按元素计算
- 矩阵乘法可以使用dot函数或创建矩阵对象实现。

示例如下：

```
>>> a = array([20, 30, 40, 50])
>>> b = arange(4)
>>> b
array([0, 1, 2, 3])
>>> c = a - b
>>> c
array([20, 29, 38, 47])
>>> b ** 2
array([0, 1, 4, 9])
>>> 10 * sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a < 35
array([ True,  True, False, False], dtype=bool)
```



基本运算

- 非数组的运算可以利用ndarrary类方法实现。
- 通用函数(ufunc) -- NumPy提供常见的数学函数如sin, cos和exp。
  - 在NumPy里这些函数作用按数组的元素运算，产生一个数组作为输出。示例如下：

通用函数示例

```
>>> a = random.random((2, 3))
>>> a
array([[ 0.52732678,  0.92066148,  0.25701814],
       [ 0.66596685,  0.24443251,  0.39027655]])
>>> a.sum()
3.0056823003535524
>>> a.min()
0.24443250574359088
>>> a.max()
0.92066148305911222
>>> b = arange(12).reshape(3, 4)
>>> b
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> b.sum(axis = 0)
array([12, 15, 18, 21])
>>> #sum of each column
>>>
>>> b.min(axis = 1) # min of each row
array([0, 4, 8])
>>>
>>> b.cumsum(axis = 1) # cumulative sum along each row
array([[ 0,  1,  3,  6],
       [ 4,  9, 15, 22],
       [ 8, 17, 27, 38]], dtype=int32)
```

索引、切片与迭代

- 数组还可以被索引、切片和迭代，示例如下：

```
>>> a = arange(10) ** 3
>>> a
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729], dtype=int32)
>>> a[2]
8
>>> a[2:5]
array([ 8, 27, 64], dtype=int32)
>>> a[:6:2] = -1000 # equivalent to a[0:6:2] = -1000;from start to posi
tion6, exclusive, set every 2nd element to -1000
>>> a
array([-1000,     1, -1000,    27, -1000,   125,   216,   343,   512,
729], dtype=int32)
>>> a[::-1] # reversed a
array([  729,   512,   343,   216,   125, -1000,    27, -1000,     1, -
1000], dtype=int32)
```

```
>>> for i in a:
        print(i ** (1 / 3.),)


nan
1.0
nan
3.0
nan
5.0
6.0
7.0
8.0
9.0
```

python 语言程序设计

矩阵运算

- NumPy对于多维数组的运算，缺省情况下并不使用矩阵运算，对数组进行矩阵运算，可调用相应的函数。
- numpy库也提供了matrix类，使用matrix类创建的是矩阵对象，它们的加减乘除运算缺省采用矩阵方式计算，用法和matlab十分类似。示例如下：

```
>>> a = matrix([[1, 2, 3], [5, 5, 6], [7, 9, 9]])
>>> a * a ** -1
matrix([[  1.00000000e+00,   0.00000000e+00,   0.00000000e+00],
        [  4.44089210e-16,   1.00000000e+00,   4.44089210e-16],
        [  0.00000000e+00,  -4.44089210e-16,   1.00000000e+00]])
```

- 矩阵中更高级的一些运算可以在NumPy的线性代数子库linalg中找到。例如inv函数计算逆矩阵，solve函数可以求解多元一次方程组。

函数和方法的总览

- NumPy库提供作用丰富的函数和方法，下面是一个分类排列目录总揽，大家可以通过手册即用即学。

分类	函数
创建数组	arange, array, copy, empty, empty_like, eye, fromfile, fromfunction, identity, linspace, logspace, mgrid, ogrid, ones, ones_like, r , zeros, zeros_like
转化	astype, atleast 1d, atleast 2d, atleast 3d, mat
操作	array split, column stack, concatenate, diagonal, dsplit, dstack, hsplit, hstack, item, newaxis, ravel, repeat, reshape, resize, squeeze, swapaxes, take, transpose, vsplit, vstack