Numpy

<https://www.numpy.org.cn>

## 基本知识

NumPy的主要对象是同构多维数组。它是一个元素表（通常是数字），所有类型都相同，由非负整数元组索引。在NumPy维度中称为 轴

例如，3D空间中的点的坐标[1, 2, 1]具有一个轴。该轴有3个元素，所以我们说它的长度为3.在下图所示的例子中，数组有2个轴。第一轴的长度为2，第二轴的长度为3。

[[ 1., 0., 0.],

[ 0., 1., 2.]]

ndarray对象更重要的属性是：

* **ndarray.ndim** - 数组的轴（维度）的个数。在Python世界中，维度的数量被称为rank。
* **ndarray.shape** - 数组的维度。这是一个整数的元组，表示每个维度中数组的大小。对于有 *n* 行和 *m* 列的矩阵，shape 将是 (n,m)。因此，shape 元组的长度就是rank或维度的个数 ndim。
* **ndarray.size** - 数组元素的总数。这等于 shape 的元素的乘积。
* **ndarray.dtype** - 一个描述数组中元素类型的对象。可以使用标准的Python类型创建或指定dtype。另外NumPy提供它自己的类型。例如numpy.int32、numpy.int16和numpy.float64。
* **ndarray.itemsize** - 数组中每个元素的字节大小。例如，元素为 float64 类型的数组的 itemsize 为8（=64/8），而 complex32 类型的数组的 itemsize 为4（=32/8）。它等于 ndarray.dtype.itemsize 。
* **ndarray.data** - 该缓冲区包含数组的实际元素。通常，我们不需要使用此属性，因为我们将使用索引访问数组中的元素

>>> import numpy as np

>>> a = np.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

'int64'

>>> a.itemsize

8

>>> a.size

15

>>> type(a)

<type 'numpy.ndarray'>

>>> b = np.array([6, 7, 8])

>>> b

array([6, 7, 8])

>>> type(b)

<type 'numpy.ndarray'>

函数zeros创建一个由0组成的数组，函数 ones创建一个完整的数组，函数empty 创建一个数组，其初始内容是随机的，取决于内存的状态。默认情况下，创建的数组的dtype是 float64 类型的。

>>> np.zeros( (3,4) )

array([[ 0., 0., 0., 0.],

[ 0., 0., 0., 0.],

[ 0., 0., 0., 0.]])

>>> np.ones( (2,3,4), dtype=np.int16 ) #dtype can also bespecified

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)

>>> np.empty( (2,3) ) #uninitialized, output may vary

array([[ 3.73603959e-262, 6.02658058e-154, 6.55490914e-260],

[ 5.30498948e-313, 3.14673309e-307, 1.00000000e+000]])

为了创建数字组成的数组，NumPy提供了一个类似于range的函数，该函数返回数组而不是列表。

>>> np.arange( 10, 30, 5 )

array([10, 15, 20, 25])

>>> np.arange( 0, 2, 0.3 ) # it accepts float arguments

array([ 0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8])

当arange与浮点参数一起使用时，由于有限的浮点精度，通常不可能预测所获得的元素的数量。出于这个原因，通常最好使用linspace函数来接收我们想要的元素数量的函数，而不是步长（step）：

>>> from numpy import pi

>>> np.linspace( 0, 2, 9 ) # 9 numbers from 0 to 2

array([ 0. , 0.25, 0.5 , 0.75, 1. , 1.25, 1.5 , 1.75, 2. ])

>>> x = np.linspace( 0, 2\*pi, 100 )

# useful to evaluate function at lots of points

>>> f = np.sin(x)