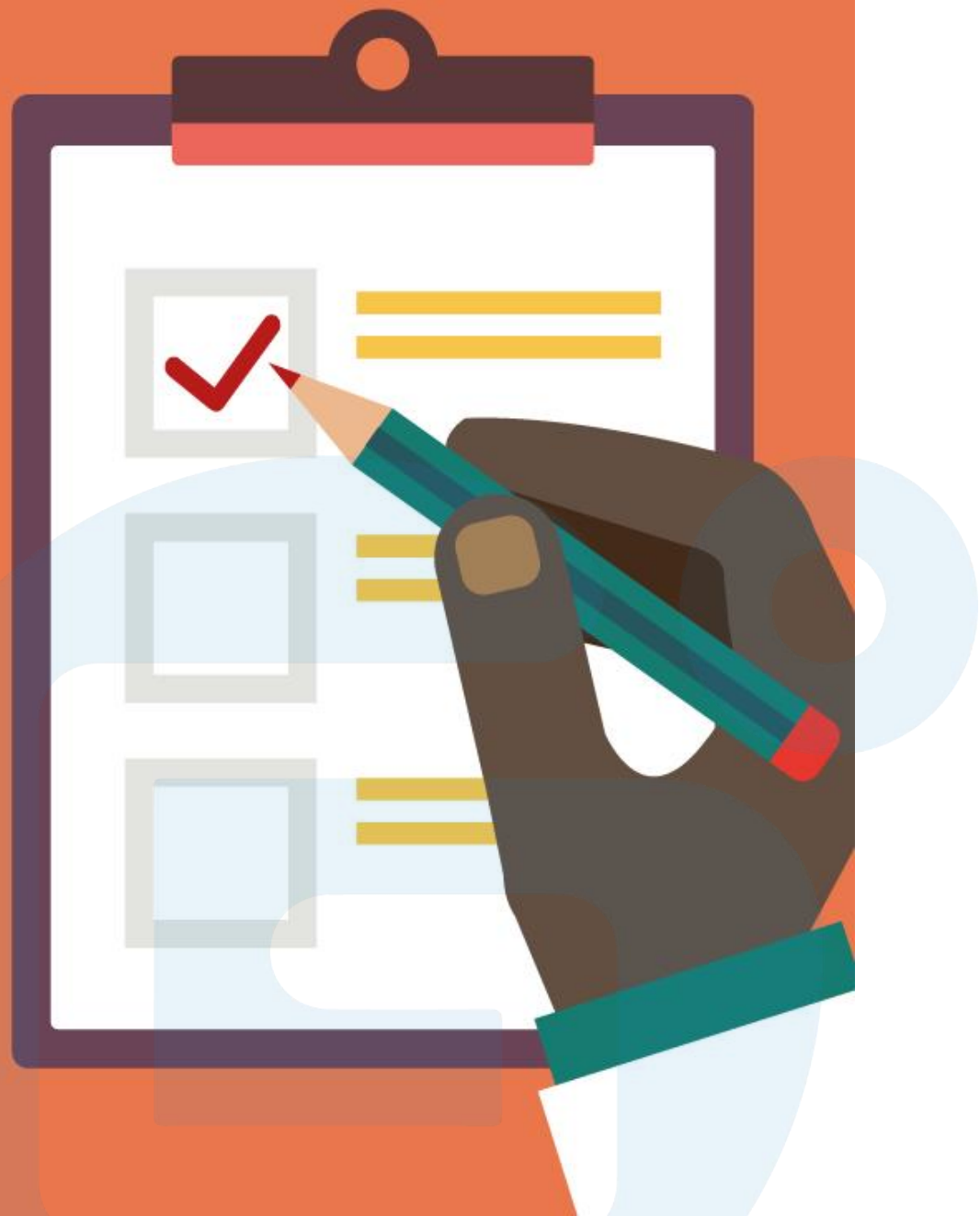


Talk is cheap, show me the code

第二课: Numpy 基础

Python进阶课程系列



OUTLINE

➤ Numpy简介

➤ Numpy数据类型与转换

➤ 索引与切片

➤ 数组文件的输入输出

➤ Numpy通用函数

➤ 数组计算



一 Numpy简介



Numpy简介

Python作为一种通用的编程语言，可以用于Web开发、爬虫脚本、GUI设计、**AI、机器学习、数据分析、数值计算**等。

Numpy (Numerical Python)是**高性能数值计算**和**数据分析技术**的基础库，它极大的简化了多维数组的操纵和处理，同时大幅提升了计算效率，作为一个基础库，它一般与Scipy、matplotlib、pandas等库一起使用。

- Numpy内置了可以对数组和矩阵进行快速运算的**标准函数**；
- Numpy是一个**免费开源**软件，而且由全世界的协作者共同开发维护；
- Numpy提供了很多**矢量运算**的接口，方便用户连接底层代码，比原生python的循环方法快很多；

原生的python里不含Numpy，通常采用两种方式安装Numpy：

1. 直接安装**Anaconda**，其内置了大量科学领域里的常用库，包括Numpy，pandas，matplotlib等；
2. 使用内置包管理工具**pip命令**安装，`pip install numpy`；

```
PS C:\Windows\system32> cd "C:\Program Files\Python38"
PS C:\Program Files\Python38>
PS C:\Program Files\Python38> python -m pip install numpy
Collecting numpy
  Using cached https://files.pythonhosted.org/packages/d0/1d/dcf7dec400df56c412f6e91824f21ab...
Installing collected packages: numpy
Successfully installed numpy-1.18.4
WARNING: You are using pip version 19.2.3, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\Program Files\Python38>
```



<https://www.anaconda.com/products/individual>

Numpy简介

判断是否安装成功

打开任意IDE，或者在命令行下进入交互式解释器，使用以下命令。如无报错则说明安装成功。

```
>>>import numpy as np
>>>print (np.version.version)
1.16.5
```

展现Numpy优势(编写效率&运行效率)的一个小例子

```
>>>import numpy as np
>>> my_arr = np.arange(1000000);
>>> my_list = list(range(1000000));
>>> %time for _ in range(10): my_arr2 = my_arr * 2 # %time为IPython解释器里的魔术命令
Wall time: 16 ms

>>>%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
Wall time: 736 ms
```

Numpy优缺点总结		
优点	编写便捷	对于同样的数据处理任务，利用Numpy比直接使用python原生代码开发更方便
	性能卓越	Numpy的数组存储效率和输入输出性能均优于原生Python，且数组中的元素越多，这种优势越明显
	代码高效	Numpy大部分底层算法代码是C语言，更贴近机器语言，因此运行效率比原生python高效的多
缺点	处理大文件能力有限	Numpy利用内存映射文件以达到最优的数据读写性能，因此内存大小限制了其对大文件（数百G乃至TB级）的处理
	数组通用性不足	Numpy数组的通用性不如原生python的list集合，在数据科学以外的领域，无太多优势



二 Numpy数据类型与转换



Numpy数据类型与转换

Numpy里最常用的是**N维数组对象ndarray**，它是用于存放**同类型元素**的多维数组，ndarray中的每个元素在**内存中占有相同大小的区域**。创建ndarray对象时，可以通过dtype为其指定数据类型，如果没有指定，则按元素内容自动确定。创建方法如下：

方法名称	说明
<code>np.array(array_like)</code>	将传递的序列类型数据（如list、tuple、ndarray等）转换为ndarray, 返回一个新的ndarray对象
<code>np.asarray(array_like)</code>	将传递的序列类型数据（list、tuple等）转换为ndarray，返回一个新的ndarray对象，但当传递ndarray时，则不会生成新的对象
<code>np.arange(start,stop,step)</code>	根据传递的参数，返回等间隔的数据组成的ndarray，与range()方法类似，但这里步长可以是小数
<code>np.empty(shape)</code>	用于创建指定形状的空数组，数组元素为随机值
<code>np.zeros(shape)</code>	用于创建指定形状的数组，数组元素都为0
<code>np.zeros_like(ndarray对象)</code>	创建一个形状与ndarray对象相同，元素都为0的数组

Numpy数据类型与转换

<code>np.ones(shape)</code>	用于创建指定形状的数组， 数组元素都为1
<code>np.ones_like(ndarray对象)</code>	创建一个形状与ndarray对象相同 ，元素都为1的数组
<code>np.full(形状, 值)</code>	创建指定形状的数组，数组中所有元素相同， 为指定值
<code>np.linspace(start, stop, num)</code>	创建 等差数组 ，指定开始、结束以及元素个数，程序自动计算， num默认为50
<code>np.logspace(start, stop, num)</code>	创建 等比数组 ，指定开始、结束以及元素个数，程序自动计算， num默认为50
<code>np.eye(n, m)</code>	创建 $n*m$ 的单位矩阵 ，对角线为1，其余为0
<code>np.identity(n)</code>	创建 $n*n$ 的单位方阵 ，对角线为1，其余为0

ndarray对象不支持直接输入数据，必须以list或tuple变量为参数

```
In [24]: import numpy as np
...: lst1=[1.0,2.2,3.3]
...: a=np.array(lst1)
...: print (a)
...: print (type(a))
...: print (a.dtype)
[1.  2.2 3.3]
<class 'numpy.ndarray'>
float64
```

```
In [25]: import numpy as np
...: a=np.array(1.0,2.0,3.3)
Traceback (most recent call last):

  File "<ipython-input-25-d6e1bedec96b>", line 2, in <module>
    a=np.array(1.0,2.0,3.3)
ValueError: only 2 non-keyword arguments accepted

In [28]: import numpy as np
...: a=np.array([[1.0,2.0,3.3],[4.0,5.0,6.0]]);\
...: a
Out[28]:
array([[1. , 2. , 3.3],
       [4. , 5. , 6. ]])
```

Numpy数据类型与转换

ndarray对象属性

ndarray的属性	说明
shape	返回一个元组，表示数组各个维度的长度，元组的长度为数组的维度（与ndim相同），元组的每个元素的值代表了数组每个维度的长度
ndim	ndarray对象的维度
size	ndarray中元素的个数，相当于各个维度长度的乘积
dtype	ndarray中存储的元素的数据类型
itemsize	ndarray中每个元素的字节数
nbytes	ndarray中所有元素所占字节数

```
In [28]: import numpy as np
...: a=np.array([[1.0,2.0,3.3],[4.0,5.0,6.0]]);\
...: a
Out[28]:
array([[1. , 2. , 3.3],
       [4. , 5. , 6. ]])

In [29]: a.ndim
Out[29]: 2

In [30]: a.shape
Out[30]: (2, 3)
```

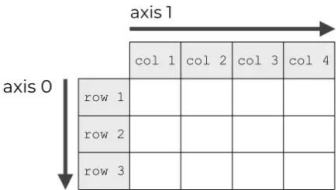
ndarray对象方法

方法	说明
reshape()	用于将原来数组的数据重新按照维度划分，划分后只是改变数据显示方式，并未重新创建数组，如果指定的维度和数组的元素数目不匹配，将抛出异常
flatten()	将一个多维数组转换成一维数组形式，可以指定转换的顺序，转化后生成了一个新的数组
astype()	显式指定数组中元素的类型，将会创建一个新的数组
sum()	对所有元素求和，指定轴时，按轴求和
cumsum()	累积求和，指定轴时，按轴累积求和
max()	对所有元素求最大值，指定轴时，按轴求最大值
min()	对所有元素求最小值，指定轴时，按轴求最小值
mean()	对所有元素求平均值，指定轴时，按轴求平均值

```
In [39]: import numpy as np
...: a=np.array([[1.0,2.0,3.3],[4.0,5.0,6.0]]);\
...: a.max()
Out[39]: 6.0

In [40]: a.max(0)
Out[40]: array([4., 5., 6.])

In [41]: a.max(1)
Out[41]: array([3.3, 6. ])
```



Numpy数据类型与转换

ndarray数据类型可以通过参数dtype 设定，而且可以使用`astype`转换类型，在处理文件时候这个会很实用，注意`astype` 调用会返回一个新的数组，也就是原始数据的一份拷贝，而原数组并未改变。

```
In [14]: numeric_strings2 = np.array([1.23,2.34,3.45],dtype=np.string_)
...: numeric_strings2
Out[14]: array([b'1.23', b'2.34', b'3.45'], dtype='<S4')

In [15]: numeric_strings2.astype(float)
Out[15]: array([1.23, 2.34, 3.45])

In [16]: numeric_strings2
Out[16]: array([b'1.23', b'2.34', b'3.45'], dtype='<S4')

In [17]: numeric_2=numeric_strings2.astype(float)

In [18]: numeric_2
Out[18]: array([1.23, 2.34, 3.45])
```

bool	?, b1	float16	f2, e
int8	b, i1	float32	f4, f
uint8	B, u1	float64	f8, d
int16	h, i2	complex64	F4, F
uint16	H, u2	complex128	F8, D
int32	i, i4	str	a, S (S后面添加数字表示字符串长度，不写则为最大长度)
uint32	I, u4	unicode	U
int64	q, i8	object	O
uint64	Q, u8	void	V



三 索引与切片



索引与切片

ndarray对象的索引和切片操作类似于原生Python里序列容器的索引和切片。索引支持正向索引（从左至右，下标从0开始增大）和反向索引（从右至左，下标从-1开始减小）。切片操作可以通过slice函数，设置start、stop、step参数进行。也可以采用冒号分割切片参数start:stop:step进行。

注意两者之间的区别：序列容器切片之后，会生成一个新的序列，等于将有关元素复制出来组成新序列。但ndarray切片结果不会单独生成新ndarray，切片结果类似cpp里的引用，仅仅是取出地址，对该切片的修改会影响原始数据。

```
In [1]: a=list(range(10))
In [2]: a
Out[2]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [3]: b=a[2:5]

In [4]: b
Out[4]: [2, 3, 4]

In [5]: b[1]=100

In [6]: b
Out[6]: [2, 100, 4]

In [7]: a
Out[7]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

VS

```
In [7]: import numpy as np
...: a=np.array(range(10))
...: a
Out[7]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [8]: b=a[2:5];\
...: b
Out[8]: array([2, 3, 4])

In [9]: b[1]=100;\
...: b
Out[9]: array([ 2, 100,  4])

In [10]: a
Out[10]: array([ 0,  1,  2, 100,  4,  5,  6,  7,  8,  9])
```


索引与切片

思考为什么这么设计？

- Numpy 设计是为了处理大量数据，如果切片采用数据复制话会产生极大的性能和内存消耗问题，而仅利用地址传递则会提高性能。
- 如果希望在numpy里生成切片副本，在不改变原数组的情况下进行修改，可以使用`.copy()`方法。

```
In [7]: import numpy as np
...: a=np.array(range(10))
...: a
Out[7]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [8]: b=a[2:5];\
...: b
Out[8]: array([2, 3, 4])

In [9]: b[1]=100;\
...: b
Out[9]: array([ 2, 100,  4])

In [10]: a
Out[10]: array([ 0,  1,  2, 100,  4,  5,  6,  7,  8,  9])
```

直接对ndarray对象切片

```
In [13]: import numpy as np
...: a=np.array(range(10))
...: a
Out[13]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [14]: b_copy=a[2:5].copy();\
...: b_copy
Out[14]: array([2, 3, 4])

In [15]: b_copy[1]=100;\
...: b_copy
Out[15]: array([ 2, 100,  4])

In [16]: a
Out[16]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

用`.copy()`方法对ndarray对象切片

多维数组索引、切片

```
In [17]: import numpy as np
...: a=np.arange(60).reshape(3,4,5);\
...: a
Out[17]:
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, 24],
        [25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34],
        [35, 36, 37, 38, 39]],

       [[40, 41, 42, 43, 44],
        [45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54],
        [55, 56, 57, 58, 59]]])
```

- 将60个数据，存储在3*4*5的三维数组中。
- 可以直接利用各维度的序号选定某个元素，例如
- 维度序号可以为缺省值，用冒号替换具体数值即可，例如
- 多个冒号可以用省略号替代，

```
In [21]: a[1,...]
Out[21]:
array([[20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34],
       [35, 36, 37, 38, 39]])
```

```
In [18]: a[2,2,2]
Out[18]: 52
```

```
In [19]: a[0,:,:]
Out[19]:
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

- 切片可以是不连续的数据，利用step参数或者直接指定，

```
In [22]: a[0,0,::2]
Out[22]: array([0, 2, 4])
```

```
In [25]: a[1,1,[0,1,3]]
Out[25]: array([25, 26, 28])
```

```
In [26]: a[1,1,[0,-1]]
Out[26]: array([25, 29])
```

高级索引

1. 整数数组索引：通过在x[]放入整数数组来表达要索引元素的下标

```
In [17]: import numpy as np
...: a=np.arange(60).reshape(3,4,5);\
...: a
Out[17]:
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, 24],
        [25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34],
        [35, 36, 37, 38, 39]],

       [[40, 41, 42, 43, 44],
        [45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54],
        [55, 56, 57, 58, 59]]])
```

```
In [31]: a[[0,0,0,1],[1,2,3,1],[1,2,3,2]] #获取的是[0,1,1],[0,2,2],[0,3,3],[1,1,2]4个位置的数
Out[31]: array([ 6, 12, 18, 27])
```

2. 布尔索引：通过布尔运算（如：比较运算符）来获取符合指定条件的元素的数组

以获取所有能整除5为数为目标，可写成`a[a%3==0]`，多个条件之间用布尔运算符`&`和`|`连接

```
In [33]: a[a%3==0]
Out[33]:
array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48,
        51, 54, 57])

In [34]: a[(a%3==0)|(a%7==0)]
Out[34]:
array([ 0,  3,  6,  7,  9, 12, 14, 15, 18, 21, 24, 27, 28, 30, 33, 35, 36,
        39, 42, 45, 48, 49, 51, 54, 56, 57])
```

```
In [35]: a[a>30]=0
```

```
In [36]: a
Out[36]:
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, 24],
        [25, 26, 27, 28, 29],
        [30,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0]],

       [[ 0,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0]]])
```


高级索引

3.花式索引

主要为`np.ix_`方法。对输入的数组产生笛卡尔积（线性代数里的叉乘）的映射关系，然后按照生成的整数数组索引去生成数组。

```
In [17]: import numpy as np
...: a=np.arange(60).reshape(3,4,5);\
...: a
Out[17]:
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, 24],
        [25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34],
        [35, 36, 37, 38, 39]],

       [[40, 41, 42, 43, 44],
        [45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54],
        [55, 56, 57, 58, 59]]])
```

```
In [2]: a[[1,2],[0,2],[1,1]]
Out[2]: array([21, 51])

In [3]: a[np.ix_([1,2],[0,2],[1,1])]
Out[3]:
array([[ [21, 21],
        [31, 31]],

       [[41, 41],
        [51, 51]]])
```

从最后一个维度开始遍历，实际的索引下标是

101,101
121,121
201,201
221,221

```
In [13]: a[np.ix_([1,2,0],[0,1,2])]
Out[13]:
array([[ [20, 21, 22, 23, 24],
        [25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34]],

       [[40, 41, 42, 43, 44],
        [45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54]],

       [[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14]]])
```

从最后一个维度开始遍历，实际的索引下标是

10:,11:,12:
20:,21:,22:
00:,01:,02:



四 数组文件的输入输出





数组文件的输入输出

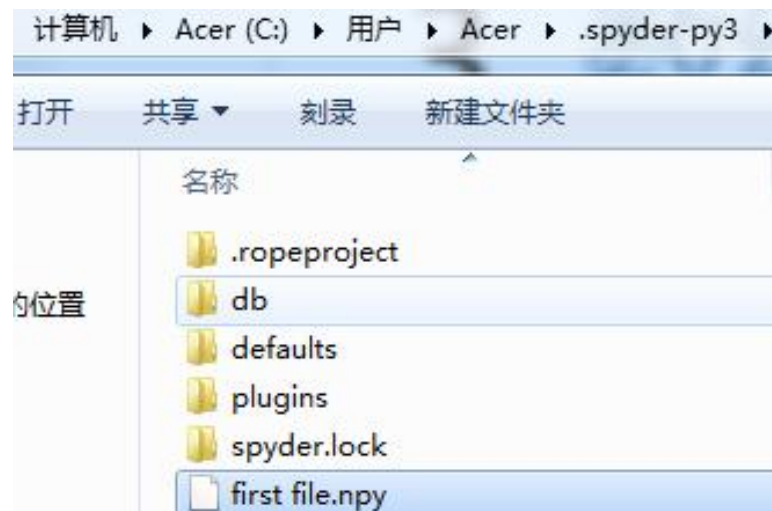
在numpy中也可以使用函数对数据文件进行读取操作，类似于原生python。不同之处在于numpy的主要操作对象是数值和数组。

np.save、**np.load** 函数是读写磁盘的两个主要函数，默认情况下，数组以未压缩的原始二进制格式保存在扩展名为.npy的自带格式文件中。

```
In [3]: arr=np.arange(10)

In [4]: np.save("first file",arr)

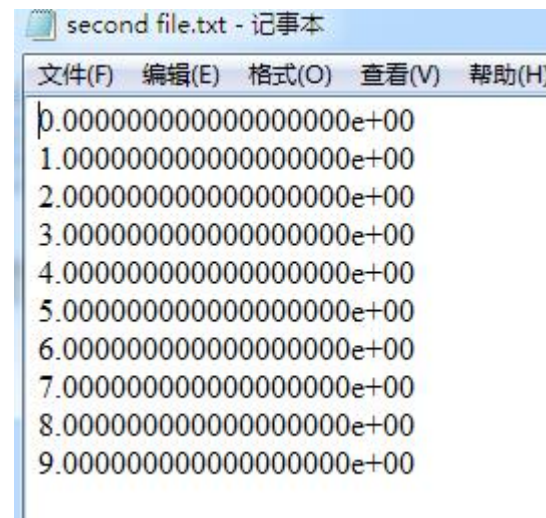
In [5]: np.load("first file.npy")
Out[5]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```



数组文件的输入输出

用 **np.savetxt** 函数将 numpy 数组保存为 .txt 格式，具体写法：`np.savetxt(“文件名”，数组)`。存取的时候要注意数据格式问题，如果不主动设置存取格式，有可能在保存时将int型变成了float型。

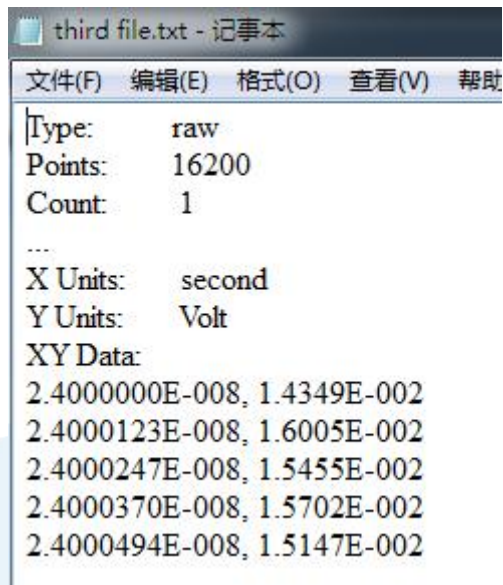
```
In [7]: arr=np.arange(10);\n...: np.savetxt("second file.txt",arr)\n\nIn [8]: np.loadtxt("second file.txt")\nOut[8]: array([0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])\n\nIn [10]: np.loadtxt("second file.txt",dtype=int)\nOut[10]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```



```
second file.txt - 记事本\n文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)\n0.0000000000000000e+00\n1.0000000000000000e+00\n2.0000000000000000e+00\n3.0000000000000000e+00\n4.0000000000000000e+00\n5.0000000000000000e+00\n6.0000000000000000e+00\n7.0000000000000000e+00\n8.0000000000000000e+00\n9.0000000000000000e+00
```

用 **np.loadtxt** 函数可以读入 .txt 格式，此外还有一个更加复杂，功能也更强大的函数**np.genfromtxt** 可以选用。 `genfromtxt` 运行两个主循环。第一个循环以字符串序列转换文件的每一行。第二个循环将每个字符串转换为适当的数据类型。这种机制比单个循环慢，但具有更大的灵活性和更强大的功能。可以读取csv和txt两种格式的文件，也可以处理字符串。

数组文件的输入输出



```
third file.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助
Type:      raw
Points:    16200
Count:     1
...
X Units:   second
Y Units:   Volt
XY Data:
2.4000000E-008, 1.4349E-002
2.4000123E-008, 1.6005E-002
2.4000247E-008, 1.5455E-002
2.4000370E-008, 1.5702E-002
2.4000494E-008, 1.5147E-002
```

np.genfromtxt应用举例

```
In [3]: import numpy as np
...: data = np.genfromtxt("third file.txt", delimiter=',', skip_header=7, skip_footer=1)

In [4]: data
Out[4]:
array([[2.4000000e-08, 1.4349000e-02],
       [2.4000123e-08, 1.6005000e-02],
       [2.4000247e-08, 1.5455000e-02],
       [2.4000370e-08, 1.5702000e-02],
       [2.4000494e-08, 1.5147000e-02]])
```

numpy.genfromtxt

numpy.genfromtxt(*fname*, *dtype*=<type 'float'>, *comments*='#', *delimiter*=None, *skiprows*=0, *skip_header*=0, *skip_footer*=0, *converters*=None, *missing*='', *missing_values*=None, *filling_values*=None, *usecols*=None, *names*=None, *excludelist*=None, *deletechars*=None, *replace_space*='_', *autostrip*=False, *case_sensitive*=True, *defaultfmt*='%i', *unpack*=None, *usemask*=False, *loose*=True, *invalid_raise*=True)

[\[source\]](#)

Load data from a text file, with missing values handled as specified.

Each line past the first *skip_header* lines is split at the *delimiter* character, and characters following the *comments* character are discarded.

Parameters: *fname* : file or str

File, filename, or generator to read. If the filename extension is *gz* or *bz2*, the file is first decompressed. Note that generators must return byte strings in Python 3k.



五 Numpy通用函数



数学函数

在numpy中，提供了众多的内置数学函数，可以对ndarray对象中的元素进行基本的数学运算，无需用户自己写循环。

方法	说明
<code>np.abs()</code> 、 <code>np.fabs()</code>	计算整数、浮点数的绝对值
<code>np.sqrt()</code>	计算各元素的平方根
<code>np.square()</code>	计算各元素的平方
<code>np.exp()</code>	计算各元素的指数 e^x
<code>np.log()</code> 、 <code>np.log10()</code> 、 <code>np.log2()</code>	计算各元素的自然对数、底数为10的对数、底数为2的对数
<code>np.sign()</code>	计算各元素的符号，1（正数）、0（零）、-1（负数）
<code>np.ceil()</code> 、 <code>np.floor()</code> 、 <code>np rint()</code>	对各元素分别向上取整、向下取整、四舍五入
<code>np.modf()</code>	将各元素的小数部分和整数部分以两个独立的数组返回
<code>np.cos()</code> 、 <code>np.sin()</code> 、 <code>np.tan()</code>	对各元素求对应的三角函数
<code>np.add()</code> 、 <code>np.subtract()</code> 、 <code>np.multiply()</code>	对两个数组的各元素执行加法、减法、乘法等操作

随机数函数

numpy.random函数对python内置的random方法提供了有效的补充，添加了一些可以**高效生成多种概率分布**的样本值函数。

方法	说明
<code>np.random.rand(4, 5)</code>	随机生成4行5列数组，每个元素都是[0,1)之间的小数
<code>np.random.randint(a,b,(3,4))</code>	随机生成3行4列数组，每个元素都是[a,b)之间的整数
<code>np.random.randn(4,5)</code>	随机生成4行5列数组，元素的值符合标准正态分布
<code>np.random.shuffle(数组)</code>	随机打乱数组的顺序
<code>np.random.uniform(a, b, 15)</code>	均匀分布，随机生成15个在[a, b)之间的小数
<code>np.random.choice(一维数组, size= (3,4))</code>	随机从一维数组中抽取指定数量的元素
<code>np.random.binomial()</code>	随机生成符合二项分布的样本
<code>np.random.normal()</code>	随机生成符合正态分布的样本
<code>np.random.beta()</code>	随机生成符合Beta分布的样本
<code>np.random.gamma()</code>	随机生成符合Gamma分布的样本

统计函数

有一些ndarray的对象方法也是numpy函数。例如all, any, argmax, argmin, copy, cumprod, cumsum, diagonal, imag, max, mean, min, nonzero...

方法	说明
<code>np.sum()</code>	对数组中元素进行求和，空数组的和为0
<code>np.mean()</code> 、 <code>np.median()</code>	获取一组数据的平均数、中位数，空数组的平均值为NaN
<code>np.std()</code> 、 <code>np.var()</code>	获取一组数据的标准差和方差
<code>np.min()</code> 、 <code>np.max()</code>	获取最大值和最小值，可以指定轴
<code>np.argmin()</code> 、 <code>np.argmax()</code>	获取最大、最小元素的索引
<code>np.cumsum()</code>	对数组中元素累积求和，可指定轴
<code>np.cumprod()</code>	对数组中元素累积求积，可指定轴
<code>np.ptp()</code>	计算一组数中最大值与最小值的差，可指定轴
<code>np.unique()</code>	删除数组中的重复数据，并对数据进行排序
<code>np.nonzero()</code>	返回数组中非零元素的索引

numpy的其它常用函数

方法	说明
<code>np.tile()</code>	将数组的数据按照行列复制扩展
<code>np.repeat()</code>	将数组中的每个元素重复若干次
<code>np.savetxt()</code>	将数据保存到txt文件中去
<code>np.loadtxt()</code>	从文件中加载数据
<code>np.genfromtxt()</code>	根据文件内容中生成数据，可以指定缺失值的处理等
<code>np.any()</code>	如果数组中存在一个为True的元素（或者能转为True的元素），则返回True
<code>np.all()</code>	如果数组中所有元素都为True（或者能转为True的元素），则返回True
<code>np.where(条件, x, y)</code>	如果条件为True，对应值为x，否则对应值为y
<code>np.sort()</code>	对数组进行排序，返回一个新的排好序的数组，原数组不变
<code>np.argsort()</code>	返回的是数组值从小到大排序后元素对应的索引值
<code>np.mat()</code>	将一个数组转换成矩阵
<code>np.transpose()</code>	数组行列转置，只是改变元素访问顺序，并未生成新的数组



六 数组计算



数组计算

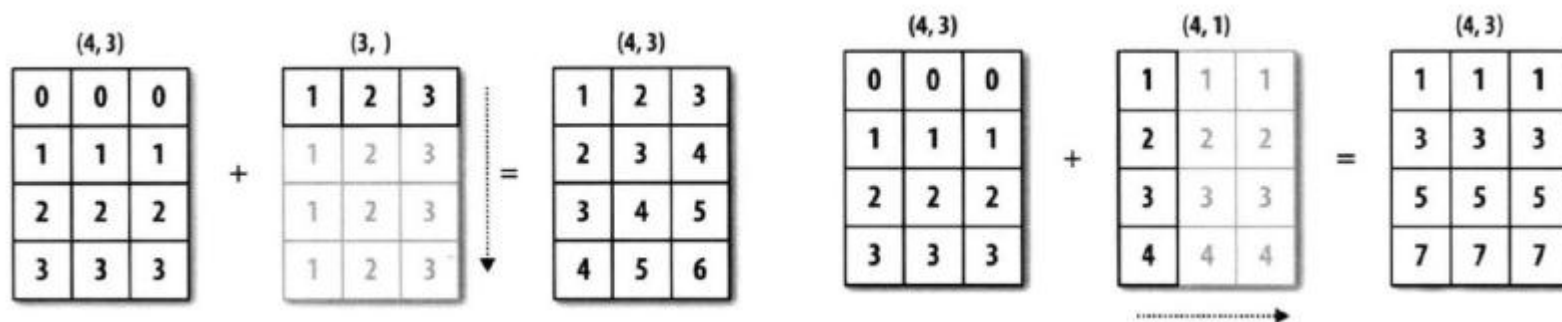
在任何一种涉及到数值计算的编程语言中，数组计算（或者矩阵运算、矢量计算、多重线性代数，随便你怎么称呼），都是必不可少甚至是最核心的功能。

numpy数组计算的特殊机制——**广播**。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，这种机制叫做广播（broadcasting）。

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。

- 让**所有输入数组都向其中形状最长的数组看齐**，形状中不足的部分都通过在前面加1补齐。
- 输出数组的形状是输入数组形状的各个维度上的**最大值**。
- 如果输入数组的某个维度和输出数组的对应维度的长度相同或者其长度为1时，这个数组能够用来计算，否则出错。
- 当输入数组的某个维度的长度为1时，沿着此维度运算时都用此维度上的第一组值。
- 若条件不满足，抛出 "ValueError: frames are not aligned" 异常。

数组计算



```
In [6]: a = np.array([[ 0, 0, 0],
...:                  [10,10,10],
...:                  [20,20,20],
...:                  [30,30,30]])
...: b = np.array([1,2,3],
...:               )
...: print(a + b)
[[ 1  2  3]
 [11 12 13]
 [21 22 23]
 [31 32 33]]
```

=

```
In [8]: a = np.array([[ 0, 0, 0],
...:                  [10,10,10],
...:                  [20,20,20],
...:                  [30,30,30]])
...: b = np.array([[1,2,3],
...:               [1,2,3],
...:               [1,2,3],
...:               [1,2,3]])
...: print(a + b)
[[ 1  2  3]
 [11 12 13]
 [21 22 23]
 [31 32 33]]
```

```
In [9]: a = np.array([[ 0, 0, 0],
...:                  [10,10,10],
...:                  [20,20,20],
...:                  [30,30,30]])
...: b = np.array([1,2,3],
...:               [4,5,6],
...:               [7,8,9],
...:               [10,11,12]])
...: print(a + b)
[[ 1  2  3]
 [14 15 16]
 [27 28 29]
 [40 41 42]]
```

```
In [13]: a = np.array([ 0, 1, 2])
...: b = np.array([[5],
...:               [6],
...:               [7]])
...: print(a + b)
[[5 6 7]
 [6 7 8]
 [7 8 9]]
```

```
In [10]: a = np.array([[ 0, 0, 0],
...:                  [10,10,10],
...:                  [20,20,20],
...:                  [30,30,30]])
...: b = np.array([1,2,3],
...:               [4,5,6]])
...: print(a + b)
```

Traceback (most recent call last):

```
File "<ipython-input-10-d91022db1280>", line 7, in <module>
    print(a + b)
```

ValueError: operands could not be broadcast together with shapes (4,3) (2,3)

维度既不相等，也不为1，
则报错

数组间的集合运算

```
In [19]: a=np.array([5,3,4,5,6,8,9,2,1]);\  
...: b=np.array([1,5,7,9,4,4,2]);\  
...: print(np.intersect1d(a,b)) #a和b的交集  
[1 2 4 5 9]  
  
In [20]: print(np.setdiff1d(a,b)) #差集, a中存在, b不存在的元素  
[3 6 8]  
  
In [21]: print(np.setdiff1d(b,a))  
[7]  
  
In [22]: print(np.union1d(a,b)) #并集  
[1 2 3 4 5 6 7 8 9]  
  
In [23]: print(np.setxor1d(a,b)) #异或集  
[3 6 7 8]  
  
In [24]: #XOR集合 (合并两组整数, 去掉在两组整数中都出现的整数后形成的集合)  
  
In [25]: print(np.in1d(a,b)) #判断a中元素是否在b  
[ True False  True  True False False  True  True  True]  
  
In [26]: print(np.in1d(b,a))  
[ True  True False  True  True  True  True]
```

数组间的连接和分隔

```
In [9]: import numpy as np
...: a=np.array([[2,3,4],[5,6,7]])
...: b=np.array([[9,8,7],[6,3,3],[2,2,1]])
...: print(np.concatenate((a,b),axis=0))
...: print(np.concatenate((a,b))) #np.concatenate((a,b))表示沿指定轴连接两个或多个数组，指定轴上的元素个数要相同，默认axis=0
```

```
[[2 3 4]
 [5 6 7]
 [9 8 7]
 [6 3 3]
 [2 2 1]]
[[2 3 4]
 [5 6 7]
 [9 8 7]
 [6 3 3]
 [2 2 1]]
```

```
In [13]: print(np.vstack((a,b)))
[[2 3 4]
 [5 6 7]
 [9 8 7]
 [6 3 3]
 [2 2 1]]
```

```
In [14]: print(np.hstack((a,b)))
Traceback (most recent call last):
```

```
File "<ipython-input-14-cb6f7dac968d>", line 1, in <module>
    print(np.hstack((a,b)))
```

```
File "C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\shape_base.py", line 340, in hstack
    return _nx.concatenate(arrs, 1)
```

```
ValueError: all the input array dimensions except for the concatenation axis must match exactly
```

数组间的连接和分隔

- **np.split()**:沿特定轴将数组分隔为子数组，如果为整数则用该数平均切分，如果是数组，则沿轴对应位置切分（左开右闭）；
- **np.hsplit()**:用于水平分隔数组，通过指定要返回的相同形状的数组数量来拆分原数组
- **np.vsplit()**:用于垂直切分

```
In [19]: a=np.arange(48).reshape(6,8)

In [20]: a
Out[20]:
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],
       [24, 25, 26, 27, 28, 29, 30, 31],
       [32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47]])

In [21]: print(np.split(a,3))
[array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14, 15]]), array([[16, 17, 18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29, 30, 31]]), array([[32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47]])]

In [22]: print(np.split(a,[1,3,5]))
[array([[0, 1, 2, 3, 4, 5, 6, 7]]), array([[ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20, 21, 22, 23]]), array([[24, 25, 26, 27, 28, 29, 30, 31],
       [32, 33, 34, 35, 36, 37, 38, 39]]), array([[40, 41, 42, 43, 44, 45, 46, 47]])]

In [23]: print(np.hsplit(a,[1,3,5]))
[array([[ 0],
       [ 8],
       [16],
       [24],
       [32],
       [40]]), array([[ 1,  2],
       [ 9, 10],
       [17, 18],
       [25, 26],
       [33, 34],
       [41, 42]]), array([[ 3,  4],
       [11, 12],
       [19, 20],
       [27, 28],
       [35, 36],
       [43, 44]]), array([[ 5,  6,  7],
       [13, 14, 15],
       [21, 22, 23],
       [29, 30, 31],
       [37, 38, 39],
       [45, 46, 47]])]

In [25]: print(np.vsplit(a,[1,3,5]))
[array([[0, 1, 2, 3, 4, 5, 6, 7]], array([[ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20, 21, 22, 23]]), array([[24, 25, 26, 27, 28, 29, 30, 31],
       [32, 33, 34, 35, 36, 37, 38, 39]]), array([[40, 41, 42, 43, 44, 45, 46, 47]])]
```


矩阵转置

```
In [27]: a = np.array([[1,0],[2,3]])
...: print(a)
[[1 0]
 [2 3]]

In [28]: print (a.transpose())
[[1 2]
 [0 3]]
```

```
In [34]: from numpy.linalg import inv

In [35]: a = np.array([[1,5],[2,3]])

In [36]: b=inv(a)

In [37]: print(b)
[[-0.42857143  0.71428571]
 [ 0.28571429 -0.14285714]]

In [38]: np.dot(a,b)
Out[38]:
array([[1.00000000e+00, 1.11022302e-16],
       [0.00000000e+00, 1.00000000e+00]])
```

numpy.linalg中有一组标准的矩阵分解运算以及诸如求逆和行列式之类的东西。它们跟MATLAB和R等语言所使用的是相同的行业标准线性代数库，如BLAS、LAPACK、Intel MKL等

函数	说明
diag	以一维数组的形式返回方阵的对角线（或非对角线）元素，或将一维数组转换为方阵（非对角线元素为0）
dot	矩阵乘法
trace	计算对角线元素的和
det	计算矩阵行列式
eig	计算方阵的本征值和本征向量
inv	计算方阵的逆
pinv	计算矩阵的Moore-Penrose伪逆
qr	计算QR分解
svd	计算奇异值分解（SVD）
solve	解线性方程组Ax = b，其中A为一个方阵
lstsq	计算Ax = b的最小二乘解

以左边4*5的二维数组（矩阵）为例

```
In [39]: a = np.arange(20).reshape(4,5)
```

```
In [40]: a
```

```
Out[40]:
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

- $a[2,3]$ 、 $a[2]$ 、 $a[2][3]$ 、 $a[2][:3]$ 、 $a[:,3]$ 、 $a[:,3]$ 各自表示什么含义？试着运行一下；
- 如果想获取第二行和第四行的数据，如何表示？
- 如果想获取第一列和第三列的数据，如何表示？
- 如果想获得所有大于10且能被3整除的数据，如何表示？
- 如何通过一个表达式获取第三行第四列，第二行第二列的数据？
- 垂直平均分割该数组；
- 水平分割该数组，分别为：第一列，第二至第四列，第五列。



感谢参与 下堂课见

