Python语言与数据科学（二）

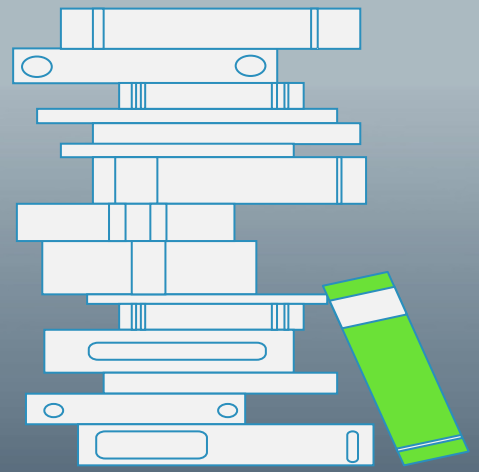
数值计算模块NumPy

东南大学 生物科学与医学工程学院 夏小俊
13913915329

东南大學 SOUTHEAST UNIVERSITY

01

Numpy简介



关于NumPy

- NumPy(Numerical Python)是一个开源的Python科学计算库
- NumPy是Python进行科学计算的基础软件包，是数据分析、机器学习、科学计算领域的底层核心
- NumPy极大简化了对数组和矩阵的操作，并包含了很多高效实用的数学函数，涉及线性代数、随机数等
- NumPy的底层算法代码使用C语言编写，运行效率很高

NumPy的历史与辉煌

Review

Array programming with NumPy

https://doi.org/10.1038/s41586-020-2649-2

Received: 21 February 2020

Accepted: 17 June 2020

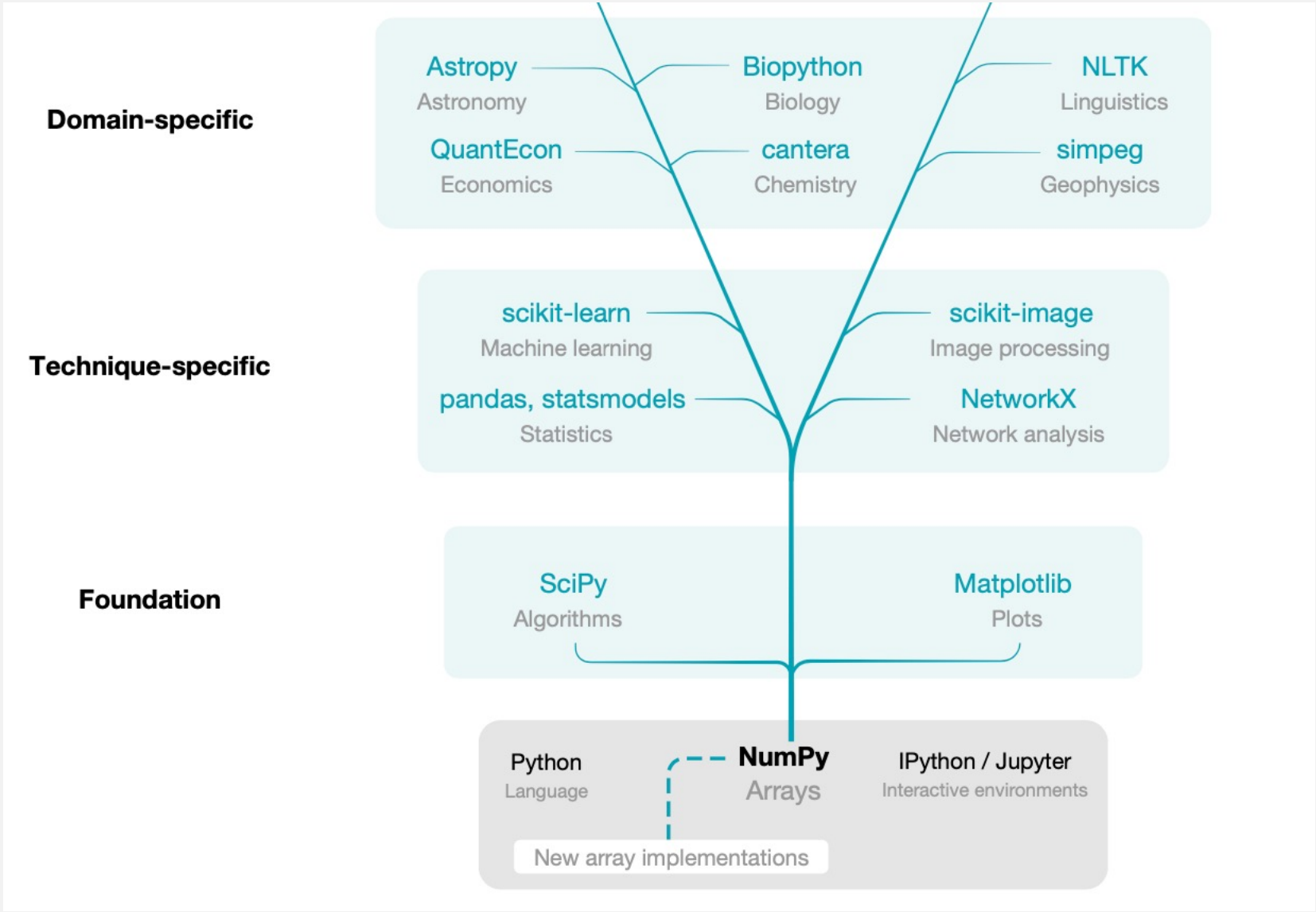
Published online: 16 September 2020

Open access

Charles R. Harris[1], K. Jarrod Millman[2,3,4]✉, Stéfan J. van der Walt[2,4,5]✉, Ralf Gommers[6]✉, Pauli Virtanen[7,8], David Cournapeau[9], Eric Wieser[10], Julian Taylor[11], Sebastian Berg[4], Nathaniel J. Smith[12], Robert Kern[13], Matti Picus[4], Stephan Hoyer[14], Marten H. van Kerkwijk[15], Matthew Brett[2,16], Allan Haldane[17], Jaime Fernández del Río[18], Mark Wiebe[19,20], Pearu Peterson[6,21,22], Pierre Gérard-Marchant[23,24], Kevin Sheppard[25], Tyler Reddy[26], Warren Weckesser[4], Hameer Abbasi[6], Christoph Gohlke[27] & Travis E. Oliphant[6]

- 2020年NumPy团队在Nature正刊发表论文，详细介绍了Numpy的历史、特性，并对其现状和未来进行了总结和展望

NumPy是Python科学计算生态的基础



NumPy的下载安装

- NumPy不是Python内置的标准库，需要单独下载安装

  pip install numpy

或 conda install numpy

- 国内使用pip或conda自身服务器比较慢，建议使用镜像服务器（清华、阿里云、豆瓣、中科大等）
- Anaconda自带了Numpy等科学计算模块，可直接使用

pip镜像服务器设置

- 临时使用：

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple
some-package
```

- 长久使用：

```
pip config set global.index-url
https://pypi.tuna.tsinghua.edu.cn/simple
```

conda镜像服务器设置

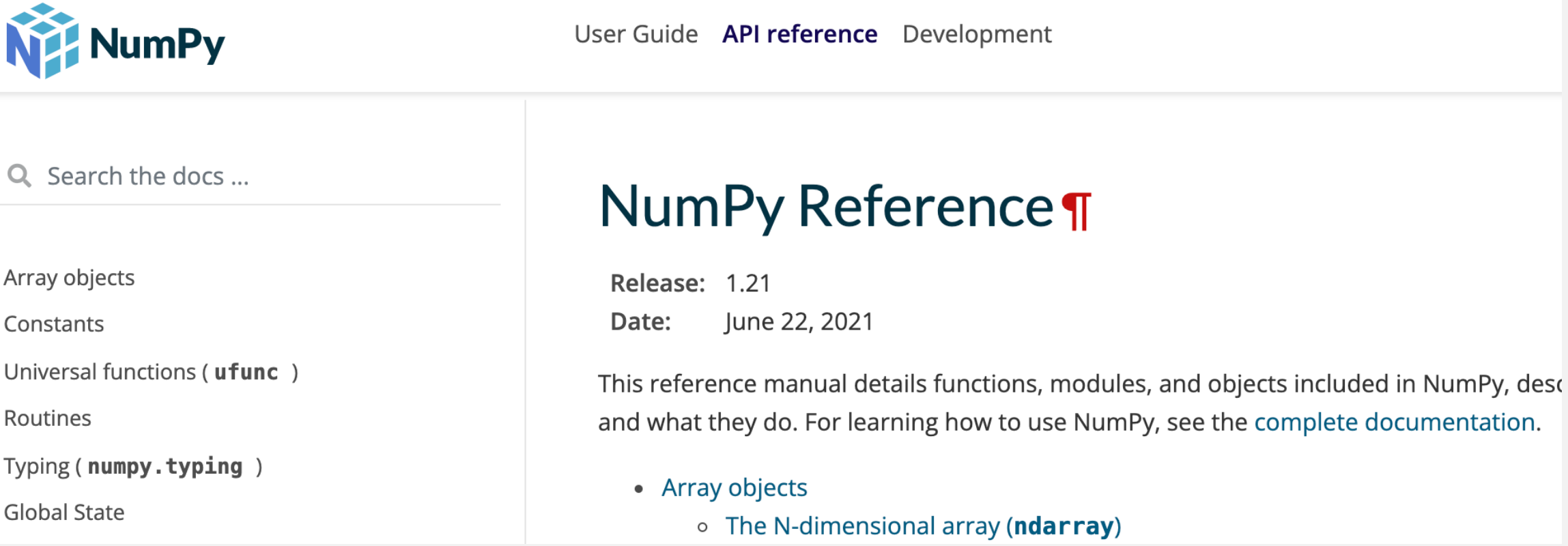
- 添加镜像：

```
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
```

- 查看配置：

```
conda config --show
```

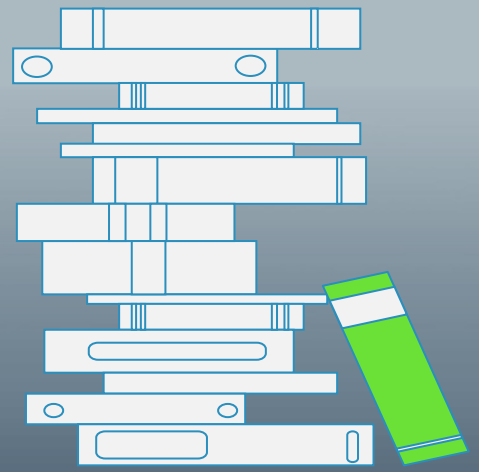
numpy官网和文档



https://numpy.org/doc/stable/

02

多维数组



创建多维数组

```
import numpy as np

arr1 = np.array([1, 2, 3])  # 一维数组

arr2 = np.array([[1, 2, 3], [4, 5, 6]])  # 二维数组
```

- 在Jupyter Notebook下，可以通过对象名来直接查看对象，其显示效果和print语句有所不同

多维数组的属性

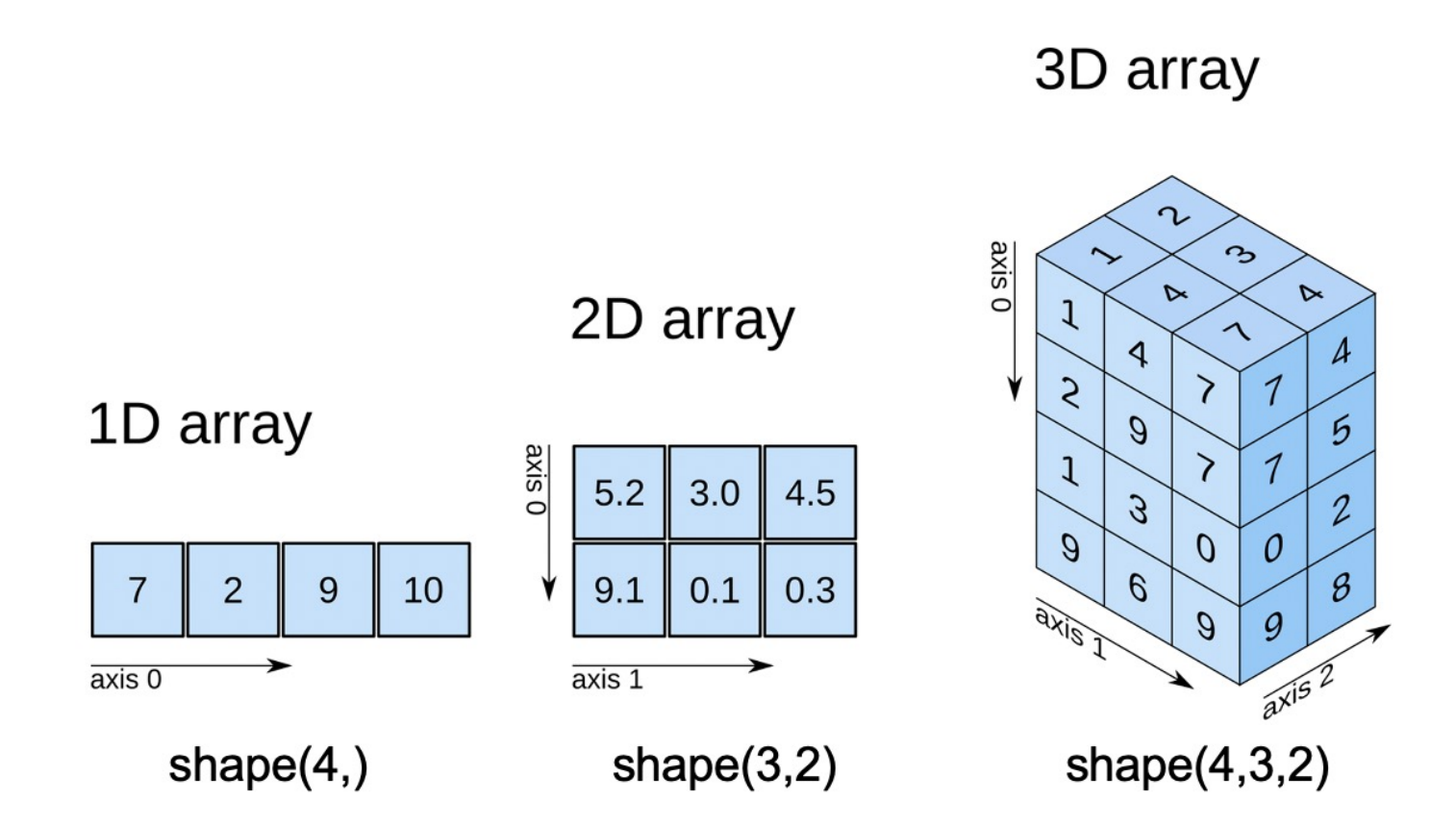
属性名	意义	备注
ndim	数组维度	
shape	数组形状	包含各维度大小的元组
dtype	元素数据类型	int64等

- 尽管多维数组元素的类型可以不同，但强烈建议不要这样做

多维数组的属性

属性名	意义	备注
size	元素数量	
itemsize	元素内存大小	和dtype对应
flags	内存信息描述	默认C风格的数组保存方法
strides	跨度	从当前维度到下一维度所需要“跨过”的字节数

多维数组的轴



可以理解为：设axis = i，则沿着第i个下标变化的方向进行操作

多维数组的创建

方法名	参数	备注
np.array	原生列表或元组	将原生列表或元组等转换为数组
np.arange	起点/终点/步长	可以使用实数
np.zeros	shape	全零实数数组
np.ones	shape	全一实数数组
np.full	shape/数值	指定数值填充
np.eye	长度	单位矩阵

多维数组的创建

方法名	参数	备注
np.linspace	等差数列起点/终点/项数	endpoint参数设置0是1否取到终点

```
import numpy as np

x = np.linspace(10, 20, 5)

print(x)

y = np.linspace(10, 20, 5, endpoint=False)

print(y)
```

多维数组的创建

方法名	参数	备注
np.logspace	等比数列起点/终点/项数/底	底默认为10

```
import numpy as np

x = np.logspace(1, 5, num=3, base=2)

print(x)

y = np.logspace(2, 3, num=4, base=10)

print(y)
```

NumPy中的随机数函数

函数名	功能	函数名	功能
rand	0和1之间的随机实数	normal	正态分布的随机数
randint	指定范围内的随机整数	uniform	均匀分布
randn	标准正态的随机数	poisson	泊松分布
choice	随机抽取样本	shuffle	随机打乱顺序

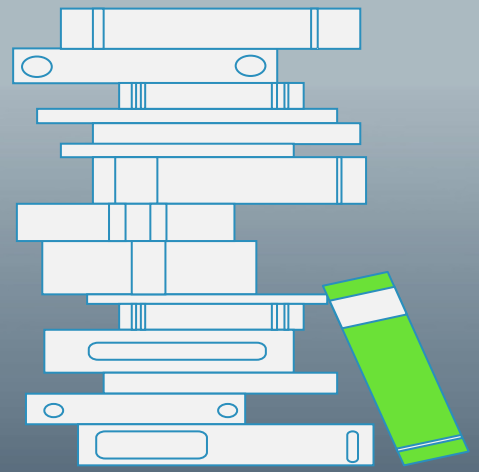
NumPy中包含了random模块，提供了实用的随机数功能

NumPy中的随机数函数

```
import numpy as np

x = np.random.randint(0, 100, (3, 5))

print(x)

y = np.random.normal(0, 1, (3, 4))

print(y)
```

03

形状与重构



修改多维数组的形状

- 多维数组的形状可以自由灵活转变，但需要注意两个问题
  - 改变是否就地改变
  - 得到的新数组空间是否独立
- 设计轴变换的重构比较抽象，需要重点掌握

修改多维数组的形状

```
import numpy as np

a = np.arange(1, 7)

a.reshape(2, 3)  # 不是就地修改

print(a)

b = a.reshape(2, 3)  # a和b共享内存

print(b)

b[0][0] = 100

print(a)
```

修改多维数组的形状

```
import numpy as np
a = np.arange(1, 7)
a.resize(2, 3)   # 就地修改
print(a)
a.resize(2, 4)   # 不足就补0或空
print(a)
a.resize(2, 2)   # 超过则丢弃
print(a)
```

轴的交换

```
import numpy as np

a = np.arange(1, 7)

a.resize(2, 3)

b = a.swapaxes(0, 1)

print(a, a.strides)

print(b, b.strides)
```

轴的交换

```
import numpy as np

a = np.arange(1, 9).reshape(2, 2, 2)

b = a.swapaxes(2, 0)  # 仍然共享同一片区域，存储细节变化

print(b)
```

轴的交换

```
import numpy as np

a = np.arange(1, 7).reshape(2, 3)

b = a.T

c = np.transpose(a) # 这里都是行列互换

print(b)

print(c)
```

轴的滚动

```
import numpy as np

a = np.arange(1, 9).reshape(2, 2, 2)

b = np.rollaxis(a, 2)  # 下标顺序由012改成201

c = np.rollaxis(a, 2, 1)  # 下标顺序由012改成021

print(a, b, c, sep='\n\n')
```

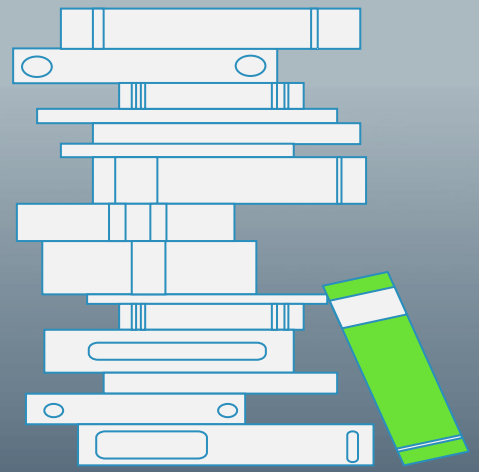
类似轴和维度的操作都不改变数据本身，而是修改了strides

折叠与转换

```
import numpy as np

a = np.arange(1, 7).reshape(2, 3)

b = a.flatten()

print(b)

c = a.tolist()

print(c)
```

04

切片与读取



多维数组的切片

- Numpy中的切片与列表等基本相同，但用法更灵活多变
- 切片的结果产生新的数组，但默认和原数组同一区域
- Numpy切片支持使用省略号来代表所有索引

```
import numpy as np

a = np.arange(1, 7)

b = a[2:4]

b[0] = 100

print(a)
```

多维数组的切片

- 多维数组的每个维度可以分别进行切片

```
import numpy as np

a = np.arange(1, 17).reshape(4, 4)

print(a)

print(a[0, 2:4])

print(a[2:, 2:])

print(a[0::2, ::2])
```

花式索引

- 在索引时可提供指定的位置列表，打破规律限制
- 花式索引的结果是新的数组

```
import numpy as np
a = np.arange(1, 17).reshape(4, 4)
print(a)
b = a[[0, 1, 3], 1:]
print(b)
b[0] = 100
print(a)
```

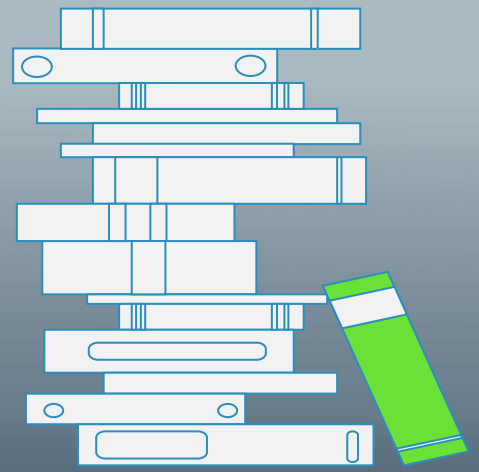
布尔索引

- 在索引时可提供指定的逻辑表达式，数组会进行广播运算

```
import numpy as np

a = np.arange(1, 17).reshape(4, 4)

b = a[a > 10]

print(b)
```

05

数学运算与广播



广播原则

- 当操作两个数组时，NumPy会逐个比较它们的shape
- 满足以下条件则称两个数组兼容
  - 相等
  - 其中一个为1
- 如果两个数组兼容，则可以对其进行广播运算，也就是将两个数组的维度拉平

广播示例

```
import numpy as np

a = np.arange(1, 7).reshape(2, 3)

b = a + 1

print(b)
```

本例当中，将数字1广播为和a相同形状的数组

广播示例

```
import numpy as np

arr1 = np.arange(1, 13).reshape(4, 3)

arr2 = np.array([10, 10, 10])

print(arr1 + arr2)
```

本例当中，将arr2的形状由(3,)广播为(4, 3)

数学运算

- 除了支持运算符之外，NumPy本身还提供了大量的数学函数，如

  arr1 + arr2可以写成np.add(arr1, arr2)

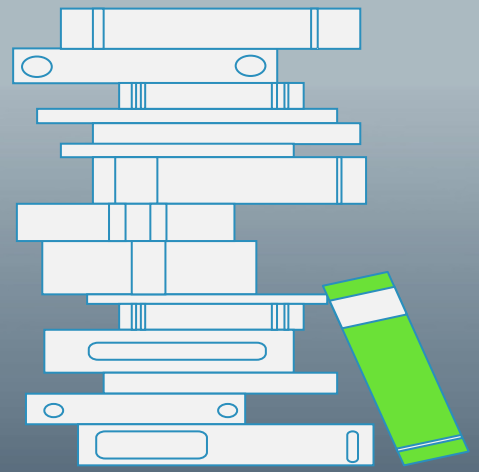
二元函数名	功能	二元函数名	功能
add	加法	power	幂运算
subtract	减法	fmax	更大值
multiply	乘法	fmin	更小值
divide	除法	greater, greater_equal	大于，大于等于
less, less_equal	小于，小于等于	equal,not_equal	相等，不等
logical_and	逻辑与	logical_or	逻辑或
logical_xor	逻辑异或		

数学运算

一元函数名	功能	一元函数名	功能
abs, fabs	绝对值	sqrt, square, exp	根号/平方/e的指数
log, log10, log2	对数	sign, ceil , floor	符号/向上取整/向下取整
around/rint	四舍五入/四舍五入到整数	modf	拆分为小数和整数数组
sin, cos, tan	三角函数	sinh, cosh, tanh	双曲型三角函数
logical_not	逻辑非	isnan	是否为非数字类型

06

切割、堆叠与操纵



切割

```
import numpy as np

x = np.arange(11, 19)

x1, x2, x3 = np.split(x, [3, 5])   # 按列表的顺序切割

print(x1, x2, x3)
```

切割

```
import numpy as np

x = np.arange(1, 49).reshape(6, 8)

y = np.vsplit(x, 3)  # 分成3份

print(y)

z = np.vsplit(x, [2, 4])  # 试试hspilt

print(z)
```

切割

```
import numpy as np

x = np.arange(1, 25).reshape(2, 3, 4)

upper, lower = np.split(x, [1], axis=0)  # 试试axis = 1

print(upper, lower)
```

堆叠

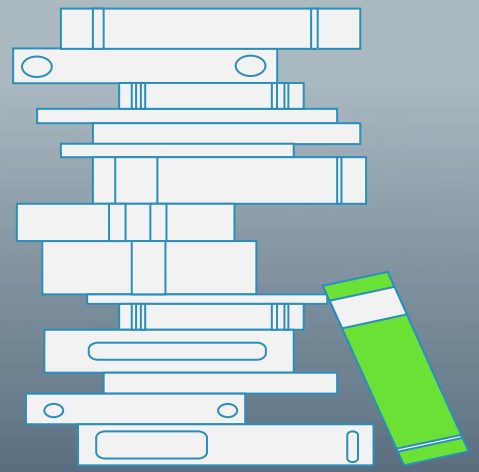
```
import numpy as np
x = np.arange(1, 25).reshape(2, 3, 4)
upper, lower = np.split(x, [1], axis=0)
a = np.concatenate((upper, lower), axis=0)
b = np.vstack([upper, lower])
c = np.hstack([upper, lower])
print(a, b, c, sep='\n\n')
print(b.shape, c.shape)
```

操纵

```
import numpy as np
x = np.arange(8)
print(np.delete(x, 2))
y = np.arange(9).reshape(3, 3)
print(np.delete(y, [1], axis=0))
print(np.insert(x, 0, -1))
print(np.insert(y, 1, [0, 0, 0], axis=0))
```

07

排序、统计与迭代



排序

```
import numpy as np
my_array = np.array([11, 18, 13, 12, 19, 15, 14, 17, 16])
print(my_array)
print(np.sort(my_array))
print(my_array)
print(np.argsort(my_array)) # 排序后的索引
```

按轴排序

```
import numpy as np
my_array2 = np.array([[21, 22, 23, 24, 25],
                      [35,  34, 33, 32, 31],
                      [1, 2,  3, 100, 4]])
print(np.sort(my_array2, axis=0))
# np.sort(MyArray, axis=1)
```

统计函数

- NumPy中有很多有用的统计函数，用于从数组中给定的元素/轴中查找最小，最大，百分标准差和方差等
- 配合统计条件可以很简便地抽取所需数据

统计函数——最大值

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

统计函数——argmax

```
import numpy as np
a = np.array([[30, 40, 70], [80, 20, 10], [50, 90, 60]])
print(np.argmax(a))
print(np.argmax(a, axis=0))
print(np.argmax(a, axis=1))
```

统计函数——百分位数

```
import numpy as np
a = np.array([[3, 7, 5], [8, 4, 3], [2, 4, 9]])
print(np.percentile(a, 50))
print(np.percentile(a, 50, axis=0))
print(np.percentile(a, 50, axis=1))
```

统计函数——中位数

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

统计函数——平均值

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

统计函数——加权平均

```
import numpy as np
a = np.array([1, 2, 3, 4])
b = np.average(a)
wts = np.array([4, 3, 2, 1])
c = np.average(a, weights=wts)
print(c)
```

统计函数——方差与标准差

```
import numpy as np
print(np.std([1, 2, 3, 4]))  # 标准差
print(np.var([1, 2, 3, 4]))  # 方差
```

统计函数——nan、any和all

```
import numpy as np
a = np.array([1, 2, 3, np.nan])
print(np.isnan(a))
print(np.any(np.isnan(a)))
print(np.all(np.isnan(a)))
print(np.sum(a))
print(np.nansum(a))
```

统计函数——where和extract

```
import numpy as np
a = np.array([[30, 40, 0], [0, 20, 10], [50, 0, 60]])
print(np.nonzero(a))
b = np.where(a > 40)
print(b)
print(a[b]) # array索引
c = np.extract(np.mod(a, 3) == 0, a)
print(c)
```

统计函数——bincount与众数

```
import numpy as np
nums = [1, 2, 3, 6, 5, 6, 6]
a = np.bincount(nums)  # 返回0～最大值出现的次数
np.argmax(a)  # 非负整数的众数
```

数组上的迭代

```
import numpy as np
a = np.arange(6).reshape(2, 3)
for x in np.nditer(a):
    print(x)
    print(type(x))  # x也是ndarray
```

NumPy包含一个迭代器对象numpy.nditer，可使用Python的标准Iterator接口来访问

数组上的迭代

```
import numpy as np
a = np.arange(6).reshape(2, 3)
b = a.T
for x in np.nditer(b):
    print(x)
    print(type(x))
```

迭代匹配的是数组内容上的顺序，转置函数等操作不影响数组内容排序

数组上的迭代

```
import numpy as np
a = np.arange(6).reshape(2, 3)
c = a.copy(order='F')  # fortran风格
for x in np.nditer(c):
    print(x)  # 输出和前面不同
```

理解strides属性对理解Numpy内存管理很关键

数组上的迭代

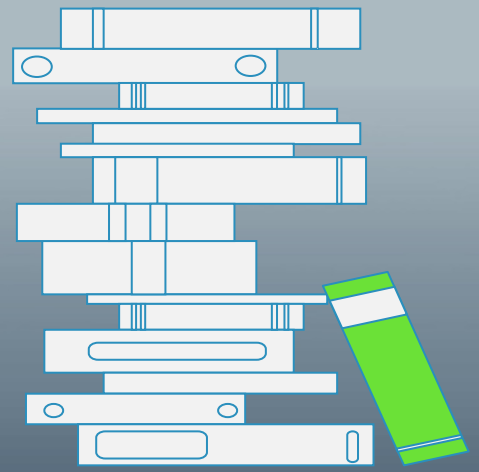
```
import numpy as np
x = np.arange(1, 7).reshape(2, 3)
for a in np.nditer(x, op_flags=['readwrite']):
    a[...] = a * 2  # 这里很神奇!
print(x)
```

数组上的迭代

```
import numpy as np
a = np.arange(12).reshape(3, 4)
b = np.arange(4)
for x, y in np.nditer([a, b]):  # 类似广播的效果
    print(f"{x}:{y}")
```

08

字符串



Numpy中的字符串操作

```
import numpy as np


a = np.char.add(['hello'], [' seu'])
b = np.char.add(['hello', 'hi'], [' seu', ' baby'])
print(a)
print(b)
```

对dtype为numpy.string_或numpy.unicode_的数组执行向量化字符串操作，在numpy.char类当中定义

Numpy中的字符串操作

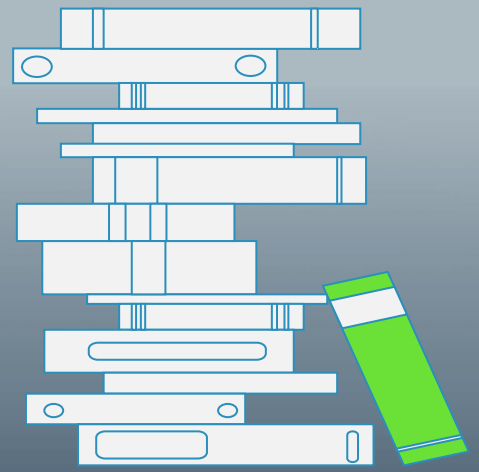
```
import numpy as np

a = np.char.multiply('Hello', 3)

b = np.char.center('hello', 20, fillchar='*')

c = np.char.capitalize('hello world')

d = np.char.title('hello how are you?')

e = np.char.lower(['HELLO', 'WORLD'])

f = np.char.upper(['hello', 'world'])
```

Numpy中的字符串操作

```
import numpy as np
a = np.char.split('TutorialsPoint,Hyderabad,Telangana',
sep=',')
b = np.char.splitlines('hello\rhow are you?')  # \n\r 都是
分行符
c = np.char.strip(['arora', 'admin', 'java'], 'a')
d = np.char.join(':', 'dmy')
e = np.char.join([':', '-'], ['dmy', 'ymd'])
f = np.char.replace('He is a good boy', 'is', 'was')
g = np.char.encode('我', 'utf8')
h = np.char.decode(g, 'utf8')
```

09

视图、复制与文件IO



引用

```
import numpy as np
a = np.arange(6).reshape(2, 3)
b = a
b.resize(3, 2)
print(a.shape)
print(a)
```

视图

```
import numpy as np
a = np.arange(6).reshape(2, 3)
b = a.view()
b.resize(3, 2)
print(a.shape)
print(a)
b[0][0] = 7
print(a)
```

复制

```
import numpy as np
a = np.arange(6).reshape(2, 3)
b = a.copy()
b.resize(3, 2)
print(a.shape)
print(a)
b[0][0] = 7
print(a)
```

文件IO

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
np.save('outfile', a)  # 默认使用python序列化生成一个npy文件
b = np.load('outfile.npy')
print(b)
```

文件IO

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

10

Numpy实战案例