PYTHON : NumPy

2019.11.21

- 14:20-18:00
- 不要早退

编写一个Python程序和一个C语言程序，令一个变量初始值为0，对其累加100000000（一亿次），打印程序运行时间

Python:

```
import time
time.perf_count()
start = time.time()
 …
end = time.time()
elapse = end - start
```

C:

```
  #include <time.h>
  clock_t start, end;
  float elapse;
  start=clock();
…
  end=clock();
  elapse = float(end-start)/CLOCKS_PER_SEC;
```

课堂测试

在14：45之前把源文件和程序运行截图发送到助教邮箱 fangf26@mail2.sysu.edu.cn

以邮件发送时间为准，邮件标题统一为：测试1-学号-姓名

Anaconda

pip install scipy

快速数组计算

绘图

符号计算

表格处理和数据分析

函数库

http://old.sebug.net/paper/books/scipydoc/index.html

ndarray对象

排序检索

ufunc运算

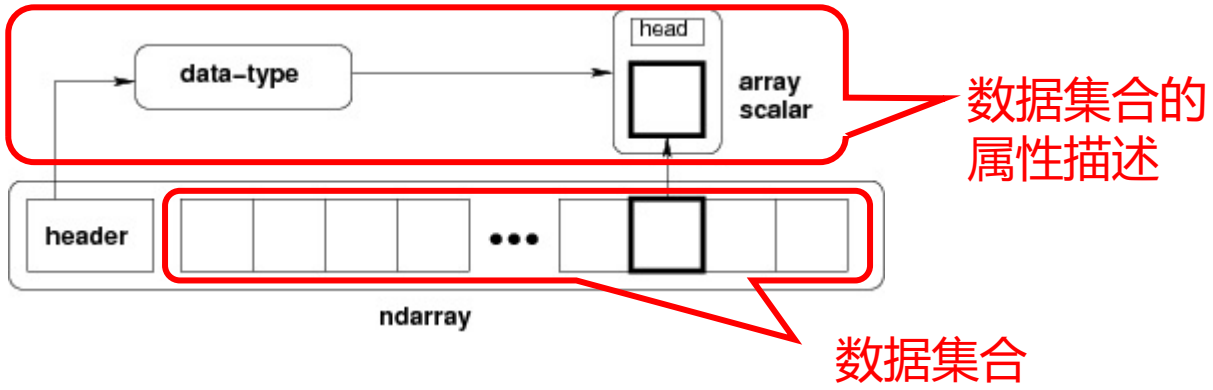
文件存取

矩阵运算

```
import numpy as np

np.array()
...
```

Numpy



N 维数组类型。 它描述相同类型的元素集合。

形式上像列表（list）

专门优化用于计算

ndarray 对象

numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)

数据类型，默认自动

维度，默认自动

多维数组的存储，'C', 'F'

```
a = np.array([1,2,3]) #建立一个ndarray数组，内容为[1,2,3]

b = np.array((1,2,3)) #可以从任何暴露数组接口的对象，或返回数组的任何方法创建一个ndarray。

c = np.array([[1, 2, 3, 4],[4, 5, 6, 7], [7, 8, 9, 10]])

#如果传递的是多层嵌套的序列，将创建多维数组。
```

暴露数组接口：list(object)起作用

ndarray 创建I

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

```
x = np.empty((3,2)) #产生空数组np.array(shape=(3,2))
```

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

```
#产生全部填充0的数组
```

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

```
#产生全部填充1的数组
```

ndarray 创建II

numpy.arange(start=0, stop, step=1, dtype=None)

```
x = np.arange(5)     #array([1,2,3,4,5])
x = np.arange(2,10,1.5) #array([2., 3.5, 5., 6.5, 8., 9.5])
```

numpy.linspace(start, stop, num, endpoint=True, retstep=False)

#产生从start到stop（默认包含）的线性均匀分布的有num个元素的数组

是否返回间隔值

numpy.logspace(start, stop, num, endpoint=True, retstep=False)

#产生从start到stop（默认包含）的对数均匀分布的有num个元素的数组

ndarray 创建III

numpy.fromfunction(func, shape)

```
def func(i):
    return i%4+1

np.fromfunction(func, (10,))

def func2(i, j):
    return (i+1)*(j+1)

np.fromfunction(func2, (9,9))
```

注意一维数组的shape的格式

函数的参数必须对应数组指标

array[i,j]=func2(i,j)

ndarray 创建III

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

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

ndarray.shape

print(a.shape) (2,3)

a.shape=(3,2)

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

正常使用中不直接修改属性

ndarray.dtype

print(a.dtype) dtype('int64')

ndarray.itemsize

ndarray.ndim

ndarray.data

ndarray.T

ndarray 属性

https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html

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

ndarray.max(axis)

b = a.max() 6

a.max(0) [4,5,6]

a.max(1) [3,6]

min(axis)

mean(axis)

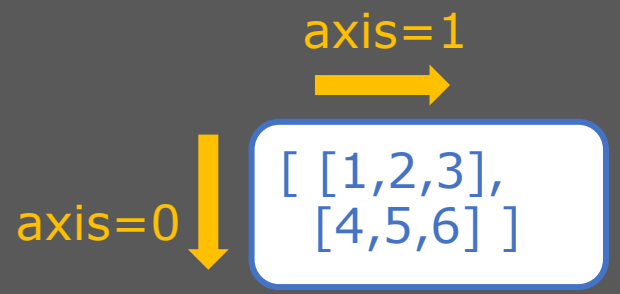
sum(axis)

ndarray.argmin(axis)

ndarray.reshape()

numpy.amax(ndarray, axis)

b = np.amax(a)



ndarray 方法

https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html

实例 1

```
import numpy as np # 使用标量类型
dt = np.dtype(np.int32)
print(dt)
```

输出结果为：

int32

实例 2

```
import numpy as np
# int8, int16, int32, int64 四种数据类型可以使用字符串 'i1', 'i2','i4','i8' 代替
dt = np.dtype('i4')
print(dt)
```

输出结果为：

int32

数据类型

字符	对应类型
b	布尔型
i	(有符号) 整型
u	无符号整型 integer
f	浮点型
c	复数浮点型
m	timedelta (时间间隔)
M	datetime (日期时间)
O	(Python) 对象
S, a	(byte-)字符串
U	Unicode
V	原始数据 (void)

名称	描述
bool_	布尔型数据类型 (True 或者 False)
int_	默认的整数类型 (类似于 C 语言中的 long，int32 或 int64)
intc	与 C 的 int 类型一样，一般是 int32 或 int 64
intp	用于索引的整数类型 (类似于 C 的 ssize_t，一般情况下仍然是 int32 或 int64)
int8	字节 (-128 to 127)
int16	整数 (-32768 to 32767)
int32	整数 (-2147483648 to 2147483647)
int64	整数 (-9223372036854775808 to 9223372036854775807)
uint8	无符号整数 (0 to 255)
uint16	无符号整数 (0 to 65535)
uint32	无符号整数 (0 to 4294967295)
uint64	无符号整数 (0 to 18446744073709551615)
float_	float64 类型的简写
float16	半精度浮点数，包括：1 个符号位，5 个指数位，10 个尾数位
float32	单精度浮点数，包括：1 个符号位，8 个指数位，23 个尾数位
float64	双精度浮点数，包括：1 个符号位，11 个指数位，52 个尾数位
complex_	complex128 类型的简写，即 128 位复数
complex64	复数，表示双 32 位浮点数 (实数部分和虚数部分)
complex128	复数，表示双 64 位浮点数 (实数部分和虚数部分)

结构数组

假设我们需要定义一个结构数组，它的每个元素都有name, age和weight字段。在NumPy中可以如下定义：

```
import numpy as np
persontype = np.dtype({
    'names':['name', 'age', 'weight'],
    'formats':['S32','i', 'f']})
a = np.array([("Zhang",32,75.5),("Wang",24,65.2)],
    dtype=persontype)
```

在C语言中我们可以通过struct关键字定义结构类型，结构中的字段占据连续的内存空间，每个结构体占用的内存大小都相同，因此可以很容易地定义结构数组。

和C语言一样，在NumPy中也很容易对这种结构数组进行操作。只要NumPy中的结构定义和C语言中的定义相同，NumPy就可以很方便地读取C语言的结构数组的二进制数据，转换为NumPy的结构数组。

```
print(a[0]['name'])
print(a[0][0])
```

```
a.tofile("bindata")
b = np.fromfile("bindata",
                dtype=persontype)
print(a)
print(b)
```

```
>>> a[0,3:5]
array([3,4])
>>> a[4:,4:]
array([[44,45],[54,55]])
>>> a[:,2]
array([2,12,22,32,42,52])
>>> a[2::2,::2]
array([[20,22,24],
       [40,42,44]])
```

ndarray.reshape, ndarray.T, …

切片给出的是数据视图，赋值给另一个变量后数据空间共享

[起始:结束:步长]

[起始:结束]

[位置]

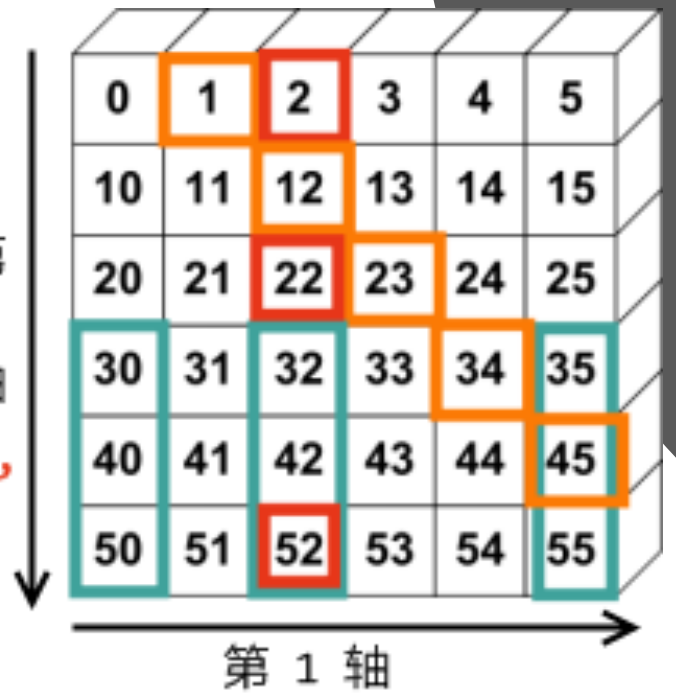
切片和索引 I

大部分的ndarray方法返回的数组都和原数组共享数据空间

切片和索引实际上也是调用内置方法

ndarray.copy()
numpy.copy(ndarray)

```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([1,12,23,34,45])
>>> a[3:,[0,2,5]]
array([[30,32,35],
       [40,42,45],
       [50,52,55]])
>>> mask=np.array([1,0,1,0,0,1],
                  dtype=np.bool)
>>> a[mask,2]
array([2,22,52])
```



a[(0,1,2,3,4),(1,2,3,4,5)]：用于存取数组的下标和仍然是一个有两个元素的组元，组元中的每个元素都是整数序列，分别对应数组的第0轴和第1轴。从两个序列的对应位置取出两个整数组成下标： a[0,1], a[1,2], ..., a[4,5]。

a[3:, [0, 2, 5]]：下标中的第0轴是一个范围，它选取第3行之后的所有行；第1轴是整数序列，它选取第0, 2, 5三列。

a[mask, 2]：下标的第0轴是一个布尔数组，它选取第0，2，5行；第1轴是一个整数，选取第2列。

切片和索引 II

```
x = np.linspace(0, 2*np.pi, 10)
y = np.sin(x)
print(y)
```

```
array([ 0.00000000e+00,  6.42787610e-01,  9.84807753e-01,
        8.66025404e-01,  3.42020143e-01, -3.42020143e-01,
       -8.66025404e-01, -9.84807753e-01, -6.42787610e-01,
       -2.44921271e-16])
```

ufunc是对数组的每个元素进行操作的函数

速度远远快于for循环（底层是用C语言写的）

调用ufunc方法的运算简写

y = x1 + x2 :	add(x1, x2 [, y])
y = x1 - x2 :	subtract(x1, x2 [, y])
y = x1 * x2 :	multiply (x1, x2 [, y])
y = x1 / x2 :	divide (x1, x2 [, y]), 如果两个数组的元素为整数，那么用整数除法
y = x1 / x2 :	true divide (x1, x2 [, y]), 总是返回精确的商
y = x1 // x2 :	floor divide (x1, x2 [, y]), 总是对返回值取整
y = -x :	negative(x [,y])
y = x1**x2 :	power(x1, x2 [, y])
y = x1 % x2 :	remainder(x1, x2 [, y]), mod(x1, x2, [, y])

ufunc (universal function)

将一个数组中的所有大于0的值取对数

```
for i in range(len(a)):
    if a[i]>0:
        a[i]=math.log10(a[i])
```

```
mask = a>0

a[mask] =np.log10(a[mask])

a[a>0] =np.log10(a[a>0])
```

遮蔽用生成一个布尔数组mask，mask中对应a中元素大于0的位置值为True，小于0处为False

用mask索引a中相应的元素，用numpy中的ufunc log10对全题数组计算对数，并赋值回原位置

可以省去mask数组（节省空间，耗费时间）

快速循环

```
a = np.arange(0, 60, 10).reshape(-1, 1)
b = np.arange(0, 5)
c = a+b
```

a: [[0], [10], [20], [30], [40], [50]]
a.shape: (6,1)

0	0	0	0	0
10	10	10	10	10
20	20	20	20	20
30	30	30	30	30
40	40	40	40	40
50	50	50	50	50

b: [[0, 1, 2, 3, 4]]
b.shape: (1,5)
b: [0, 1, 2, 3, 4]
b.shape: (5,)

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
0	1	2	3	4

c.shape: (6,5)

0	1	2	3	4
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34
40	41	42	43	44
50	51	52	53	54

广播

当我们使用ufunc函数对两个数组进行计算时，ufunc函数会对这两个数组的对应元素进行计算，因此它要求这两个数组有相同的大小(shape相同)。如果两个数组的shape不同的话，会进行如下的广播(broadcasting)处理：

1.让所有输入数组都向其中shape最长的数组看齐，shape中不足的部分都通过在前面加1补齐
2.输出数组的shape是输入数组shape的各个轴上的最大值
3.如果输入数组的某个轴和输出数组的对应轴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
4.当输入数组的某个轴的长度为1时，沿着此轴运算时都用此轴上的第一组值

线性代数

函数	描述
dot(array,array) @ 运算符	一维数组：内积； 多维数组：矩阵乘法; dot(a, b)[i,j] = sum(a[i,:] * b[:,j]) dot(数组,标量)：对数组内所有元素乘以标量;
vdot(array,array)	两个矩阵对应位置元素乘积之和 vdot(a, b) = sum(a*a)
inner(array,array)	一维数组的内积，多维时取最后一维的内积 inner(a, b)[i,j] = sum(a[i,:]*b[j,:])
outer(array,array)	一维数组的外积，多维数组也展平为一维计算 outer(a, b)[i,j] = a.flatten()[i]*b.flatten()[j]
matmul	基本等同于dot
linalg.det(array)	数组的行列式
linalg.solve(a,x)	求解线性矩阵方程
linalg.inv(array)	计算矩阵的乘法逆矩阵

使用numpy的函数库

ndarray中有部分方法实现相同的功能

统计函数

函数	描述
amin(array, axis)	最小值
amax(array,axis)	最大值
mean(array,axis)	平均值
average (a,axis,weights)	平均数，与weights结合计算加权平均值
median(array,axis)	中位数
std(array, axis)	标准差
ptp(array)	数组中元素最大值与最小值的差
percentile(a,q,axis)	数组a在q%处的百分位数
var(array, axis)	方差
histogram(array, bins=10, range=None, weights=None, Density=None)	计算数组元素的直方分布，返回值为两个数组第一个是直方图的Y值，第二个是直方图的区间划分。bin指定区间划分数量，或是用一个数组指定，range手动指定统计区间上限和下限，weights声明统计时每个元素的权重，density=True时返回的时概率密度分布，否则返回数量分布。

numpy.sort(a, axis, kind, order)

参数说明：
•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， axis=0 按列排序，axis=1 按行排序
•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

```
dt = np.dtype([('name', 'S10'),('age',
int)])
a =
np.array([("raju",21),("anil",25),("ravi",
17), ("amar",27)], dtype = dt)
print (a)
print ('按 name 排序：')
print (np.sort(a, order = 'name'))
```

排序

numpy.argsort(a, axis, kind, order)

函数返回的是数组值从小到大的索引值。

```
x = np.array([3, 1, 2])
y = np.argsort(x)
print (y)
print (x[y])
for i in y:
    print (x[i], end=" ")
```

非直接排序

numpy.lexsort(keys, axis=-1)

函数返回按照多个数组内容排序的索引。

```
nm = ('raju','anil','ravi','amar’)
dv = ('f.y.', 's.y.', 's.y.', 'f.y.’)
nv = (6, 2, 4, 3)

ind = np.lexsort((dv,nm,nv))
```

第三顺序　第二顺序第一顺序

关键字排序

搜索

numpy.argmax(a)　numpy.argmin(a)

返回最大值或最小值的索引。

numpy.nonzero(a)

返回非0值的索引。

numpy.where(a)

返回满足给定条件元素的索引。

a>3: 返回和a形状一样的布尔数组

np.where(a>3):返回索引数组，a有n维就返回n个数组

a[condition]: 都可以用来返回元素

文件读写

ndarray.tofile(filename)

numpy.fromfile(filename, dtype)

np.save(filename,array)

np.savez(filename,name1=array1,name2=array2…)

a = np.load(filename)

多次save到同一个文件会顺序存储，可以用load依次读出

np.savetxt(filename,array , fmt="", delimiter=" ")

np.loadtxt(filename , fmt="", delimiter=" ")

三门问题

•现在有三扇门，只有一扇门有汽车，其余两扇门的都是山羊。
•汽车事前是等可能地被放置于三扇门的其中一扇后面。
•参赛者在三扇门中挑选一扇。他在挑选前并不知道任意一扇门后面是什麽。
•主持人知道每扇门后面有什么。
•如果参赛者挑了一扇有山羊的门，主持人必须挑另一扇有山羊的门。
•如果参赛者挑了一扇有汽车的门，主持人等可能地在另外两扇有山羊的门中挑一扇门。
•参赛者会被问是否保持他的原来选择，还是转而选择剩下的那一扇门。
转换选择可以增加参赛者拿到汽车的机会吗？

一维随机游走

假设一个醉鬼，走路时只会向前迈步或向后迈步，迈步的方向完全随机，步长一定，请问他在100步后离开原点距离为20的概率为多少。更进一步，在n步后离开原点距离为m的概率为多少？（m,n均为整数）

课堂练习

大量粒子在势场中的随机游走

（速度时间单位忽略）
有10000个粒子，每个粒子质量为1，初始在一个100x100的二维盒子中均匀分布。每个粒子在下一秒的位置为以其目前位置为中心的圆面上的均匀分布（即运动方向随机，运动距离随机）。在盒子中心有一个质量为100的黑洞，会对粒子施加引力。粒子之间没有引力。请计算经过10000秒后各个粒子的位置

作业