NumPy 数值计算基础

2018/1/20

数据分析：是指用适当的统计分析方法对收集来的大量数据进行分析，将它们加以汇总和理解并消化，以求最大化地开发数据的功能，发挥数据的作用。数据分析是为了提取有用信息和形成结论而对数据加以详细研究和概括总结的过程。

目的：把隐藏在一些看似杂乱无章的数据背后的信息提炼出来，总结出所研究对象的内在规律，获得有价值的产品、服务，或深刻的洞见!

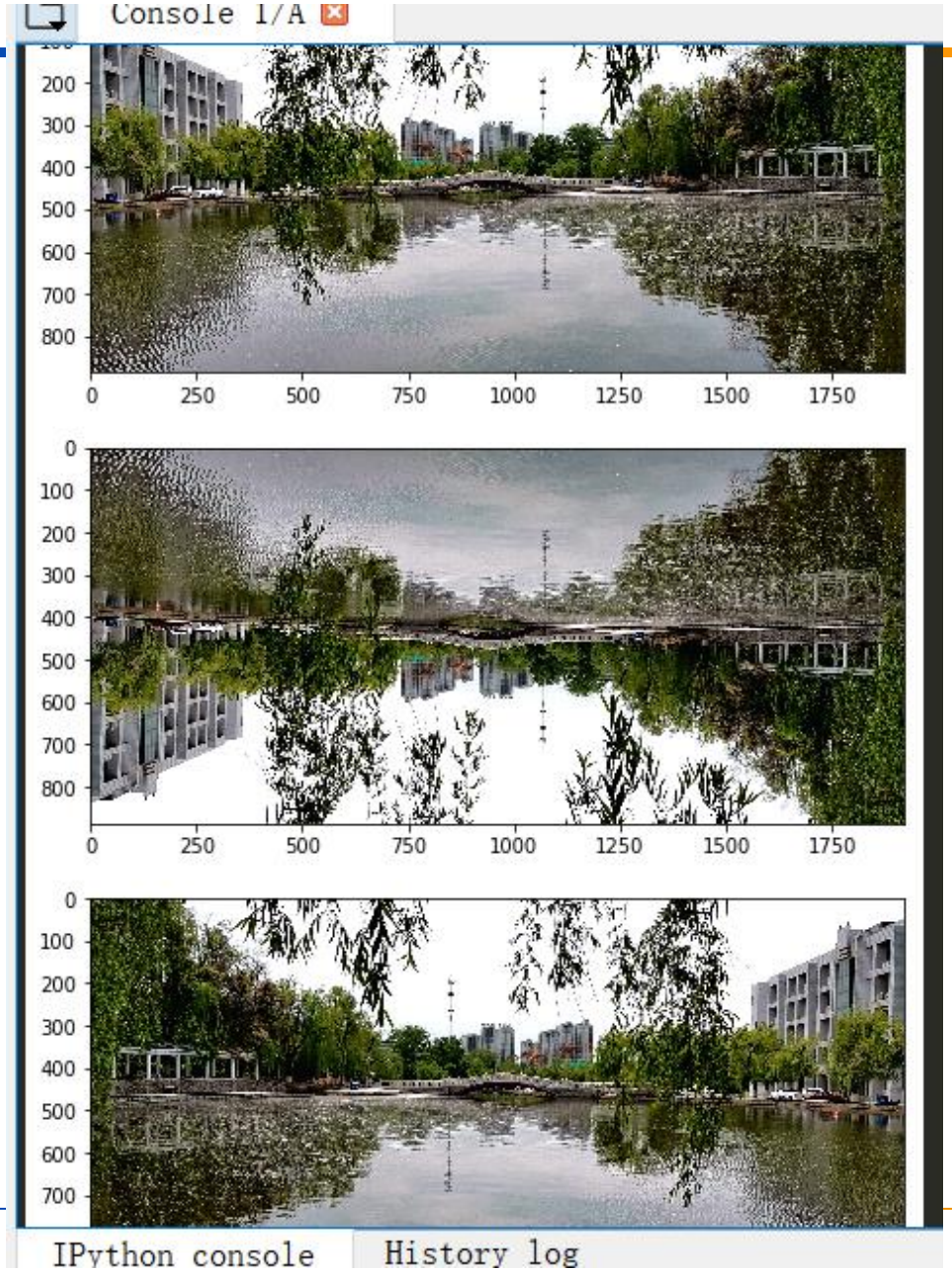
数据分析三剑客：**Numpy、Pandas、Matplotlib。**

数据在哪里：数值、图像、声音、文本。。。。。

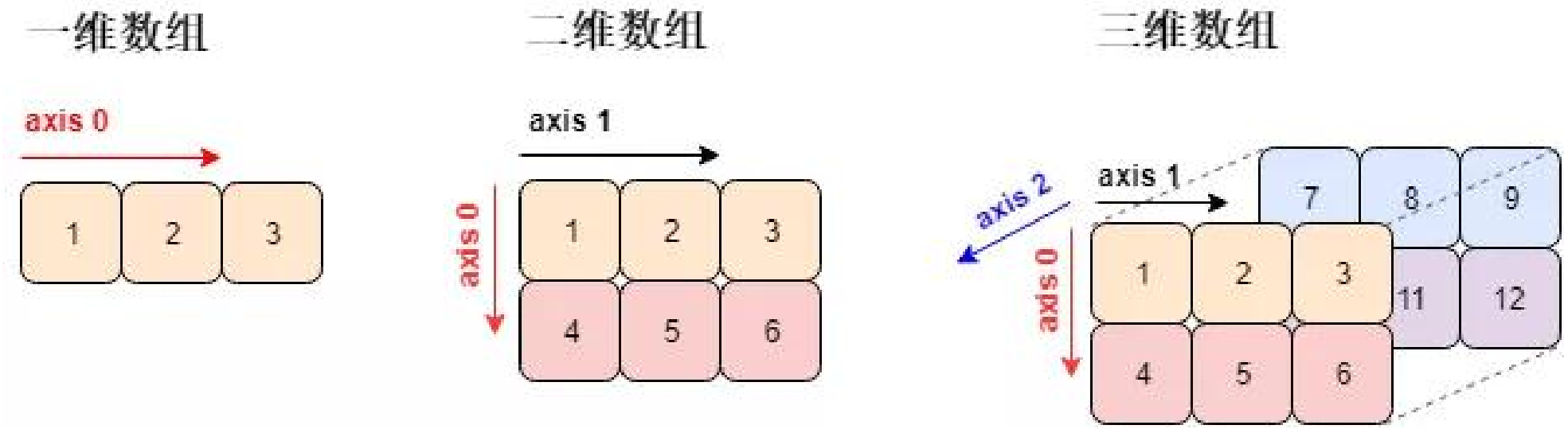
如何组织和存储，常量 变量 列表 集合 字典

```
import matplotlib.pyplot as plt
import numpy as np
ymh=plt.imread('ymlake.jpg')
print(type(ymh))
print(ymh.shape)
```

```
f, axs = plt.subplots(3,1,figsize=(15,15))
axs[0].imshow(ymh)
axs[1].imshow(ymh[::-1,:,:])
axs[2].imshow(ymh[:,::-1,:])
```



数组



数组 (array) 是相同类型的元素 (element) 的集合所组成数据结构 (data structure)。numpy 数组中的元素用的最多是「数值型」元素，平时我们说的一维、二维、三维数组长下面这个样子(对应着线、面、体)。四维数组很难被可视化。

注意一个关键字 axis，中文叫「轴」，一个数组是多少维度就有多少根轴。由于 Python 计数都是从 0 开始的，那么

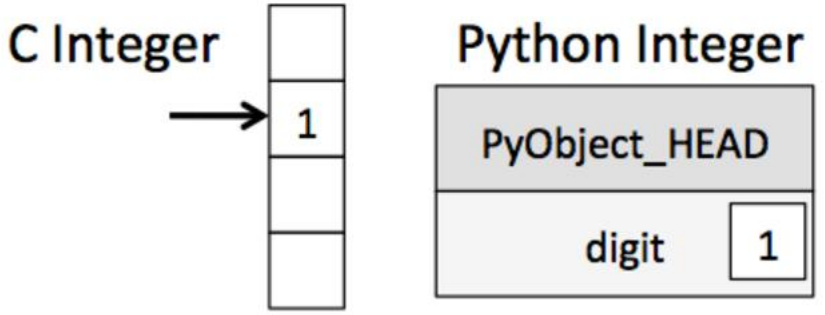
第 1 维度 = axis 0
第 2 维度 = axis 1
第 3 维度 = axis 2

目录

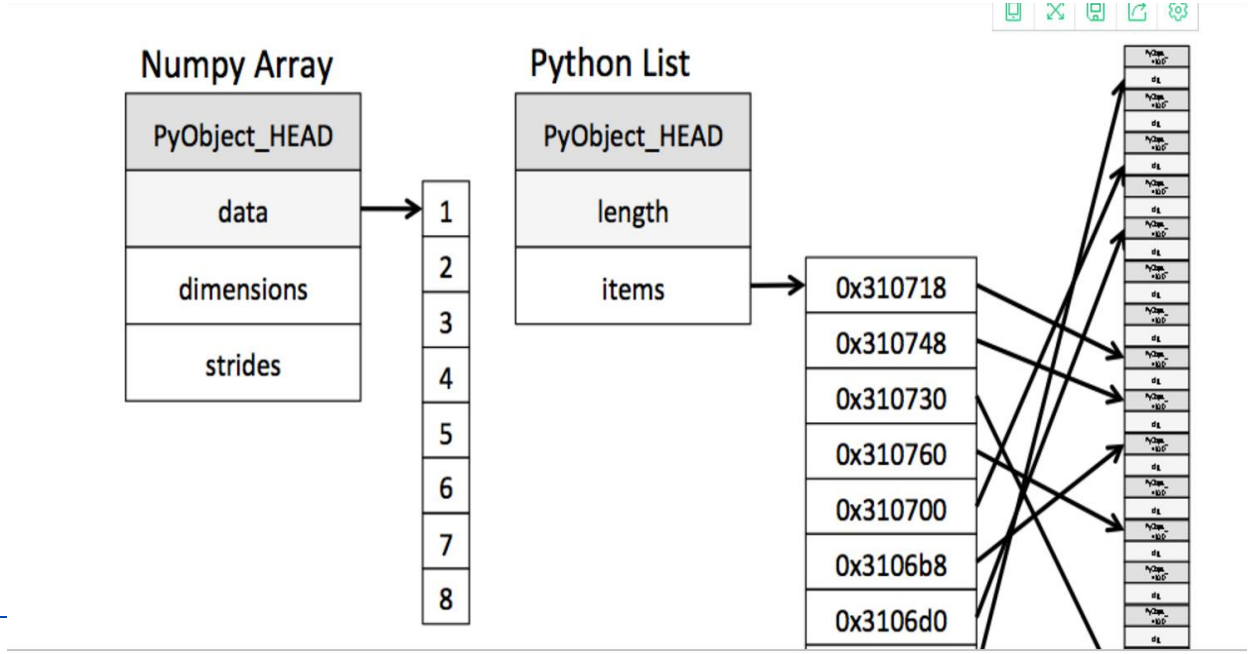
理解Python数据类型

C数据类型需明确声明，Python是动态的。任何数据可指定给任何变量。

- **Python对象都是C语言结构体。包含：值以及其它信息。**
- **优点：自由、动态的编码**
- **缺点：存储访问开销大、效率低**



- **Python组合数据类型，可作为同构或异构数组使用。**
- **在表达存储密集型(同类)数据时，很多信息就会多余**
- **不如存储于固定类型数组高效**
- **Numpy比List高效**
- **List比Numpy灵活**



理解Python数据类型

我们发现「numpy 数组」效率是「列表」效率的 27 倍左右。如果元素全是数值型变量 (numerical variable)，那么 numpy 数组明显是个很好的数据结构。

为什么要专门学习数组呢？看下面「numpy 数组」和「列表」之间的计算效率对比：两个大小都是 1000000，把每个元素翻倍，运行 10 次用 %time 记时。

In [3]:
```
my_arr = np.arange(1000000)
my_list = list(range(1000000))
```

In [11]:
```
%time for _ in range(10): my_arr2 = my_arr * 2
```

```
Wall time: 31.9 ms
```

In [12]:
```
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

```
Wall time: 1.01 s
```

创建数组对象

1．数组属性：ndarray（数组）是存储单一数据类型的多维数组。如果不匹配，往上转换

属性	说明
ndim	返回 int。表示数组的维数
shape	返回 tuple。表示数组的尺寸，对于 n 行 m 列的矩阵，形状为(n,m)
size	返回 int。表示数组的元素总数，等于数组形状的乘积
dtype	返回 data-type。描述数组中元素的类型
itemsize	返回 int。表示数组的每个元素的大小（以字节为单位）。

创建数组对象

2．数组创建

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

参数名称	说明
object	接收array。表示想要创建的数组。无默认。
dtype	接收data-type。表示数组所需的数据类型。如果未给定，则选择保存对象所需的最小类型。默认为None。
ndmin	接收int。指定生成数组应该具有的最小维数。默认为None。

创建数组对象

- 创建数组并查看数组属性

In[1]:	import numpy as np #导入NumPy库
	arr1 = np.array([1, 2, 3, 4]) #创建一维数组
	print('创建的数组为：',arr1)

Out[1]:	创建的数组为： [1 2 3 4]

In[2]:	arr2 = np.array([[1, 2, 3, 4],[4, 5, 6, 7], [7, 8, 9, 10]]) #创建二维数组
	print('创建的数组为：\n',arr2)

Out[2]:	创建的数组为: [[1 2 3 4] [4 5 6 7] [7 8 9 10]]
In[3]:	print('数组维度为：',arr2.shape) #查看数组结构
Out[3]:	数组维度为： (3, 4)

In[4]:	print('数组维度为：',arr2.dtype) #查看数组类型
Out[4]:	数组维度为： int32
In[5]:	print('数组元素个数为：',arr2.size) #查看数组元素个数
Out[5]:	数组元素个数为： 12
In[6]:	print('数组每个元素大小为：',arr2.itemsize) #查看数组每个元素大小
Out[6]:	数组每个元素大小为： 4

创建数组对象

- 重新设置数组的 shape 属性

In[7]:	arr2.shape = 4,3 #重新设置 shape
	print('重新设置 shape 后的 arr2 为：',arr2)
	arr3=arr2.reshape(3,4)

Out[7]:	重新设置shape维度后的arr2为： [[1 2 3] [4 4 5] [6 7 7] [8 9 10]] arr3.resize((4,3))

- 使用 arange 函数创建数组　　等差数组

In[8]:	print('使用 arange 函数创建的数组为：\n',np.arange(0,1,0.1))
Out[8]:	使用arange函数创建的数组为： [0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]

创建数组对象

- 使用 linspace 函数创建数组 创建均匀分布数组

In[9]:	print('使用 linspace 函数创建的数组为：',np.linspace(0, 1, 12))
Out[9]:	使用linspace函数创建的数组为： [0. 0.09090909 … 1.]

- 使用 logspace 函数创建等比数列

In[10]:	print('使用logspace函数创建的数组为：',np.logspace(0, 2, 20))# 10^0-10^2
Out[10]:	使用logspace函数创建的数组为： [1. 1.27427499 1.62377674 ..., 61.58482111 78.47599704 100.]

创建数组对象

- 使用zeros函数创建数组

In[11]:	print('使用zeros函数创建的数组为：',np.zeros((2,3)))
Out[11]:	使用zeros函数创建的数组为： [[0. 0. 0.] [0. 0. 0.]]

```
import numpy as np
import matplotlib.pyplot as plt
nd5 = np.zeros(shape=(46,76,3))
#np.full(shape(100,100,3),fill_value=1.0)
#np.full((100,100,3),1.0)
print(nd5.dtype)
plt.imshow(nd5)
```

- 使用eye函数创建数组 单位矩阵

In[12]:	print('使用eye函数创建的数组为：',np.eye(3))
Out[12]:	使用eye函数创建的数组为： [[1. 0. 0.] [0. 1. 0.] [0. 0. 1.]]

创建数组对象

- 使用diag函数创建数组 对角矩阵

```
In[13]:  print('使用diag函数创建的数组为：',np.diag([1,2,3,4]))
```

```
Out[13]: 使用diag函数创建的数组为:
         [[1 0 0 0]
          [0 2 0 0]
          [0 0 3 0]
          [0 0 0 4]]
```

- 使用ones函数创建数组

```
In[14]:  print('使用ones函数创建的数组为：',np.ones((5,3)))
```

```
Out[14]: 使用ones函数创建的数组为:
         [[ 1.  1.  1.]
          [ 1.  1.  1.]
          [ 1.  1.  1.]
          [ 1.  1.  1.]
          [ 1.  1.  1.]]
```

创建数组对象

3．数组数据类型

NumPy基本数据类型与其取值范围（只展示一部分），基于C开发 布尔、整形、浮点、复数

np.zeros(10,dtype='int16')　np.zeros(10,dtype=np.int32)

类型	描述
bool	用一位存储的布尔类型（值为TRUE或FALSE）
int	由所在平台决定其精度的整数（一般为int32或int64）
int8	整数，范围为−128至127
int16	整数，范围为−32768至32767
int32	……

生成随机数

- 无约束条件下生成随机数 np.random.seed(0) 随机种子

```
In[25]:    print('生成的随机数组为：',np.random.random(100))
```

```
Out[25]: 生成的随机数组为： [ 0.15343184 0.51581585
0.07228451 ... 0.24418316
0.92510545 0.57507965]
```

- 生成服从均匀分布的二维随机数

```
In[26]:    print('生成的随机数组为：\n',np.random.rand(10,5))
```

```
Out[26]: 生成的随机数组为：
[[ 0.39830491 0.94011394 0.59974923 0.44453894
0.65451838]
...
[ 0.1468544 0.82972989 0.58011115 0.45157667
0.32422895]]
```

生成随机数

- 生成服从正态分布的随机数 randn标准正态分组 normal 普通正态分布

```
In[27]:  print('生成的随机数组为：\n',np.random.randn(10,5))#np.random.normal(0,1,(10,5))
```

```
Out[27]: 生成的随机数组为：
 [[-0.60571968  0.39034908 -1.63315513  0.02783885 -1.84139301]
...,
 [-0.27500487  1.41711262  0.6635967   0.35486644 -0.26700703]]
```

- 生成给定上下范围的随机数，如创建一个最小值不低于 2、最大值不高于 10 的 2 行 5 列数组

```
In[28]:  print('生成的随机数组为：',np.random.randint(2,10,size = [2,5]))  #  (2,10,(2,5))
```

```
Out[28]: 生成的随机数组为： [[6 6 6 6 8]
[9 6 6 8 4]]
```

生成随机数

random模块常用随机数生成函数

函数	说明
seed	确定随机数生成器的种子。
permutation	返回一个序列的随机排列或返回一个随机排列的范围。
shuffle	对一个序列进行随机排序。
binomial	产生二项分布的随机数。
normal	产生正态（高斯）分布的随机数。
beta	产生beta分布的随机数。
chisquare	产生卡方分布的随机数。
gamma	产生gamma分布的随机数。
uniform	产生在[0,1)中均匀分布的随机数。

通过索引访问数组

1．一维数组的索引

In[29]:	arr = np.arange(10) print('索引结果为：',arr[5]) #用整数作为下标可以获取数组中的某个元素
Out[29]:	索引结果为： 5
In[30]:	print('索引结果为：',arr[3:5]) #用范围作为下标获取数组的一个切片，包括arr[3]不包括arr[5]
Out[30]:	索引结果为： [3 4]
In[31]:	print('索引结果为：',arr[:5]) #省略开始下标，表示从arr[0]开始
Out[31]:	索引结果为： [0 1 2 3 4]
In[32]:	print('索引结果为：',arr[-1]) #下标可以使用负数，-1表示从数组后往前数的第一个元素
Out[32]:	索引结果为： 9

通过索引访问数组

1．一维数组的索引

续表

In[33]:	arr[2:4] = 100,101 print('索引结果为：',arr) #下标还可以用来修改元素的值
Out[33]:	索引结果为： [0 1 100 101 4 5 6 7 8 9]

In[34]:	#范围中的第三个参数表示步长，2表示隔一个元素取一个元素
	print('索引结果为：',arr[1:-1:2])

Out[34]:	索引结果为： [1 101 5 7]
In[35]:	print('索引结果为：',arr[5:1:-2]) #步长为负数时，开始下标必须大于结束下标
Out[35]:	索引结果为： [5 101]

通过索引访问数组

2．多维数组的索引

In[36]:
```
arr = np.array([[1, 2, 3, 4, 5],[4, 5, 6, 7, 8], [7, 8, 9, 10, 11]])
print('创建的二维数组为：',arr)
```

Out[36]:
```
创建的二维数组为： [[ 1  2  3  4  5]
 [ 4  5  6  7  8]
 [ 7  8  9 10 11]]
```

In[37]:
```
print('索引结果为：',arr[0,3:5])  #索引第0行中第3和4列的元素
```

Out[37]:
```
索引结果为： [4 5]
```

通过索引访问数组

2．多维数组的索引

续表

In[38]:	#索引第2和3行中第3～5列的元素
	print('索引结果为：',arr[1:,2:])

Out[38]:	索引结果为： [[6 7 8] [9 10 11]]
In[39]:	print('索引结果为：',arr[:,2]) #索引第3列的元素
Out[39]:	索引结果为： [3 6 9]

通过索引访问数组

2．多维数组的索引（使用整数和布尔值索引访问数据）

In[40]:	#从两个序列的对应位置取出两个整数来组成下标：arr[0,1], arr[1,2], arr[2,3]
	print('索引结果为：',arr[(0,1,2),(1,2,3)])

Out[40]:	索引结果为： [2 6 10]
In[41]:	print('索引结果为：',arr[1:,(0,2,3)]) #索引第2、3行中第0、2、3列的元素
Out[41]:	索引结果为： [[4 6 7] [7 9 10]]

In[42]:	mask = np.array([1,0,1],dtype = np.bool)
	#mask是一个布尔数组，它索引第0、2行中第2列的元素
	print('索引结果为：',arr[mask,2])

Out[42]:	索引结果为： [3 9]

变换数组的形态

- 改变数组形状

In[43]:	arr = np.arange(12) #创建一维数组
	print('创建的一维数组为：',arr)

Out[43]:	创建的一维数组为： [0 1 2 3 4 5 6 7 8 9 10 11]
In[44]:	print('新的一维数组为：',arr.reshape(3,4)) #设置数组的形状
Out[44]:	新的一维数组为： [[0 1 2 3] [4 5 6 7] [8 9 10 11]]
In[45]:	print('数组维度为：',arr.reshape(3,4).ndim) #查看数组维度
Out[45]:	数组维度为： 2

变换数组的形态

- 使用ravel函数展平数组（展开后可以直接修改原数组）

In[46]:	arr = np.arange(12).reshape(3,4) print('创建的二维数组为：',arr)
Out[46]:	创建的二维数组为： [[0 1 2 3] [4 5 6 7] [8 9 10 11]]
In[47]:	print('数组展平后为：',arr.ravel())
Out[47]:	数组展平后为： [0 1 2 3 4 5 6 7 8 9 10 11]

变换数组的形态

- 使用flatten函数展平数组（返回一个原数组copy）

In[48]:	print('数组展平为：',arr.flatten()) #横向展平
Out[48]:	数组展平为： [0 1 2 3 4 5 6 7 8 9 10 11]
In[49]:	print('数组展平为：',arr.flatten('F')) #纵向展平
Out[49]:	数组展平为： [0 4 8 1 5 9 2 6 10 3 7 11]

变换数组的形态

组合数组-数组的拼接与分裂

- **使用hstack函数实现数组横向组合：np.hstack((arr1,arr2))**
- **使用vstack函数实现数组纵向组合：np.vstack((arr1,arr2))**
- **使用concatenate函数实现数组横向组合：np.concatenate((arr1,arr2),axis = 1))**
- **使用concatenate函数实现数组纵向组合：np.concatenate((arr1,arr2),axis = 0))**
- **x = np.array([1, 2, 3])**
- **y = np.array([3, 2, 1])**
- **np.hstack((x,y))　　np.concatenate([x, y]) array([1, 2, 3, 3, 2, 1])**
- **np.vstack((x,y))**
- **array([[1, 2, 3],**
- **　　[3, 2, 1]])**

变换数组的形态

切割数组

- **使用hsplit函数实现数组横向分割： np.hsplit(arr, 2)**
- **使用vsplit函数实现数组纵向分割： np.vsplit(arr, 2)**
- **使用split函数实现数组横向分割： np.split(arr, 2, axis=1)**
- **使用split函数实现数组纵向分割： np.split(arr, 2, axis=0)**

目录

创建NumPy矩阵

创建与组合矩阵

- 使用mat函数创建矩阵： matr1 = np.mat("1 2 3;4 5 6;7 8 9")
- 使用matrix函数创建矩阵：matr2 = np.matrix([[123],[456],[789]])
- 使用bmat函数合成矩阵：np.bmat("arr1 arr2; arr1 arr2")

创建NumPy矩阵

矩阵的运算

- 矩阵与数相乘：matr1*3
- 矩阵相加减：matr1±matr2
- 矩阵相乘：matr1*matr2
- 矩阵对应元素相乘：np.multiply(matr1,matr2)
- 矩阵特有属性：

属性	说明
T	返回自身的转置
H	返回自身的共轭转置
I	返回自身的逆矩阵
A	返回自身数据的2维数组的一个视图

认识ufunc函数

全称通用函数（universal function），是一种能够对数组中所有元素进行操作的函数。比循环获得更高的效率

- 四则运算：加（+）、减（-）、乘（*）、除（/）、幂（**）三角、指数对数。数组间的四则运算表示对**每个数组中的元素分别进行四则运算**，所以形状必须相同。np.add(x+x) np.sin() np.exp() np.log()
- x=np.range(5) array([0, 1, 2, 3, 4]) x+1 array([1, 2, 3, 4, 5]) x+x array([0, 2, 4, 6, 8])
- 比较运算：>、<、==、>=、<=、!=。比较运算返回的结果是一个布尔数组，每个元素为每个数组对应元素的比较结果。
- x>2 array([False, False, False, True, True]) x[x>2] array([3, 4])
- 逻辑运算：np.any函数表示逻辑“or”，np.all函数表示逻辑“and”。运算结果返回布尔值。
- np.any(x>2) True

常用的统计函数（聚合 最大 最小 其他）

当axis=0时，表示沿着纵轴计算。当axis=1时，表示沿着横轴计算。默认时计算一个总值。

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

函数	说明
sum	计算数组的和　sum(a) 内置函数　np.sum(a)==a.sum()　numpy函数速度快，默认求全部元素聚合
mean	计算数组均值　a.mean()　2.5　a.mean(axis=0) 延0轴聚合，返回array([2., 3.])
std	计算数组标准差
var	计算数组方差
min	计算数组最小值
max	计算数组最大值
argmin	返回数组最小元素的索引
argmax	返回数组最小元素的索引
cumsum	计算所有元素的累计和
cumprod	计算所有元素的累计积

美国总统身高

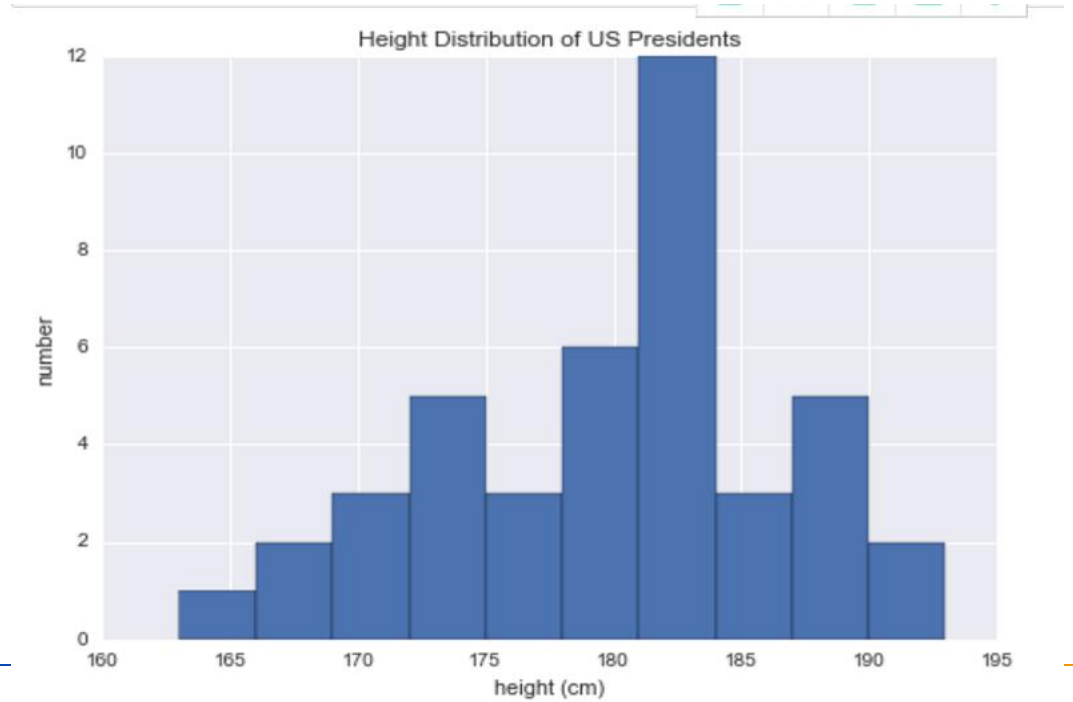
```
import numpy as np
mat =np.loadtxt("./data/president_heights.csv",dtype=np.str,delimiter=",",skiprows=1)
heights=mat[:,2].astype(np.int32)
print(heights)
```

```
order,name,height(cm)
1,George Washington,189
2,John Adams,170
3,Thomas Jefferson,189
```

```
print("Mean height:       ", heights.mean())
print("Standard deviation:", heights.std())
print("Minimum height:    ", heights.min())
print("Maximum height:    ", heights.max())
```

```
Mean height:       179.738095238
Standard deviation: 6.93184344275
Minimum height:    163
Maximum height:    193
```

```
import matplotlib.pyplot as plt
import seaborn; seaborn.set()  # set plot style
plt.hist(heights)
plt.title('Height Distribution of US Presidents')
plt.xlabel('height (cm)')
plt.ylabel('number');
```

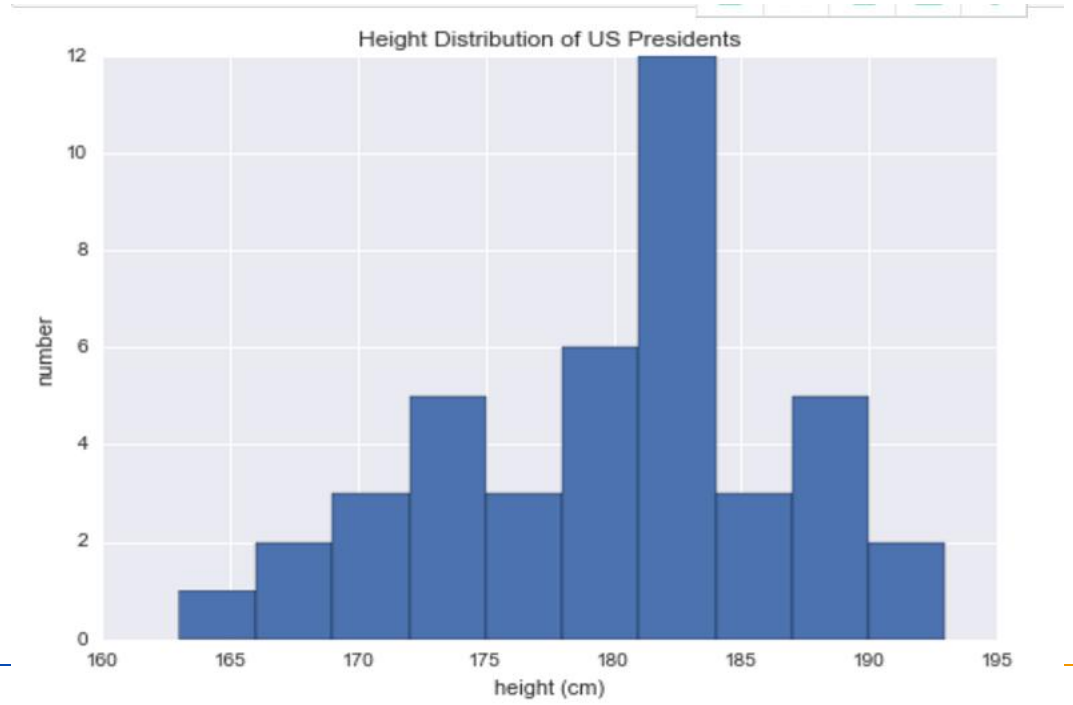


美国总统身高

```
import numpy as np
import pandas as pd
data = pd.read_csv('president_heights.csv')
heights = np.array(data['height(cm)'])
print(heights)

print("Mean height:       ", heights.mean())
print("Standard deviation:", heights.std())
print("Minimum height:    ", heights.min())
print("Maximum height:    ", heights.max())

import matplotlib.pyplot as plt
import seaborn; seaborn.set()  # set plot style
plt.hist(heights)
plt.title('Height Distribution of US Presidents')
plt.xlabel('height (cm)')
plt.ylabel('number');
```



认识ufunc函数

ufunc函数的广播机制

广播（broadcasting）是指不同形状的数组之间执行算术运算的方式。需要遵循4个原则。

- 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。
- 输出数组的shape是输入数组shape的各个轴上的最大值。
- 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。
- 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。

认识ufunc函数

数组计算的广播机制：允许两个不同大小的数组进行计算，单个值即为0维数组

- 一维数组的广播机制

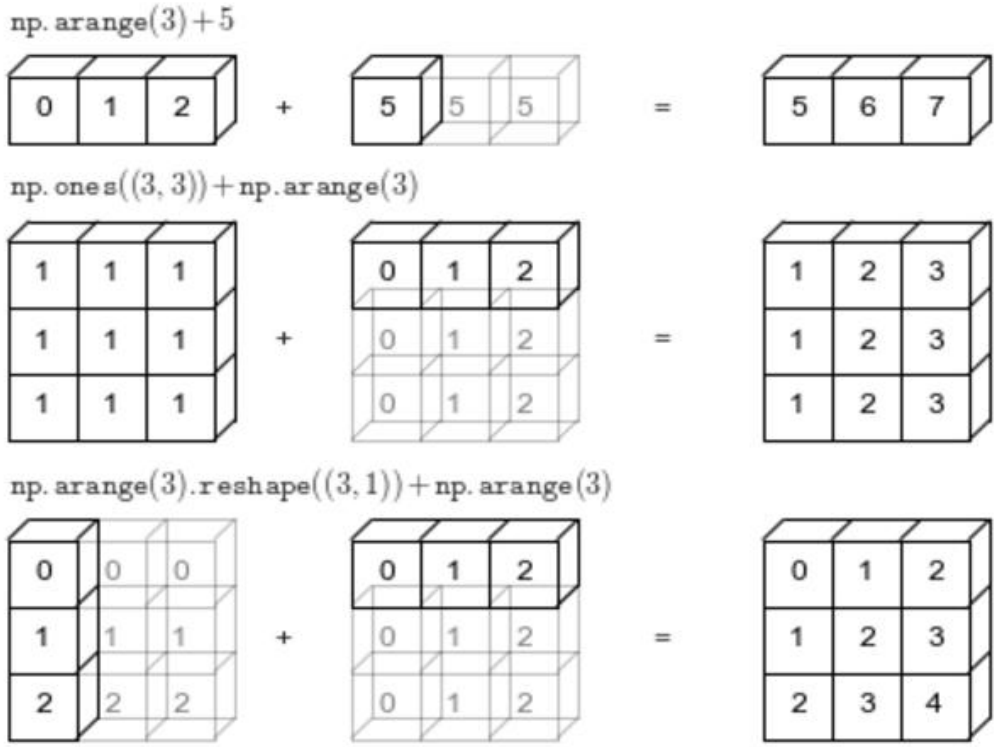
$$\begin{bmatrix}0&0&0\\1&1&1\\2&2&2\\3&3&3\end{bmatrix}+[1\ 2\ 3]\rightarrow\begin{bmatrix}0&0&0\\1&1&1\\2&2&2\\3&3&3\end{bmatrix}+\begin{bmatrix}1&2&3\\1&2&3\\1&2&3\\1&2&3\end{bmatrix}=\begin{bmatrix}1&2&3\\2&3&4\\3&4&5\\4&5&6\end{bmatrix}$$

- 二维数组的广播机制

$$\begin{bmatrix}0&0&0\\1&1&1\\2&2&2\\3&3&3\end{bmatrix}+\begin{bmatrix}1\\2\\3\\4\end{bmatrix}\rightarrow\begin{bmatrix}0&0&0\\1&1&1\\2&2&2\\3&3&3\end{bmatrix}+\begin{bmatrix}1&1&1\\2&2&2\\3&3&3\\4&4&4\end{bmatrix}=\begin{bmatrix}1&1&1\\3&3&3\\5&5&5\\7&7&7\end{bmatrix}$$

认识ufunc函数

ufunc函数的广播机制




```
np.random.seed(0)
x=np.random.randint(1,10,5)
x
Out[67]: array([6, 1, 4, 4, 8])
x.reshape(5,1)-x.reshape(1,5)

Out[70]:
array([[ 0,  5,  2,  2, -2],
       [-5,  0, -3, -3, -7],
       [-2,  3,  0,  0, -4],
       [-2,  3,  0,  0, -4],
       [ 2,  7,  4,  4,  0]])  可以计算任意两数之间的差值
```

请大家思考如果是位置坐标点
x=np.random.randint(1,10,(5,2))
如何求任意两点之间的距离？？？

目录

花哨索引

数组与元组、列表一样可以通过值与切片进行索引 arr[0] arr[:4]
可以通过索引数组来一次性获得多个元素。

```
import numpy as np
x = np.random.randint(100, size=10)
print(x)
```

```
[51 92 14 71 60 20 82 86 74 74]
```

```
[x[3], x[7], x[2]]
```

```
[71, 86, 14]
```

```
ind = [3, 7, 4]
x[ind]
```

```
array([71, 86, 60])
```

```
ind = np.array([[3, 7],
                [4, 5]])
x[ind]  结果与索引数组形状一致
```

```
array([[71, 86],
       [60, 20]])
```

```
X = np.arange(12).reshape((3, 4))
X
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
row = np.array([0, 1, 2])
col = np.array([2, 1, 3])
X[row, col]
```

```
array([ 2,  5, 11])
```

组合索引

花哨索引可以和其它索引结合

print(X)	[[0 1 2 3] [4 5 6 7] [8 9 10 11]]
X[2, [2, 0, 1]]	array([10，8，9])
X[1:, [2, 0, 1]]	array([[6，4，5], [10，8，9]])
假定有100对数据，数组x维度（100，2） 从中随机选取10对。 indices=np.random.choice(x.shape[0],10) x[indices]	返回10个0轴的随机索引位置，可以对比Pandas
a = np.array([1, 2, 3, 4]) np.add.at(a, [0, 1], 1)	a变为array([2, 3, 3, 4])

使用数组进行简单统计分析

Python有内置的sort和sorted函数可以对列表排序。np.sort快速排序效率更高

- **sort函数是最常用的排序方法。 arr.sort()**
- **sort函数也可以指定一个axis参数，使得sort函数可以沿着指定轴对数据集进行排序。axis=1为沿横轴排序；axis=0为沿纵轴排序**
- **argsort函数返回值为重新排序值的下标。 arr.argsort() argmax() 返回最大值的下标**

x = np.array([2, 1, 4, 3, 5]) np.sort(x) -np.sort(-x)#降序	array([1, 2, 3, 4, 5])
x.sort() print(x)	[1 2 3 4 5] 直接替换原始数组
x = np.array([2, 1, 4, 3, 5]) i = np.argsort(x) print(i)	[1 0 3 2 4]
x[i]	array([1, 2, 3, 4, 5]) 花哨索引
X=np.random.randint(0,10,(4,6)) np.sort(X,axis=0)	延纵轴排序

array([[2, 1, 4, 0, 1, 5],
[5, 2, 5, 4, 3, 7],
[6, 3, 7, 4, 6, 7],
[7, 6, 7, 4, 9, 9]])

任务实现

读取西雅图2014年日降水量信息表

```
rainfall=np.loadtxt("Seattle2014.csv",skiprows=1,delimiter=',', usecols=3,dtype=int)
```

要求参考身高数据，绘制直方图

统计：2014年日平均降水量

有降水的天数，降水超过10mm的天数

```
rainfall[rainfall>0].size
np.count_nonzero(rainfall>0)
np.sum(rainfall>0)
```

支持& | ~ 等逐位逻辑运算

```
 bins=np.linspace(np.min(rainfall),np.max(rainfall),40)
counts=np.zeros_like(bins) #产生统计
i=np.searchsorted(bins,rainfall)  #找到对应区间
 np.add.at(counts,i,1)#统计数量
plt.plot(bins,counts)
```

Thank you!