多维数组创建与访问

DATA

——多维数据结构与运算

主讲教师：王洪亚

数据和数据维度

- 在数据分析应用中经常需要从多个维度来表示数据
  - 一维数据、二维数据
- **维度（dimension）**称为**轴（axis）**
  - **0维数据（标量）**，一个数字，12，34.8
  - **1维数据（向量）**，一个班级同学的姓名，name[5]

多维数据

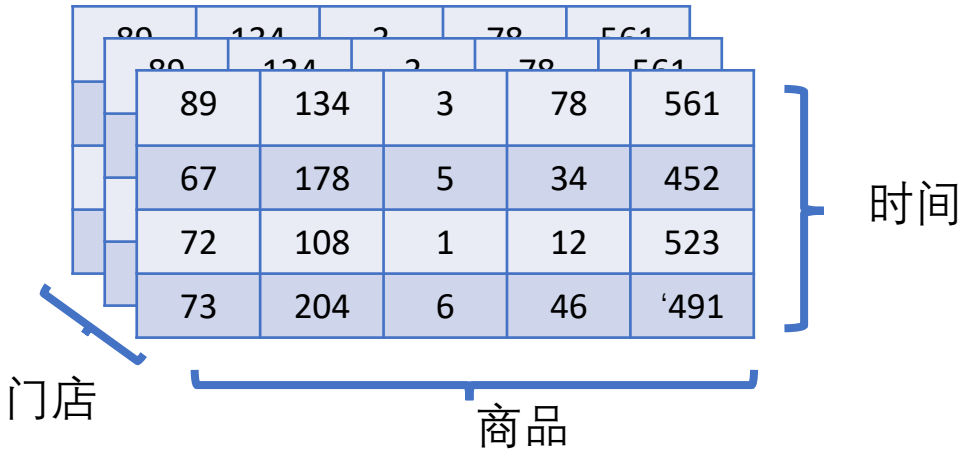
- **2维数据(矩阵),** 数据集最常见的形式
  - 人口统计数据集
    - 第0轴为样本
    - 第1轴为特征

姓名	出生年	常住城市	年收入	职业
'李利茗'	1980	'苏州'	30.5	'医生'
'张广量'	1978	'广州'	25.6	'教师'
'赵祁姗'	1989	'北京'	15.8	'公务员'
'吴孟'	1992	'上海'	18.6	'工程师'

样本

样本特征

- **3维数据**， 常用于存储时间或序列数据
  - 零售店每天各商品销量数据集
    - 第0轴: 商品
    - 第1轴: 时间
    - 第2轴: 门店



Numpy库

- NumPy库是Python进行科学计算和数据分析的基础库
  - 支持丰富数据表示方式
  - Anaconda环境中已安装

```
>>> import numpy as np
```

- ndarray: NumPy核心
  - N-dimensional array, N维数组
  - 相同数据类型的元素组成的多维数组
    - 数组大小需事先指定

案例2-1：学生课程考试成绩

- 5位同学参加了学业水平考试，考试科目共7门

姓名	Math	English	Python	Chinese	Art	Database	Physics
王微	70	85	77	90	82	84	89
肖良英	60	64	80	75	80	92	90
方绮雯	90	93	88	87	86	90	91
刘旭阳	80	82	91	88	83	86	80
钱易铭	88	72	78	90	91	73	80

课程名称
一维数据

学生姓名
一维数据

学生成绩
二维数据

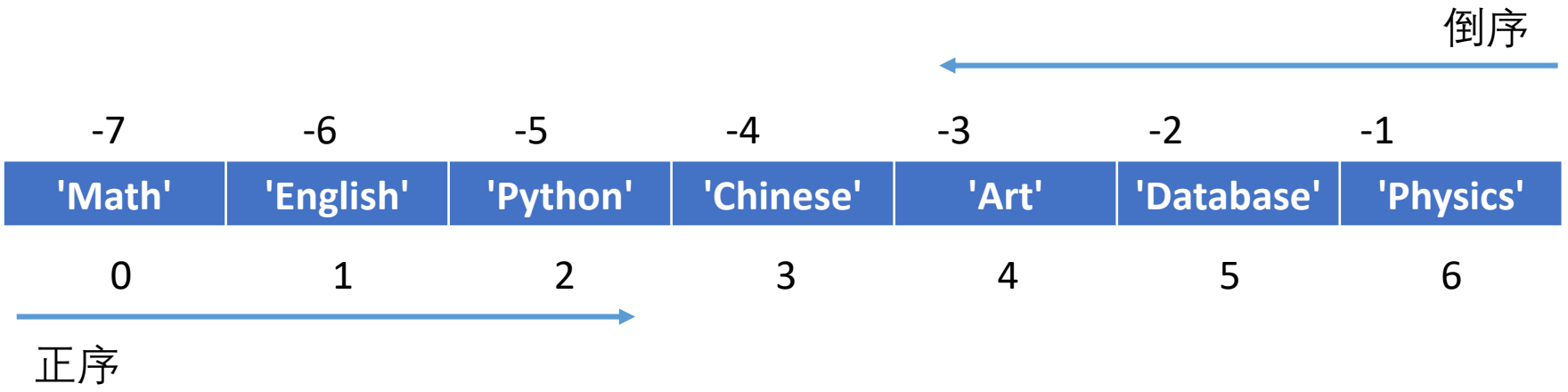
一维数组对象

- 创建一维数组分别保存学生姓名和考试科目，访问数组元素
  - np.array( )，基于列表创建一维数组

```
>>> names = np.array(['王微', '肖良英', '方绮雯', '刘旭阳','钱易铭'])
>>> names
array(['王微', '肖良英', '方绮雯', '刘旭阳','钱易铭',  dtype='<U3')
>>>   subjects   =   np.array(['Math',   'English',   'Python',
'Chinese','Art', 'Database', 'Physics'])
>>> subjects
array(['Math', 'English', 'Python', 'Chinese', 'Art', 'Database',
'Physics'],  dtype='<U8')
```

数组元素的索引

- 索引序号范围为[0, n-1]或[-*n*, -1]
  - *n*为一维数组的大小
  - 与Python序列的索引方式一致



数组元素访问

- 统计和建模分析过程中，需要对部分数据进行计算
  - 编程语言处理数组元素
    - for、while 循环

```
sub1 = []
for i in [0,2,4] :
  sub1 = sub1.append(subjects[i])
```

数组切片操作

- 使用索引列表

```
>>> subjects[ [0,2,4] ]     #两层括号，[0,2,4]为索引列表
array(['Math', 'Python', 'Art'],  dtype='<U8')
```

外层括号表示元素访问
内层括号表示列表

- 索引也可表示为：start:end:step
  - 生成 start ~ end-1，步长step的等差数列

```
>>> names[ 1:4 ]   #抽取索引为1、2、3的元素
array(['肖良英' '方绮雯' '刘旭阳'],  dtype='<U3')
>>> subjects[ : -1:2]   #抽取索引为0、2、4的元素
array(['Math', 'Python', 'Art'],  dtype='<U8')
```

start省略：从0开始
end省略：到最后
step省略：步长为1

根据条件筛选数组元素

- 使用条件表达式和关系运算符
  - 筛选出names数组中值等于“王微”或“钱易铭”的元素

```
#循环查找
name1 = []
for i in range[0,7] :
    if (names == '王微') | (names== '钱易铭'):
        sub1 = sub1.append(names[i])
```

- 数组操作

```
>>> names[ (names == '王微') | (names== '钱易铭')]
array(['王微', '钱易铭'],  dtype='<U3')
```

二维数组对象

- 创建二维数组scores，记录 “names”中同学 “subjects”的各门课程考试成绩

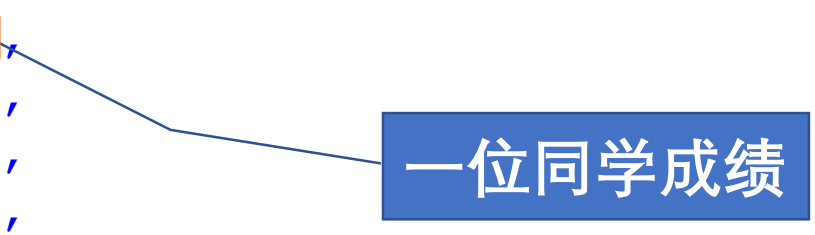
```
>>> scores= np.array([[70,85,77,90,82,84,89],
[60,64,80,75,80,92,90], [90,93,88,87,86,90,91],
[80,82,91,88,83,86,80],[88,72,78,90,91,73,80]])
>>> scores
array([[70, 85, 77, 90, 82, 84, 89],
       [60, 64, 80, 75, 80, 92, 90],
       [90, 93, 88, 87, 86, 90, 91],
       [80, 82, 91, 88, 83, 86, 80],
       [88, 72, 78, 90, 91, 73, 80]])
```

一行数据

二维数组对象特性

- 使用二维数组scores，记录“names”中同学“subjects”的各门课程考试成绩

```
>>> scores
array([[70, 85, 77, 90, 82, 84, 89],
       [60, 64, 80, 75, 80, 92, 90],
       [90, 93, 88, 87, 86, 90, 91],
       [80, 82, 91, 88, 83, 86, 80],
       [88, 72, 78, 90, 91, 73, 80]])
```



- 数组属性

```
>>> scores.ndim   #数组维数
2
>>> scores.size   #数组元素总数，行×列
35
>>> scores.shape   #数组的行数和列数
(5, 7)
>>> scores.dtype   #数组元素的类型
dtype('int32')
```

- 二维数组切片操作的基本格式：**arr[row , column]**
  - row行序号，column列序号，中间用‘，’隔开
  - 行、列切片的表示方式与一维数组相同
  - 用“:”代替行或列，表示选中对应的所有行或列

二维数组切片

- 访问指定行、列的元素，给出行和列两个索引值

```
>>> scores[1,0]
60
>>> scores[[1,3],[0,1]]
array([60, 82])
```

抽取scores[1,0],scores[3,1]

```
>>> scores[[1,3]]
array([[60, 64, 80, 75, 80, 92, 90],
       [80, 82, 91, 88, 83, 86, 80]])
```

取某行所有列值，可以省略列：

抽取部分列，行不能省略

```
>>> scores[:, [0,1]]
array([[70, 85],
       [60, 64],
       [90, 93],
       [80, 82],
       [88, 72]])
```

```
array([[70, 85, 77, 90, 82, 84, 89],
       [60, 64, 80, 75, 80, 92, 90],
       [90, 93, 88, 87, 86, 90, 91],
       [80, 82, 91, 88, 83, 86, 80],
       [88, 72, 78, 90, 91, 73, 80]])
```

条件筛选

- 条件表达式：生成布尔型的数组

```
>>> mask = (names == '王微') | (names== '钱易铭')
>>> mask
array([ True, False, False, False, True ], dtype=bool)
```

- 使用布尔型数组筛选访问**其他数组**的元素

```
>>> scores[mask, :]
array([[70, 85, 77, 90, 82, 84, 89],
       [88, 72, 78, 90, 91, 73, 80]])
```

scores

```
array([[70, 85, 77, 90, 82, 84, 89],     True
       [60, 64, 80, 75, 80, 92, 90],     Flase
       [90, 93, 88, 87, 86, 90, 91],     False
       [80, 82, 91, 88, 83, 86, 80],     False
       [88, 72, 78, 90, 91, 73, 80]])    True
```