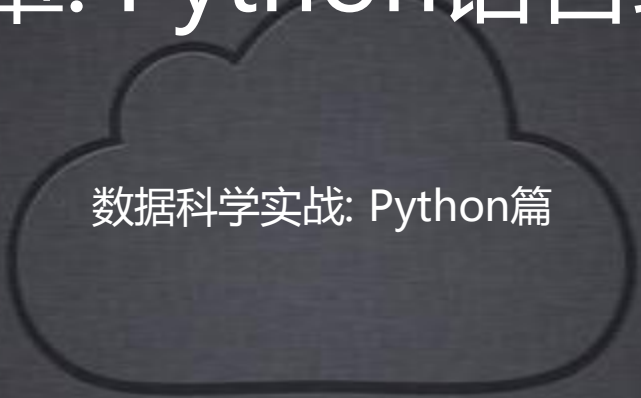


第3章: Python语言编程



数据科学实战: Python篇

讲师 : Ben

自我介绍

- 天善商业智能和大数据社区 讲师 –Ben
- 天善社区 ID - Ben_Chang
- <https://www.hellobi.com> – 学习过程中有任何相关的问题都可以提到技术社区数据挖掘版块。

- Python基础数据类型与表达式
- Python原生态数据结构
- Python控制流
- Python函数
- Python模块
- 使用pandas读写数据

1 Python的基本数据类型与表达式

基本数据类型

- Python中6种基本的数据类型:
 - string-字符串
 - number-数值类型，包括整数（ Integer ）与浮点数（ float ）
 - bool(True/False)-布尔值
 - time/datetime-日期时间
 - None-空值
 - complex-复数
- 类型识别及转换
 - type()
 - 类型转换

Python字符串简介

- 在Python中用引号引起来的字符集称之为字符串，比如： 'hello' 、 "my Python" 、 "2+3" 等都是字符串。本课程中需要掌握的内容包括：
 - Python中字符串中使用的引号可以是单引号、双引号跟三引号
 - 特殊字符(换行符、制表符等)可以通过转义字符 '\ ' 进行表示，原样输出引号内字符串可以使用在引号前加r 的形式来表示-r' '
 - 字符串可以进行运算，其中子字符串的运算经常被使用：子字符串是指字符串的子集
 - 采用格式化输出可以方便地将字符串打印到屏幕上

数值类型：整数与浮点数

- Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样
 - int (有符号整数): 通常被简称为整数，是正或负整数，不带小数点。
 - float (浮点实数值): 或浮点数，表示实数，并写入一个小数点分隔的整数部分和小数部分。浮点数也可以是科学记数法，无穷大inf也为浮点数。
- Python3中已经不再对整数与浮点数的运算结果做区分，在混合计算时，python会把整数转换为浮点数
- 数值的除法（/）总是返回一个浮点数，要获取整数使用//操作符

布尔值

- 布尔值用于逻辑判断，只有两个可能结果:True/False
- 布尔值可以看做是特殊的数值类型：True=1，False=0（整型、浮点型的“0”和复数0+0j也可以表示False，其余整型、浮点型、复数数值都被判断为True）
- bool值往往是比较运算的结果，其运算包括and/or/not
 - And：与运算，只有所有都为True运算结果才是True。
 - or：或运算，只要其中有一个为True运算结果就是True
 - Not：非运算，它是一个单目运算符，把True变成False，False变成True

日期时间

- 日期时间可以看做是代格式的数值类型
- Python 程序能用很多方式处理日期和时间，转换日期格式是一个常见的功能。
- Python 提供了一个 time 和 datetime 模块可以用于格式化日期和时间。
 - 时间间隔是以秒为单位的浮点小数
 - 时间戳都以1970年1月1日午夜（历元）经过了多长时间来表示
 - 常用日期格式化符号包括：

%y 两位数的年份表示（00-99）	%H 24小时制小时数（0-23）
%Y 四位数的年份表示（000-9999）	%I 12小时制小时数（01-12）
%m 月份（01-12）	%M 分钟数（00=59）
%d 月内中的一天（0-31）	%S 秒（00-59）

- None代表空值，类型为NoneType，支持的运算较少，也没有任何内建方法。
- None和任何其他的数据类型比较永远返回False，not None返回True。
- 要注意空值（None）与pandas中的缺失值（NaN）不是一种类型，在使用pandas进行数据处理当中，使用NaN表示数值缺失。

类型判断与类型转换

- 类型判断：使用type()可判断对象类型，函数、类、各种数据类型的变量在python中都是对象，因此都可以使用type()进行判断
- 类型转换：常用的类型转换函数包括int、float、complex、str、bool，使用时需要保证对象是可转换成相应类型的。

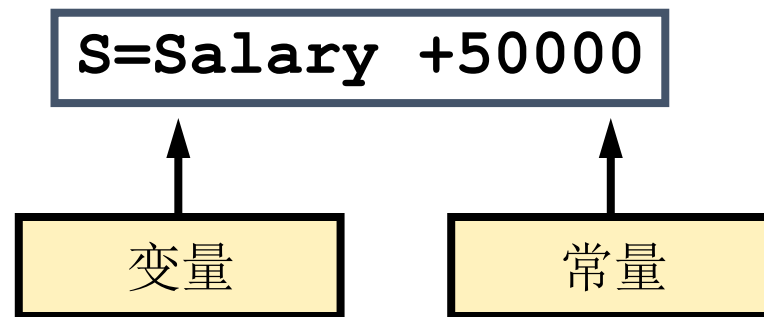
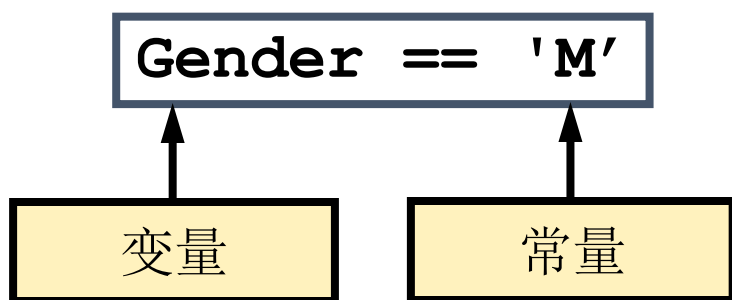
- **表达式**是由一系列运算符和操作数组成的，用于处理数据的计算结构。
 - 运算数包含常量和变量。
 - 数据科学中运算符包含算术、赋值、比较、逻辑、成员。

常量运算数是固定值。

- 字符值必须包含在引号中并且区分大小写。
- 数值不用引号。

一个变量运算数必须是之前定义的一个变量。

例如：



算数运算符

假设变量包含 $a=10$ 以及变量 $b=21$ ，那么

操作符	描述	示例
+ 加法	相加运算两侧的值	$a + b$ ，得到31
- 减法	操作符右侧数减去左侧操作数	$a - b$ ，得到-11
* 乘法	操作符两侧的值相乘	$a * b$ ，得到210
/ 除法	用运算符右侧的操作数除以左侧操作数	b / a ，得到2.1
% 模	用右手操作数除以左手操作数并返回余数	$b \% a$ ，得到1
** 指数	执行运算符指数(幂)计算	$a ** b$ 就是10 的20 次幂
//	取整（地板除） - 除法不管操作数为何种数值类型，总是会舍去小数部分，返回数字序列中比真正的商小的最接近的数字	$9 // 2$ ，得到4 $9.0 // 2.0$ ，得到4.0

赋值运算符

假设变量 $a=10$ 和变量 $b=20$ ，那么

运算符	描述	示例
=	将右侧的操作数赋值给左侧的操作数	$c = a + b$ 是将 $a + b$ 的值分配到 c
+=	相加右操作数和左操作数，并分配结果到左操作数	$c += a$ 相当于 $c = c + a$
-=	左操作数减去右操作数，并分配结果到左操作数	$c -= a$ 相当于 $c = c - a$
*=	右操作数和左操作数相乘，并分配结果到左操作数	$c *= a$ 相当于 $c = c * a$
/=	左操作数除以右操作数，并分配结果到左操作数	$c /= a$ 相当于 $c = c /$
%=	左操作数模除以右操作数，并分配结果到左操作数	$c \% = a$ 相当于 $c = c \% a$
**=	执行运算符指数(幂)计算并将结果分配值给左操作数	$c ** = a$ 相当于 $c = c ** a$
//= 地板除	对操作数进行地板除，并赋值给左操作数	$c //= a$ 相当于 $c = c // a$

比较运算符

这些运算符对它们的两侧的值进行比较，并决定它们之间的关系。它们也被称为关系运算符。
假设变量a=10和变量b=20，那么

操作符	描述	示例
==	如果两个操作数的值相等，则计算结果为 true	(a == b) 其值不为 true.
!=	如果两个操作数的值不相等，则计算结果为 true	(a != b) 其值为 true.
>	如果左操作数的值大于右操作数的值，则计算结果为 true	(a > b) 其值不为true.
<	如果左操作数的值小于右操作数的值，则计算结果为 true	(a < b) 其值为true.
>=	如果左操作数的值大于或等于右操作数的值，则计算结果为 true	(a >= b) 其值不为 true.
<=	如果左操作数的值小于或等于右操作数的值，则计算结果为 true	(a <= b) 其值为 true.

Python语言支持以下逻辑运算符。假设变量a=True和变量b=False，那么：

操作符	描述	示例
and -逻辑与	如果两个操作数为真，则条件为true	(a and b) 结果 False.
or -逻辑或	如果两个操作数为非零，条件变为true	(a or b) 结果 True.
not-逻辑非	用来扭转操作数的逻辑状态	(Not b) 结果 True.

用于测试成员是否在另外一个字符串，列表或元组中：

运算符	描述	示例
in	如果在指定的顺序中找到变量，计算结果为true，否则为 false	x in y, 如果x是序列y的成员，则返回true
not in	如果在指定的顺序中不能找到变量，计算结果为true，否则为 false	x not in y, 如果x不是序列y的成员，则返回true

2 Python原生生态数据结构

- Python常用数据结构包括：
 - 列表(list)
 - 元组/序列(tuple)
 - 集合(set)
 - 字典(dict)

列表(list)

- 列表list是Python内置的一种数据结构。list是一种有序的集合，用来存储一连串元素的容器，列表用[]来表示，其中元素的类型可不相同。列表的操作包括：
 - 可以随时添加、删除、插入、修改其中的元素，合并列表等
 - 用索引选择列表中的元素或子列表
 - 判断元素是否在列表中
 - 对元素计数
 - 判断元素在列表中的位置
 - 利用函数生成所需的列表
 -

元组/序列(tuple)

- 元组是不可变的列表，没有增，删，改的权限
- 元组里面的元素也是进行索引计算。元组的符号是()，可进行的操作包括：
 - 索引
 - 对元素计数，对元组长度计数
 - 计算元素位置
 - 查看元素是否在元组中
 -
- 与列表不同的是，列表里面的元素的值可以修改，而tuple里面的元素的值不能修改（没有append、insert等方法），只能读取。因为tuple不可变，所以代码更安全。

- 集合(set)是一组key的集合，其中key不能重复，集合的格式是：set()
- 要创建一个set，需要提供一个列表、字典或字符串作为输入。
- Python中集合主要有两个功能，一个功能是进行集合操作，另一个功能是消除重复元素，主要操作包括：
 - set可以增加或删除其中的元素(key)
 - set可用于交、差、并运算
 - 消除重复元素
 -

字典(dict)

- Python内置了字典dict，在其他语言中也称为map，使用键-值(key-value)存储，具有极快的查找速度，其格式是用大括号{}括起来key和value用冒号“:”进行对应。
- 字典的操作和列表的操作完全一致，只是把数值索引改为自定义索引。
- 常用操作包括：
 - 返回所有key、返回所有value、根据key查找value等
 - 添加、修改、删除字典里面的项目
 - 判断key是否存在
 -
- 和list比较，dict有以下特点：①查找和插入的速度极快，不会随着key的增加而增加；②需要占用大量的内存，内存浪费多。
- 需要注意：dict的key必须是不可变对象，字符串、整数等可以放心地作为key，而list是可变的，就不能作为key，但可以作为value

列表、元组、集合、字典的互相转换

- Python中列表、元组、集合、字典之间是可以互相转换的
- 其中字典转换成其它类型时需要调用字典的keys和values 方法。
- 可以利用zip方法方便地将列表、元组、集合 “缝合” 起来，再转化为字典。

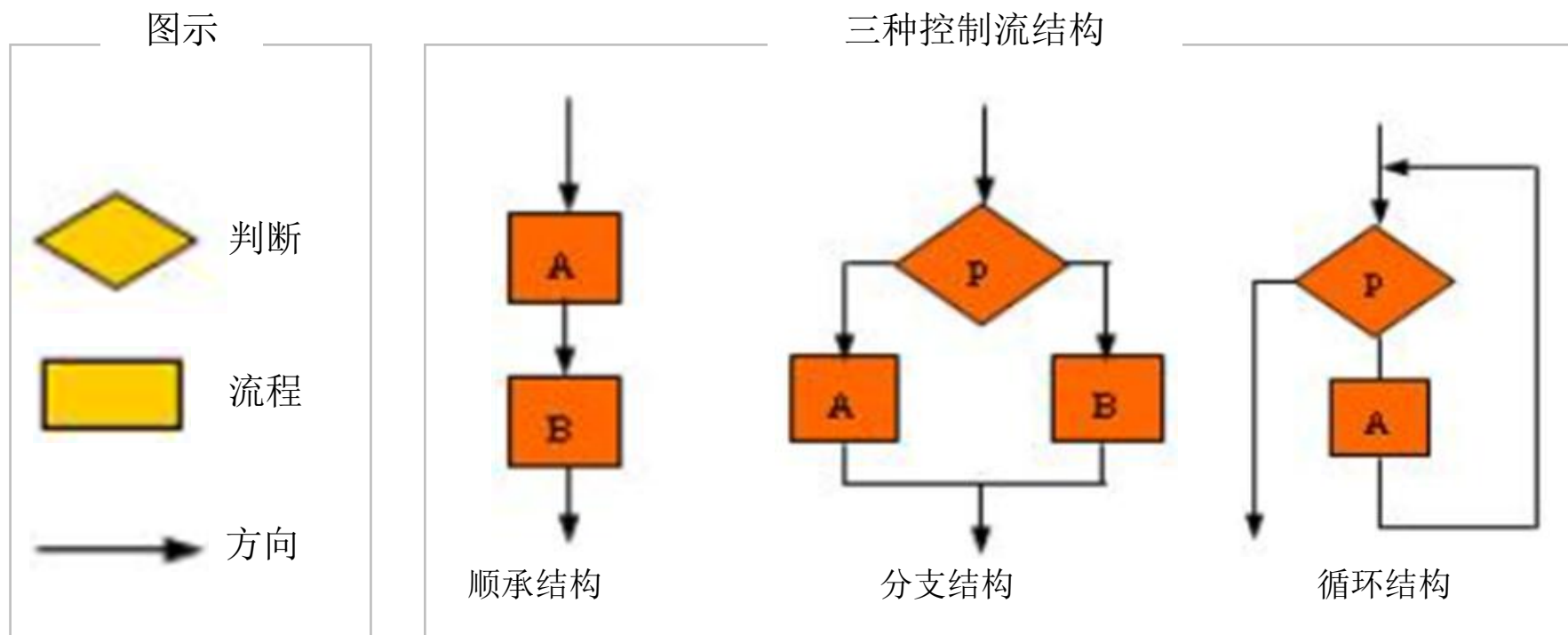
```
z1 = zip(('A', 'B', 'C'), [1, 2, 3, 4]) # zip可以将列表、元组、集  
print z1  
print dict(z1)
```

```
[('A', 1), ('B', 2), ('C', 3)]  
{'A': 1, 'C': 3, 'B': 2}
```

3 Python控制流



三种基本的编程结构



- ①顺序结构：一个命令接一个命令的执行
- ②选择/分支结构：根据判断条件选择不同的语句进行执行
- ③循环结构：在条件成立时反复执行语句

- 在Python中通常的情况下程序的执行是从上往下（从左往右执行）的，执行过程中会自动忽略空格：
 - 逻辑行主要是指在执行意义上的代码行，而物理行指的是编辑器中实际的行，通常习惯上一个逻辑行占用一个物理行。
 - 一个物理行可以包含多个逻辑行，只要在逻辑行与逻辑行之间用分号隔开，但如果一个逻辑行占了一个物理行的最后，这个逻辑行可以省略分号。
 - 一个逻辑行可以写在多个物理行中，如果代码中包含括号（任意用途的圆括号、方括号、花括号），可以在括号内换行；也可以使用行连接符“\”——放在未完成的行尾，代表与下一物理行代码合并，使用行连接符可以在注释以外的任意位置换行，甚至在变量名中间。

- 分支结构：Python中if语句是用来判断选择执行哪个语句块的，其语法结构如下：

```
if <True or Flase表达式>:  
    执行语句块  
elif <True or Flase表达式>:  
    执行语句块  
else:      # 都不满足  
    执行语句块  
  
# elif子句可以有多条，elif和else部  
分可省略
```

- Python中不使用花括号来表示语句块，而是使用缩进，一般采用四个空格的缩进方式，相同缩进量的若干条语句表示一个语句块

- 循环是让计算机做重复任务的有效的方法，Python中可以使用while循环或for循环：

while循环

```
while <True or Flase表达式>:  
    循环执行语句块  
else:    # 不满足条件  
    执行语句块
```

for循环

```
for (条件变量) in (集合):  
    执行语句块  
  
# 条件变量可以有多个
```

#else部分可以省略

- 示例中的“集合”并不单指set，而是“形似”集合的列表、元组、字典、数组等都可以进行循环；
- 有些时候，如果代码写得有问题，会让程序陷入“死循环”，可以使用ctrl+c退出，或强制结束进程

break、continue和pass

- break、continue一般与循环中的分支结构if一起使用，可以用来跳出循环；pass用于占位，解释器遇到pass什么也不做，如果在循环或分支结构中，没有想好某段代码块怎么写，可以使用pass占据该位置，以使得整个结构是符合语法规则的。
 - break:跳出整个循环结构
 - continue:跳出当前循环，执行下一个循环
 - pass:占位符，什么也不做

列表生成式

- 列表生成式用循环结构和分支结构生成指定的列表，在数据分析当中非常实用，常见有三种形式：

```
[<表达式> for (条件变量) in (集合)]
```

```
[<表达式> for (条件变量) in (集合) if <'True or  
False'表达式>]
```

```
[<表达式> if <'True or False'表达式> else <表达式>  
for (条件变量) in (集合) ]
```

- 其中：for循环可以是多个，可以生成每个集合全排列以实现多层的循环；条件变量也可以有多个使用

4 Python函数



调用函数

- 函数是用来封装特定功能的实体，可对不同类型和结构的数据进行操作，达到预定目标。
- Python内置了很多有用的函数，我们可以直接调用，进行数据分析时多数情况下是通过调用定义好的函数来操作数据的
 - 要调用一个函数，需要知道函数的名称和参数，调用函数的时候，如果传入的参数数量不对或类型不对，会报TypeError的错误
 - Python的函数具有非常灵活的参数形态，既可以实现简单的调用，又可以传入非常复杂的参数，参数类型包括必选参数、默认参数、可变参数、关键字参数等
 - 函数在python当中具有与其它变量相同的地位。

定义函数(def)

- 当系统自带函数不足以完成指定的功能时，需要用户自定义函数来完成。

```
def 函数名():  
    函数内容  
    函数内容  
    <return 返回值>
```

- 定义函数使用def语句，依次写出函数名、括号、参数和冒号“:”，然后，在缩进块中编写函数体，函数的返回值用return语句返回。
- 函数体内部的语句在执行时，一旦执行到return时，函数就执行完毕，并将结果返回。通过条件判断和循环可以实现非常复杂的逻辑。
- 如果没有return语句，函数执行完毕后也会返回结果，只是结果为None；另外函数也可以返回多个值

高阶函数(Higher-order function)

- 把另一个函数作为参数传入一个函数，这样的函数称为高阶函数
 - 函数在Python中具有与变量相同的地位，变量可以指向函数，函数名也是变量
 - 变量可以指向函数，函数的参数能接收变量，那么一个函数就可以接收另一个函数作为参数
 - Python内建的函数中就包括了像map、reduce、filter这样的高阶函数，也可以自定义高阶函数实现更丰富的功能。

匿名函数(lambda)

- 当在高阶函数中传入函数时，有些时候，不需要显式地定义传入的函数名，直接传入匿名函数更方便，Python对匿名函数提供了有限支持
 - 只有函数体没有函数名的函数成为匿名函数，python中使用lambda关键字定义匿名函数。
 - 匿名函数可以实现比较简单的函数功能，可以仅用一条语句实现的功能可以使用匿名函数
 - 匿名函数可以作为高阶函数的参数，函数return的结果也可以是匿名函数
 - 匿名函数的参数可以有一个或多个

5 Python模块



模块(module)

- 在计算机程序的开发过程中，随着程序代码越写越多，在一个文件里代码就会越来越长，越来越不容易维护。
- 为了编写可维护的代码，可以把很多函数分组，分别放到不同的文件里。这样，每个文件包含的代码就相对较少，很多编程语言都采用这种组织代码的方式。在Python中，一个.py文件就称之为一个模块（Module）。
- 模块的好处是大大提高了代码的可维护性。其次，编写代码不必从零开始。当一个模块编写完毕，就可以被其他地方引用。我们在编写程序的时候，也经常引用其他模块，包括Python内置的模块和来自第三方的模块
- 另外，还可以通过包(package)来组织模块，简单来理解可以认为包就是包含了模块的“特殊文件夹”

模块的引入(import)

- 使用import可以将模块引入当前程序中，引入后就可以使用模块中定义好的类、函数、变量等，引入方法有：
 - `import os`
 - `import pandas as pd`
 - `from pandas import DataFrame`
 - `from pandas import *`

数据分析常用的包和模块

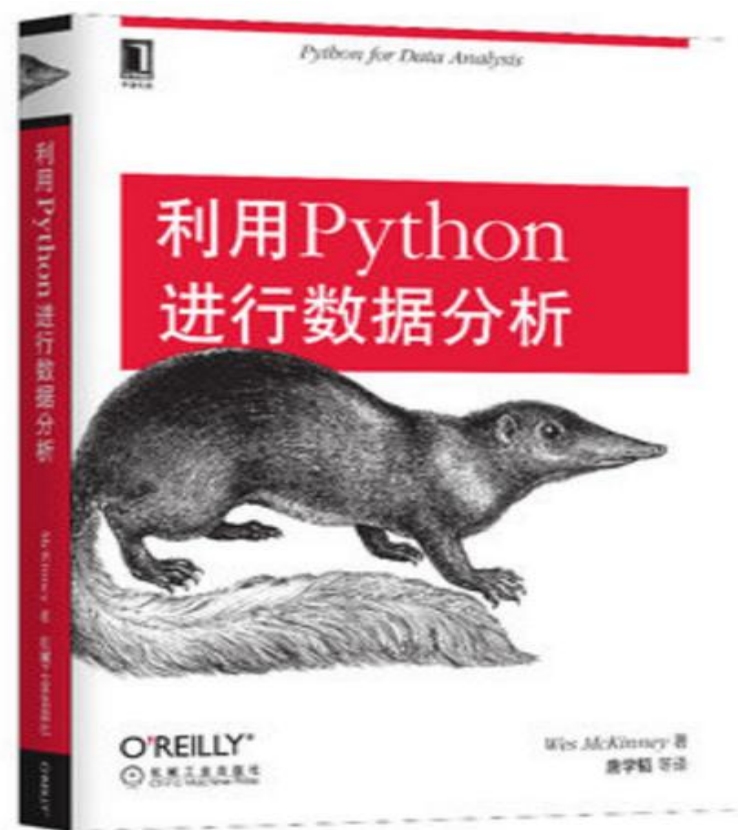
- numpy : 数组、向量、矩阵、数值运算等
- scipy : 统计推断、统计检验等
- pandas : 数据读取、数据整合、数据清洗整理等
- statsmodel : 统计建模、模型验证等
- scikit-learn : 数据清洗、机器学习建模、交叉验证等
- matplotlib : 绘图
- nltk : 自然语言处理
-

6 pandas读写结构化数据

- pandas(Python Data Analysis Library)是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。包含以下数据结构：
 - Series：一维数组，与Numpy中的一维array类似
 - Time- Series：以时间为索引的Series。
 - DataFrame：二维的表格型数据结构。
 - Panel：三维的数组，可以理解为DataFrame的容器，用于分析面板数据
- pandas可以读取文本文件、json、数据库、Excel等文件
- 使用read_csv方法读取以逗号分隔的文本文件作为DataFrame，其它还有类似read_table, read_excel, read_html, read_sql等等方法



ZedShaw完善了这个堪称世上较好的Python学习系统。只要跟着学习，你就会和迄今为止数十万Zed教过的初学者一样获得成功。



pandas作者麦金尼撰写，包括numpy、pandas数据整理、pandas绘图等内容的介绍

—— 秦路主讲 ——
七周成为数据分析师
七周为期，Get一条数据分析师职业黄金通道！



—— Python ——
数据分析与挖掘
集Python爬虫、数据采集、数据处理、数据分析与数据挖掘于一体，打造Python全栈工程师
主讲老师：韦玮
VIP会员群+在线答疑+录播复习+1年反复观看



案例为师，实战为王
开启Python机器学习之路
科学规划全套课程体系，从入门到进阶，从理论到技巧，嵌入丰富课程案例讲解，逐步推进
讲师：唐宇迪 深度学习领域多年一线实践研究专家



**独一无二的
数据仓库**建模指南系列教程升级版
• 从企业视角进行数据规划以及数据仓库模型的搭建
• 高质量的数据库模型和技巧，以及丰富的例子
• 数据仓库架构理论和实践要领
资深讲师：BAO胖子 15年+BI从业经验
涉足电力、快消品、医药、信息服务行业的BI老兵



业务知识一站通
技术+业务，挣钱有门路！
—— 讲师：陈文 ——



自己动手 丰衣足食
Python3网络爬虫实战案例
—— 循序渐进，案例为王，诠释全面，思路制胜 ——
讲师：崔庆才 北航硕士，百万级热度爬文博主



讲师 丘祐玮
人人都爱数据科学家
Python数据科学精华实战课程



数据分析报告制作
秘籍升级版
讲师：陈丹奕 知乎大神，前百度资深数据分析师



先机致胜 破冰AI
—— 深度学习模型/框架与实战 ——
讲师：唐宇迪 同济大学硕士
深度学习领域多年一线实践研究专家



BI、商业智能
数据挖掘 大数据
数据分析师
R语言 Python
机器学习
深度学习
人工智能
Hive Hadoop
Tableau
BIEE ETL
数据科学家
PowerBI