

# 法律声明

---

□ 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

■ 微信公众号：大数据分析挖掘

■ 新浪微博：ChinaHadoop



# 第一讲



# 工作环境准备及 Python数据结构讲解

--梁斌

# 目录

---

- 课程介绍
- 工作环境准备
- Python语言基础回顾
- Python数据结构讲解
- Python高级特性
- Python高阶函数

# 目录

---

- 课程介绍
- 工作环境准备
- Python语言基础回顾
- Python数据结构讲解
- Python高级特性
- Python高阶函数

# 课程介绍

## 互联网最热职位排序

## 《2016年中国互联网最热职位人才库报告》

基于领英发布的互联网行业职位信息，我们统计出了当前互联网行业最热门的六大需求职位，分别是研发工程师、产品经理、人力资源、市场营销、运营和数据分析岗位。可以看出，这些职位都是当下任何互联网公司要建立发展必不可少的岗位，尤其是数据分析人才，伴随着大数据在互联网行业更多的应用而愈加重要。

- 1 研发工程师
- 2 产品经理
- 3 人力资源

- 4 市场营销
- 5 运营
- 6 数据分析

# 课程介绍

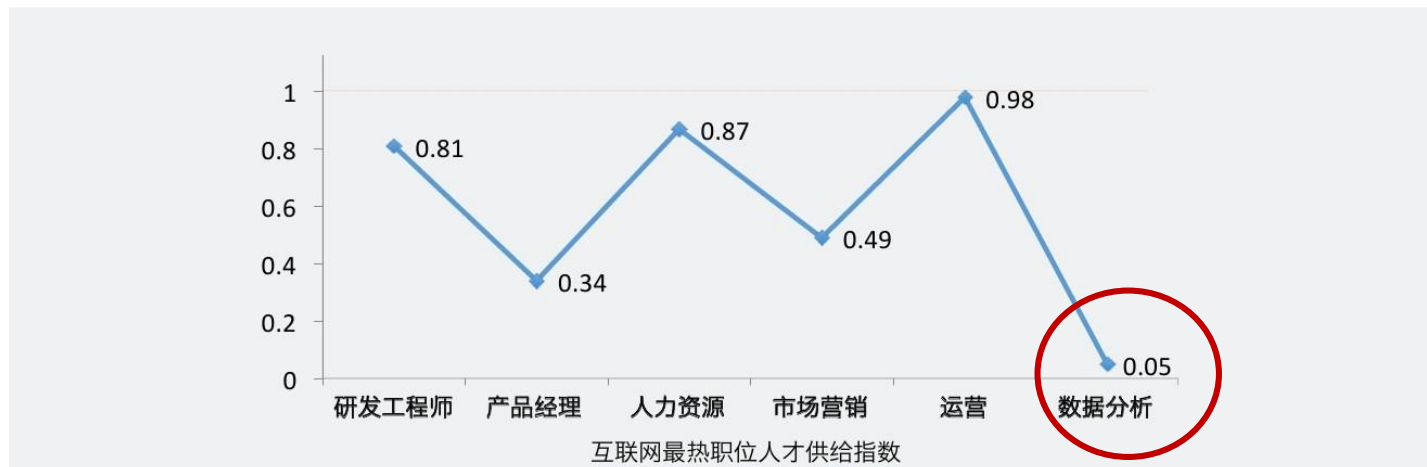
## 人才库概览

## 《2016年中国互联网最热职位人才库报告》

### ／人才供给指数／

截至2015年第四季度，领英共有将近50万的中国互联网行业人才。从互联网行业热门职位人才供给指数中发现，总体而言，这六类热招职位的人才都处于供不应求的状态，但是稀缺程度各有不同。

**数据分析人才的供给指数最低**，仅为0.05，这在一定程度上反映了行业现状，很多互联网公司都逐渐意识到了数据的重要性，但却缺乏相关的专业人才来分析和数据。第二**供不应求的职位是产品经理**，这一职位一方面在互联网公司的地位越来越重要，另一方面对人才的能力要求很高。研发工程师职位作为热门职位的第一名，需求量大，但市场供给并未如想象中的困难，得益于前几年的“计算机热”，该职位在高校中相关专业的人才储备较足。此外，在越来越多企业重视营销的今天，**市场营销人才却显得有些没跟上步伐**，不如运营人才在互联网行业供求较为平衡。



在这份数据报告中，互联网最热职位人才供给指数定义为，对互联网行业而言，在指定期间内，人才在行业内的供给比例/互联网企业的职位需求比例。如果人才供给指数超过1则表示该职位人才供过于求，如果指数小于1则表示该职位人才供不应求。

5

# 课程介绍 什么是数据分析?

---

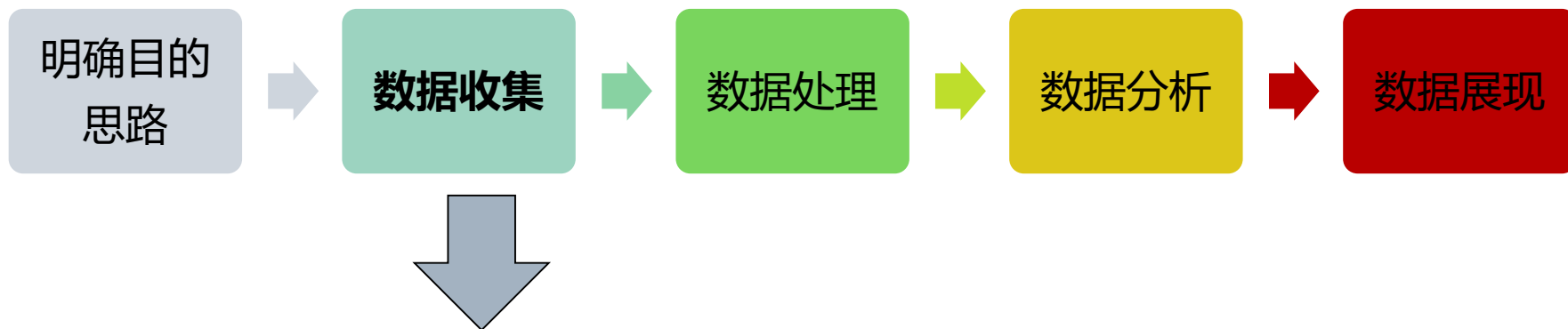
Analysis of data is a process of **inspecting**, **cleansing**, **transforming**, and **modeling** data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.

-- WIKIPIDIA



# 课程介绍 基本步骤

---



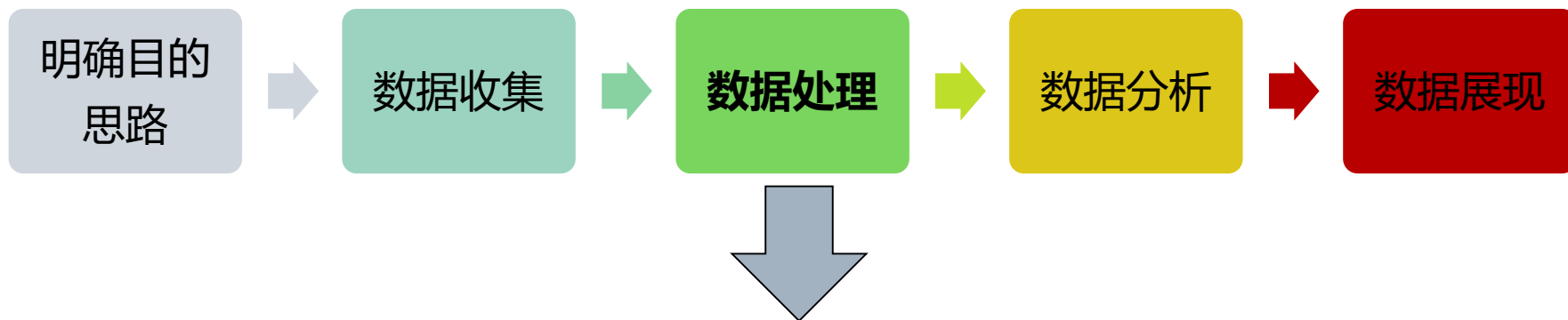
第三课：本地数据的采集与操作

第四课：网络数据的获取与表示



# 课程介绍 基本步骤

---

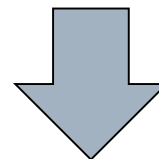
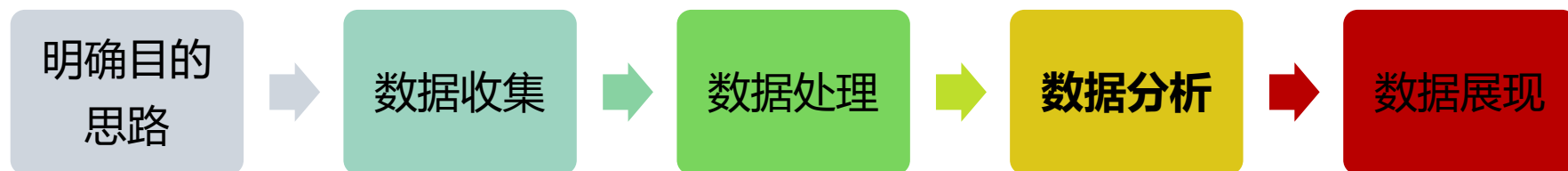


第一课：工作环境准备及Python数据结构讲解

第七课：数据的规整

# 课程介绍 基本步骤

---

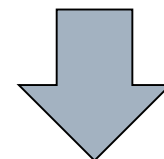


第二课：数据科学计算

第五、六课：数据分析工具Pandas

# 课程介绍 基本步骤

---



第二、七课：数据可视化

第八课：机器学习基础及scikit-learn入门

# 课程介绍 常用工具对比

---

- R/Matlab：适合数据分析，但在生产环境中需要转换成Scala或Python；
- Python：拥有众多库，可以完全只用Python构建以数据为中心的应用程序，不仅适用于研究和原型构建，同时也适用于构建生态系统
- Scala：用于处理大规模数据
- SAS：...
- SPSS：...



# 目录

---

- 课程介绍
- 工作环境准备
- Python语言基础回顾
- Python数据结构讲解
- Python高级特性
- Python高阶函数

# 环境准备 Python 2 or Python 3

---

区别：

- Python 2.x 是早期版本，Python 3.x是当前版本
- Python 2.7 (2.x的最终版)于2010年发布后很少有大的更新
- Python 2.x 比 Python3.x 拥有更多的工具库
- 大多数Linux系统默认安装的仍是 Python 2.x
- 版本选择取决于要解决的问题

建议选择 Python 2.x 的情况：

- 部署环境不可控，Python版本不能自行选择
- 某些工具库还没有提供支持 Python 3.x。如果选择使用 Python 3.x，需要确定要调用的工具库支持新版本。

# 环境准备 Python 2 or Python 3

---

如何兼容：

- Six库提供了一些简单的工具用来封装Python 2.x和Python3.x之间的差异性。
- 代码复杂

主要语法区别举例：

- print

```
# Python 2 only:  
print 'Hello'  
  
# Python 2 and 3:  
print('Hello')
```

- 同时输出多个字符串

```
# Python 2 only:  
print 'Hello', 'ChinaHadoop'  
  
# Python 2 and 3:  
from __future__ import print_function  
# (at top of module)  
  
print('Hello', 'ChinaHadoop')
```

# 环境准备 Python 2 or Python 3

---

主要语法区别：

- Print输出不换行

```
# Python 2 only:  
print 'Hello',  
  
# Python 2 and 3:  
from __future__ import print_function  
  
print('Hello', end='')
```

- 更多区别请查看ppt最后的参考链接
- 本课程使用 Python 2.7



# 环境准备 Python 2 Python 3 编码

---

- ASCII：早起计算机保存英文字符的编码方式
- GB2312：对ASCII的中文扩展
- GBK/GB18030：包括了GB2312的所有内容，同时又增加了近20000个新的汉字和符号
- Unicode：包括了全球的符合和编码。每个符号用3~4个字节表示，浪费空间
- **UTF-8**：变长的编码方式，在互联网上使用最广的一种Unicode的实现方式

# 环境准备 Python 2 Python 3 编码

---

- 建议：
  1. 设置文件编码方式
  2. 文件头部指定的编码方式与文件保存编码方式一致
  3. 尽量使用UTF-8

示例代码：`coding_utf8_example.py`

# 环境准备 Python环境及IDE

---

- Python环境

Anaconda <https://www.continuum.io/downloads>

本课程中涉及到的工具库会在相应章节中介绍其安装过程

- IDE

## Jupyter notebook

1. Anaconda自带，**无需单独安装**
2. 记录思考过程，实时查看运行过程
3. 基本web的在线编辑器（本地）
4. .ipynb文件分享

([https://github.com/ymgd/codereader/blob/master/dataanalysis/python/data\\_overview.ipynb](https://github.com/ymgd/codereader/blob/master/dataanalysis/python/data_overview.ipynb))

# 环境准备 IDE

---

- IDE

## Jupyter notebook

- 5. 可交互式
- 6. 记录历史运行结果
- 7. 支持Markdown, Latex

## IPython

- 1. Anaconda自带, 无需单独安装
- 2. Python的交互式命令行 Shell
- 3. 可查看历史操作
- 4. 及时验证想法

# 环境准备 IDE

---

- IDE -- 没有最好的，只有**最适合自己的**（以下选一个就可以）

**Eclipse + PyDev**，完全免费，适合熟悉Eclipse或Java的开发者

1. Eclipse, <https://eclipse.org/downloads/>

2. PyDev插件, <https://marketplace.eclipse.org/content/pydev-python-ide-eclipse>

**PyCharm**，部分免费，可满足不涉及web的开发，适合大多数开发者

<https://www.jetbrains.com/pycharm/download/>

**Spyder**，完全免费，适合熟悉Matlab的开发者

<https://github.com/spyder-ide/spyder>

# 环境准备 本课程的开发环境

---

- Anaconda
- Jupyter notebook
  - 演示
  - 分享
- IPython
  - 验证思路
- Eclipse + PyDev（与PyCharm类似）
  - 编写完整项目
  - 调试

# 目录

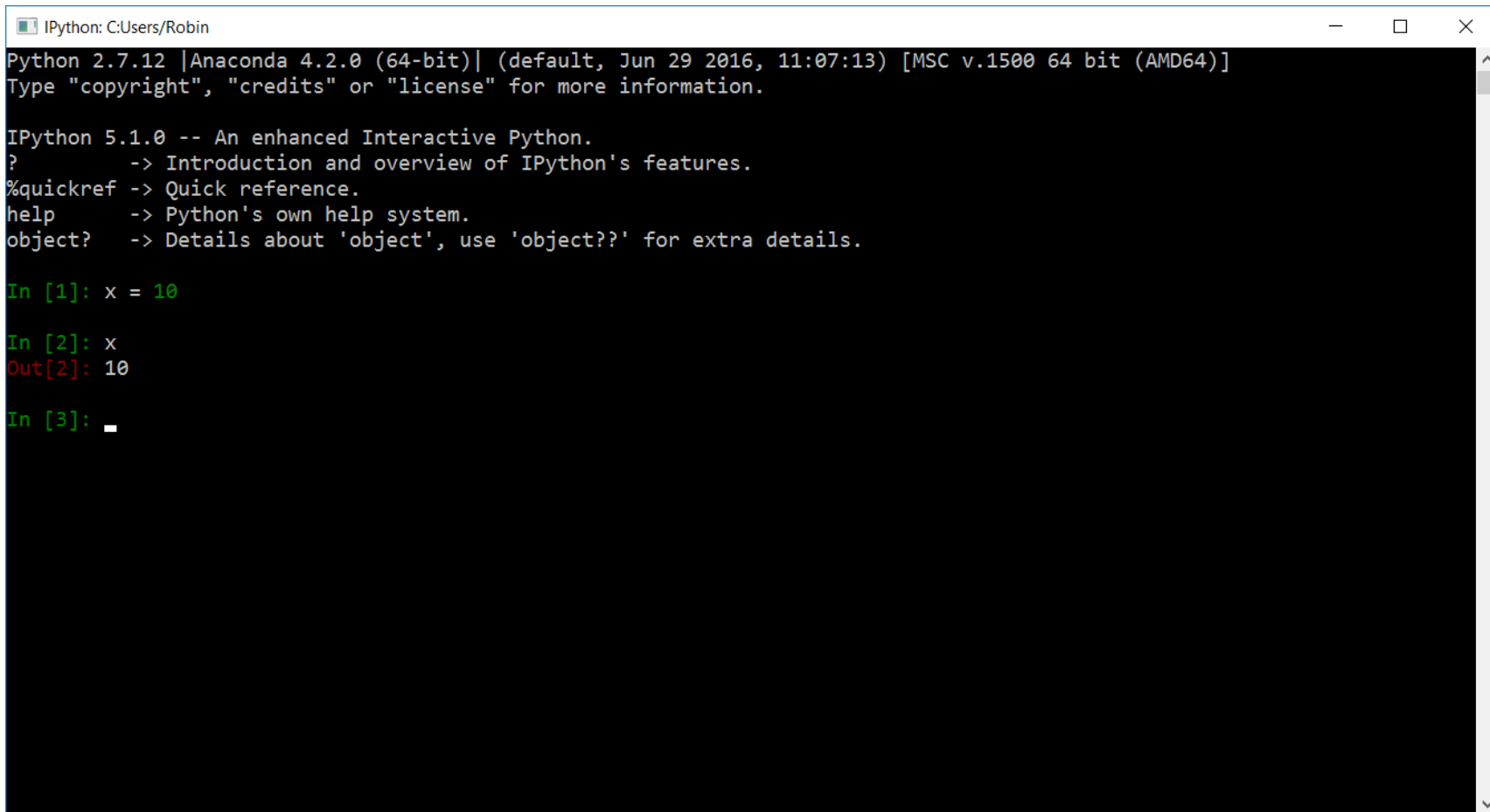
---

- 课程介绍
- 工作环境准备
- Python语言基础回顾
- Python数据结构讲解
- Python高级特性
- Python高阶函数

# Python基础

# IPython基础

启动IPython: >> ipython



```
IPython: C:\Users\Robin
Python 2.7.12 |Anaconda 4.2.0 (64-bit)| (default, Jun 29 2016, 11:07:13) [MSC v.1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]: x = 10

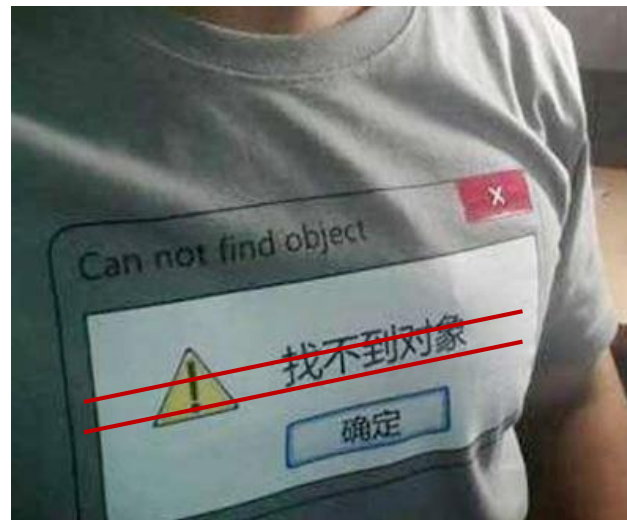
In [2]: x
Out[2]: 10

In [3]: _
```



? 和 ??

- Python对象+ (?)  
显示该对象的基本信息
- Python对象+ (??)  
如果是函数，会显示源码
- 注：python对象可为变量、函数等。



示例代码： `ipython_basic.ipynb`

## IPython魔术命令

- `%timeit`

多次执行一条语句，并返回平均时间，`%%timeit`->多条语句

- `%time`

返回执行一条语句的时间，`%%time`->多条语句

- `%reset`

删除当前空间的全部变量

- `%run *.py`

在IPython中执行Python脚本

- 魔术命令+(?)显示文档

如：`%time?`

示例代码：`ipython_basic.ipynb`

# Python基础

---

- 缩进，不是{}
  - 保证了风格统一
  - 方便阅读
  - 编写大段代码时容易迷失。。。需要配合IDE提示
- 对象模型
  - Python中的任何数值、字符串、数据结构、函数、类、模块等都是对象。
- 注释
  - #
- 引入其他工具库或模块
  - import

示例代码：`python_basic.ipynb`

# Python基础

---

- 变量不需指定类型，但是仍是“强类型语言”，只是不显示地表示
  - 注意：Python中变量是没有类型的，**对象才有类型**  
如 `x = 5`，变量`x`是没有类型的，而是指向整型对象`5`的一个变量。
- 可变与不可变
  - 大部分Python对象是可变(`mutable`)的，e.g. 列表、字典、自定义的类
  - 字符串和元组是不可变的(`immutable`)。

示例代码：`python_basic.ipynb`

# Python基础

---

- 字符串
  - 单引号(''), 双引号("")均可
  - 三引号(""" """)一般用于放置文档说明(docstring)或多行字符串
  - 字符串格式化, (%)
- 类型转换
  - `str bool int float`
- 时间和日期
  - `datetime`模块
  - `strftime`将`datetime`类型格式化为字符串

示例代码：`python_basic.ipynb`

# Python基础

---

- 时间和日期
  - `strptime`将字符串解析为`datetime`类型
- `if`、`elif`、`else`
- `for` 循环
  - `for item in collection`
  - `continue`, `break`
- `while`循环
- 没有`++`, `--`, 可用 `+= 1`, `-= 1` 代替
- `pass`用于占位
  - Python中不允许空代码块
- 处理异常
  - `try except`

示例代码：`python_basic.ipynb`

# Python基础

---

- range 和 xrange
  - 生成整数列表
  - xrange比range效率高，但是xrange不会预先生成列表，而是一个迭代器
- 关于Python中的引用
  - 变量复制、传参是值传递
  - 列表、字典的复制、传参是传引用
- 浅拷贝(copy)和深拷贝(deepcopy)
  - 浅拷贝效果：[:]切片操作，copy()。只拷贝父对象，不拷贝对象的内部子对象
  - 深拷贝：同时拷贝父对象及子对象

示例代码： `python_basic.ipynb`

# 目录

---

- 课程介绍
- 工作环境准备
- Python语言基础回顾
- **Python数据结构讲解**
- Python高级特性
- Python高阶函数



# Python数据结构

---

## 元组 tuple

- 一维、定长、不可变的对象序列
- 创建元组 ()
- 转换为元组, list->tuple, string->tuple
- 访问元组
- 合并元组 +
- 拆包
  - 函数同时返回多个值
  - 元组列表迭代
- 常用方法 count
  - 返回指定值得次数

示例代码： `python_data_structure.ipynb`

# Python数据结构

---

## 列表 list

- 变长，可变
- 创建列表 []
- 转换为列表，tuple->list
- 添加、移除元素，append, insert, pop, remove
- 合并列表 +, extend
- 排序操作 sort (注：**就地排序**，无需创建新对象)
- 切片操作 [start\_idx : stop\_idx : step]

注：结果不包含stop\_idx的元素

示例代码：`python_data_structure.ipynb`

# Python数据结构

---

## 常用序列函数

- `enumerate` , `for`循环时记录索引, 逐个返回元组(`i`, `item`)
- `sorted` 返回新的有序列表, 区别: `list`中的`sort()`是就地排序
- `zip` “压缩” 将多个序列的对应位置的元素组成元组
- `zip(*元组列表)` “解压缩”, `zip`的逆操作
- `reversed` 逆序迭代, 可配合`list`返回逆序列表

示例代码: `python_data_structure.ipynb`

# Python数据结构

---

## 字典 dict

- {key1:value1, key2:value2}

- 创建字典

- 插入元素

```
dict_variable[new_key] = new_value
```

- 删除元素

```
del 或 pop
```

- 获取键、值列表

```
keys(), values()
```

- 合并字典

```
update()
```

示例代码：`python_data_structure.ipynb`

# Python数据结构

---

## 字典 dict

- 通过多个列表创建字典
- 哪些可做“键”
  - 整数、浮点数、字符串或元组
  - 可“哈希”的对象，hash

## 集合 set

- 创建集合 {}
- 集合操作，交、并、差、异或

示例代码：`python_data_structure.ipynb`

# 目录

---

- 课程介绍
- 工作环境准备
- Python语言基础回顾
- Python数据结构讲解
- Python高级特性
- Python高阶函数

# Python高级特性

## 集合的推导式

- 列表推导式，使用一句表达式构造一个新列表，可包含过滤、转换等操作

`[exp for item in collection if condition]`

- 字典推导式

`{key_exp : value_exp for item in collection if condition}`

- 集合推导式

`{exp for item in collection if condition}`

- 嵌套列表推导式

按嵌套顺序理解

示例代码：`python_advanced_feature.ipynb`

# Python高级特性

---

## 函数列表

- Python中皆对象
- [fun1, fun2]

## 匿名函数 lambda

- 没有函数名
- 单条语句组成
- 语句执行的结果就是返回值
- 可用作sort的key函数

示例代码： `python_advanced_feature.ipynb`

## 生成器 generator

- 构造可迭代对象
- 每次返回一个值，直到下次调用时，再继续。区别：函数每次返回一个值
- yield



# 目录

---

- 课程介绍
- 工作环境准备
- Python语言基础回顾
- Python数据结构讲解
- Python高级特性
- Python高阶函数

# Python高阶函数

示例代码：

```
python_higher_order_functions.ipynb
```

## 函数式编程

- 函数本身可以赋值给变量。赋值后变量为函数
- 允许将函数本身作为参数传入另一个函数
- 允许返回一个函数

## map/reduce

- `map(func, lst)`，将传入的函数变量func作用到lst变量的每个元素中，并将结果组成新的列表返回
- `reduce(func(x,y), lst)`，其中func必须有两个参数。每次func计算的结果继续和序列的下一个元素做累积计算。
  - `lst=[a1, a2, a3, ..., an]`
  - `reduce(func(x,y), lst)`  
`= func(func(func(a1, a2), a3), ..., an)`

# Python高阶函数

---

## filter

- 筛选序列
- `filter(func, lst)`，将func作用于lst的每个元素，然后根据返回值是True或False判断是保留还是丢弃该元素。

map，reduce和filter中的函数变量均可以是匿名函数

示例代码：

```
python_higher_order_functions.ipynb
```

# 参考

---

- Python2 or Python3

<https://wiki.python.org/moin/Python2orPython3>

- Six

<https://pythonhosted.org/six/>

- Python 2.x 与 Python 3.x 语法区别

[http://python-future.org/compatible\\_idioms.html](http://python-future.org/compatible_idioms.html)

- Python进阶必读

<http://www.kuqin.com/shuoit/20151116/348975.html>

- Python生成器

<http://book.pythontips.com/en/latest/generators.html>

- Map, Reduce和Filter

[http://book.pythontips.com/en/latest/map\\_filter.html](http://book.pythontips.com/en/latest/map_filter.html)

# 参考

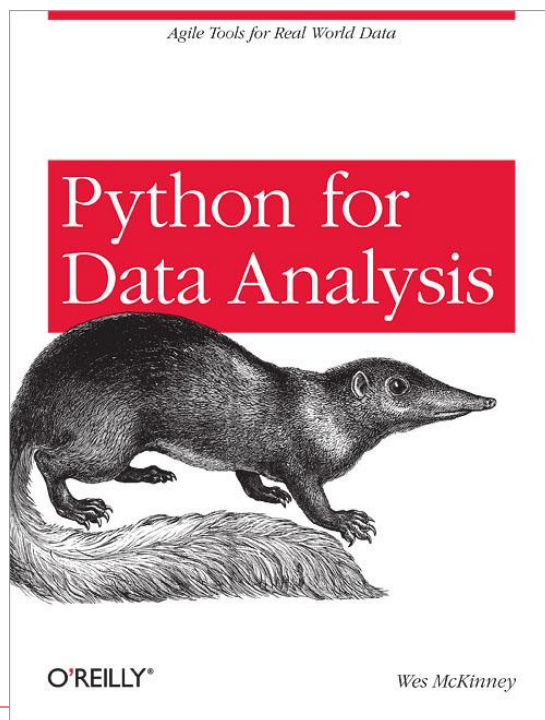
---

- Python中的可变与不可变

<http://book.pythontips.com/en/latest/mutation.html>

- Python中的浅拷贝与深拷贝

[http://www.python-course.eu/python3\\_deep\\_copy.php](http://www.python-course.eu/python3_deep_copy.php)



# 疑问

---

□ 问题答疑：<http://www.xxwenda.com/>

■ 可邀请老师或者其他回答问题

小象问答 @Robin\_TY

# 联系我们

---

## 小象学院：互联网新技术在线教育领航者

- 微信公众号：小象
- 新浪微博：ChinaHadoop

