Java

# Python安装及IDE环境配置

2017/10/29安装Python3.6版本或Python2.7.14版本python-2.7.14（分别下载到D盘soft中）

2017/10/29 Python2安装目录D:\Program Files\Python2

2017/10/29 Python3安装目录D:\Program Files\Python36

2017/10/29下载简明Python教程

2017/10/29在Eclipse中安装Pydev的IDE开发环境并设置。发现Java版本太低，不能用。

<http://www.runoob.com/python/python-ide.html>

10/29/2017安装PyCharm教育版. pycharm-edu-4.0.2仍然由于JDK7不能使用该文件

## Python标准库

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

Python标准库地址:

<https://docs.python.org/3/library/index.html>

包含两类: 1: Build-in类型; 2: 其它标准库如sys, os等

### Build-in

#### Text Sequence Type — str

Str.split()

>>> '1 2 3'.split()

['1', '2', '3']

>>> '1 2 3'.split(maxsplit=1)

['1', '2 3']

>>> ' 1 2 3 '.split()

['1', '2', '3']

## Python第三方库

Python Package Index: PyPI

<https://pypi.python.org/pypi>

## Anaconda

### 2018/02/06安装Anaconda：

Anaconda是Python的一个开源发行版本，主要面向科学计算。

python是进入python交互命令行；ipython是进入ipython交互命令行，很强大；conda是Anaconda的配置命令；jupyter notebook则会启动Web端的ipython notebook。[[1]](#footnote-1)

Anaconda安装成功之后，我们需要修改其包管理镜像为国内源：

conda config --add channels <https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/>

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/win-32/

conda config --set show\_channel\_urls yes

希望大家从此不再因为选择版本、安装Python、管理环境、添加第三方库（能用conda用conda，不能用conda用pip）、选择IDE（单文件Jupyter，项目组织开Pycharm）等等杂事所困扰，专心于代码。

### Conda

环境管理：

# 创建一个名为python34的环境，指定Python版本是3.4（不用管是3.4.x，conda会为我们自动寻找3.4.x中的最新版本）

conda create --name python34 python=3.4

# 安装好后，使用activate激活某个环境

activate python34 # for Windows

source activate python34 # for Linux & Mac

# 激活后，会发现terminal输入的地方多了python34的字样，实际上，此时系统做的事情就是把默认2.7环境从PATH中去除，再把3.4对应的命令加入PATH

# 此时，再次输入

python --version

# 可以得到`Python 3.4.5 :: Anaconda 4.1.1 (64-bit)`，即系统已经切换到了3.4的环境

# 如果想返回默认的python 2.7环境，运行

deactivate python34 # for Windows

source deactivate python34 # for Linux & Mac

# 删除一个已有的环境

conda remove --name python34 –all

安装包：

# 安装scipy

conda install scipy

# conda会从从远程搜索scipy的相关信息和依赖项目，对于python 3.4，conda会同时安装numpy和mkl（运算加速的库）

# 查看已经安装的packages

conda list

# 更新conda，保持conda最新

conda update conda

# 更新anaconda

conda update anaconda

# 更新python

conda update python

# 假设当前环境是python 3.4, conda会将python升级为3.4.x系列的当前最新版本

### Spyder

#### 快捷键

Ctrl + I : 找出函数的help文档

F9：run selection: 运行选择的部分代码段

F5：执行文件Ctrl+F5：debug

F6：重新执行上次的script

Ctrl+Enter： executes the current cell (menu entry Run > Run cell). A cell is defined as the code between two lines which start with the agreed tag #%%

Shift+Enter：执行当前cell并跳转到下一个cell

Alt+<Up Arrow>：移动当前选择行上下 moves the current line up

Ctrl+Shift+F activates the search pane across all files.

Ctrl+F10：execute the code line by line

Ctrl+F11：execute the code line stepping into it

Ctrl+F12：get out of a function and continue with the next line

F12：插入断点

Ctrl + 1： 注释/反注释函数

#### 函数：

%reset: 清空定义的reset the name space；或者ctrl+F6：选择第二种console

#### 函数注释：

def average(a, b):

"""

Given two numbers a and b, return their average value.

Parameters

----------

a : number

A number

b : number

Another number

Returns

-------

res : number

The average of a and b, computed using 0.5\*(a + b)

Example

-------

>>> average(5, 10)

7.5

"""

return (a + b) \* 0.5

### Jupyter Notebook

### Git

#### 命令行使用Git

1. Cmd环境下, 直接git命令E.g. git --version判断安装成功git
2. 初次运行 Git 前的配置git config:
   1. **用户信息:** 当安装完 Git 应该做的第一件事就是设置你的用户名称与邮件地址。
   2. $ git config --global user.name "Rui Xu"
   3. $ git config --global user.email [riven.china@gmail.com](mailto:riven.china@gmail.com)
   4. **文本编辑器**: 安装时已连接到VS Code
   5. 检查配置信息 git config –list
3. Git 基础 - 获取 Git 仓库[[2]](#footnote-2)
   1. Cmd进入准备创建repository的目录, 运行$ git init命令初始化
   2. 添加文件到Git仓库，分两步：
   3. 第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；
   4. 第二步，使用命令git commit –m “zhu shi”
4. Git基础 – 版本控制
   1. Git status :查看状态
   2. Git log: 查看日志
   3. 其它版本控制尽量用vs code等gui工具
5. Github – 远程仓库(使用HTTP协议)[[3]](#footnote-3)
   1. 本地工程与github中的repository建立关联
   2. git remote add origin <https://github.com/riven521/learngit.git>
   3. git pull origin master
   4. 修改文件，提交改动
   5. git add README.md
   6. git status
   7. git commit -m "增加README.md"
   8. git status
   9. git push origin master
6. **案例:vs code github下载到本地[[4]](#footnote-4)**
   1. 复杂方法:
   2. Git init
   3. git remote add origin [https://github.com/riven521/\*\*\*\*.git](https://github.com/riven521/****.git)
   4. git pull origin master(或vs code的发布功能) 输入用户名和密码
   5. 简单方法:
   6. 当然我们也可以使用git 的clone命令，从远程克隆一个Reps，然后直接用vscode打开文件夹， VS Code 会自动识别各项配置。
   7. **Github创建仓库, 通过vs code克隆该仓库**
   8. 快捷键ctrl+shipt+p后git clone: 输入地址 <https://github.com/riven521/MLiA.git>及存储目录
   9. 连接后-同步即可同步本地和Github.
   10. 持久化账号
   11. git config --global credential.helper wincred

# python可视化

## 可视化工具

以下是 2016 年 4 月写的关于绘图工具的概述。出于多方面的原因，绘图工具的选取更多地取决于个人偏好，因此本文介绍的 Python 绘图工具也仅代表我的个人使用偏好。[[5]](#footnote-5)

Matplotlib

Matplotlib 是一个强大的工具，它是 Pandas’ builtin-plotting 和 Seaborn 的基础。Matplotlib 能够绘制许多不同的图形，还能调用多个级别的许多 API。我发现pyplot api非常好用，你可能用不上Transforms 或者 artists，但是如果你有需求的话可以查阅帮助文档。我将从 pandas 和 seaborn 图开始介绍，然后介绍如何调用 pyplot 的API。

Pandas’ builtin-plotting

DataFrame 和 Series 拥有 .plot 的命名空间，其中有许多图形类别可供选择(line, hist, scatter, 等等)。 Pandas 对象还提供了额外的用于增强图形展现效果的数据，如索引变量。

由于 pandas 具有更少的向后兼容的限制，所以它具有更好的美学特性。从这方面来说，我认为 pandas 中的 DataFrame.plot 是一个非常实用的快速探索性分析的工具。

Seaborn

Michael Waskom 所开发的 Seaborn 提供了一个高层次的界面来绘制更吸引人统计图形。Seaborn 提供了一个可以快速探索分析数据不同特征的 API 接口，接下来我们将重点介绍它。

Bokeh

Bokeh 是一款针对浏览器开发的可视化工具。

和 matplotlib 一样，Bokeh 拥有一系列 API 接口。比如 glpyhs 接口，该接口和 matplotllib 中的 Artists 接口非常相似，它主要用于绘制环形图、方形图和多边形图等。最近 Bokeh 又开放了一个新的图形接口，该接口主要用于处理词典数据或 DataFrame 数据，并用于绘制罐头图。

Bokeh 这个框架，比起 D3.js，它的可视化选项相对较少。因此，目前来看 Bokeh 无法挑战 D3.js 的霸主地位。而且 Bokeh 过于依赖 python 的数值计算库，并非一个纯前端的框架，使得它的使用范围也小于 D3.js。而在纯 python 的数值计算领域，也已经有 matplotlib 这种提供了与 Matlab 一模一样的接口的数据可视化库，Bokeh 的适用场景也并不多。

但是，它非常适合嵌入 Flask 或者 Django 的程序中，非常好用，速度也很快。

其他工具：

以下是一些本文没有提到的可视化工具：

Lightning

HoloViews

Glueviz

vispy

bqplot

# 小技巧

## 如何测试Python已安装成功

Cmd后输入python，出现>>>, 表示成功；否则去path系统变量设置。

print("hello"),保存为\*.py文件. cmd环境目录运行python \*.py即可

## 如何推出python环境

按Ctrl+z，后回车

## 如何给python写注释？

#符号

## 如何安装python第三方库

Pip命令：<http://www.pip-installer.org/en/latest/>

Cmd后：pip install （-u：如已安装升级到最新版）\*\*\*库名即可

pip uninstall：卸载模块

pip list: 已安装哪些库？

### 如何查看库的帮助

python中每个modul，每个class，每个def都是留有写doc的地方的

import scipy

print help(scipy.quiver) 、print scipy.quiver.\_\_doc\_\_

### 出现错误name pip is not defined

Pip已安装，python也添加环境变量。原因是在cmd环境输入pip，而非>>>的环境

### 出现错误：SyntaxError: Non-UTF-8 code starting with '\xc4' in file错误

首行增加：# -!- coding: utf-8 -!-

### 常用的第三方库有哪些？

Django（开放源代码的Web应用框架）、Flask（轻量级 Web 应用框架）、

**numpy（以矩阵为基础的数学计算模块，纯数学）、scipy(科学计算库)、pandas（表格数据容器）**

**ski-learn（）**

<https://www.zhihu.com/question/24590883>

### 机器学习库有哪些？

全部参考：

<https://github.com/sandbreaker/awesome-machine-learning/blob/master/README.md>

利用Python，四步掌握机器学习 <http://python.jobbole.com/84326/>

Scikit-learn的随机森林：

#Import Library

**fromsklearn.ensemble import RandomForestClassifier**

#Assumed you have, X (predictor) and Y (target) for training data set and x\_test(predictor) of test\_dataset

# Create Random Forest object

**model= RandomForestClassifier(n\_estimators=1000)**

# Train the model using the training sets and check score

**model.fit(X, y)**

#Predict Output

**predicted= model.predict(x\_test)**

### 常用机器学习数据集有那些？

UCI机器学习Repository

<http://archive.ics.uci.edu/ml/index.php>

## Python错误集锦

### UnicodeDecodeError: 'gbk' codec can't decode byte 0xae in position 199: illegal multibyte sequence

找到该读取错误的文档,并删除或重新复制黏贴该文档.

### TypeError: 'range' object doesn't support item deletion

原因是python3.x range返回的是range对象，不返回list对象.解决方案是把range对象转化为List对象即可E.g. : dataIndex=list(range(m))#产生一个长度为m的列表

## Python小技巧

### 如何切割字符串

1 – 自带的str.split函数(str.split不支持正则及多个切割符号，不感知空格的数量)

print(bigString.split()) # 4.7. Text Sequence Type — str

2 – re包的正则表达re.split函数

import re

print(re.split(r'\W\*', bigString))

### 如何生成随机数

[a,b]之间的随机数: 0-某个list长度random.unifrom

randIndex = int(random.uniform(0,len(trainingSet)))

### 如何定义函数?

def printme( str ):

print str;

return;

# 调用函数

printme("我要调用用户自定义函数!");

printme("再次调用同一函数");

## Python数据结构技巧

### 如何创建多个0元素的list

v = [0]\*len(vocabList)

### 如何指定列表的第几个值？

v[vocabList.index(word)] = 1 指定列表v的第几个值，并赋值为1；vocabList也是列表，单词word出现的位置。

for word in inputSet:

if word in vocabList:

returnVec[vocabList.index(word)] = 1

### 如何扩展一个list为多维的list？append

trainMat = []

for postinDoc in listOPosts:

trainMat.append(\*\*\*object)

### 如何快速返回多个满足条件的list?

ree = []

for tok in listOfTokens:

if len(tok) > 2:

ree.append(tok.lower())

等同于 = return [tok.lower() for tok in listOfTokens if len(tok) > 2]

### 如何返回多维list的某些特定值?

dataSet = [[1, 1, 'yes'],

[1, 1, 'yes'],

[1, 0, 'no'],

[0, 1, 'no'],

[0, 1, 'no']]

featList = [example[0] for example in dataSet] #create a list of all the examples of this feature

#二维列表的每一个列表为example; 每个列表的第一个值为example[0]; 需要集合所有的值则增加中括号. 返回的值为[1,1,1,0,0]:dataset的第一列.

### 如何快速问去除一维列表中重复的值?

转化为set.

E.g a = [1, 1, 1, 0, 0];

B = set(a)

B = { 0 , 1}

# 示例Python+机器学习

## 3.1基于 Python 和 Scikit-Learn 的机器学习介绍

<http://python.jobbole.com/81721/>

### 数据加载 修改为py3版本

import numpy as np

import urllib.request

# url with dataset

url = "http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"

# download the file

raw\_data = urllib.request.urlopen(url)

# load the CSV file as a numpy matrix

dataset = np.loadtxt(raw\_data, delimiter=",")

# separate the data from the target attributes

x = dataset[:,0:7]

y = dataset[:,8]

### 数据标准化

我们都知道大多数的梯度方法（几乎所有的机器学习算法都基于此）对于数据的缩放很敏感。因此，在运行算法之前，我们应该进行标准化，或所谓的**规格化**。标准化包括替换所有特征的名义值，让它们每一个的值在0和1之间。而对于规格化，它包括数据的预处理，使得每个特征的值有0和1的离差。Scikit-Learn库已经为其提供了相应的函数。

from sklearn import **preprocessing**

# normalize the data attributes

normalized\_X = preprocessing.**normalize**(x)

# standardize the data attributes

standardized\_X = preprocessing.**scale**(x)

### 特征的选取

虽然特征工程是一个相当有创造性的过程，有时候更多的是靠直觉和专业的知识，但对于特征的选取，已经有很多的算法可供直接使用。如树算法就可以计算特征的信息量。

from sklearn import metrics

from sklearn.ensemble import ExtraTreesClassifier

model = ExtraTreesClassifier()

model.fit(X, y)

# display the relative importance of each attribute

print(model.feature\_importances\_)

…..

# Python打包及与其他语言相互调用

## Jython (2015年停止更新)(2017.07.01发布2.7.1版本)[[6]](#footnote-6)

## JyNI [[7]](#footnote-7)

JyNI is a compatibility layer with the goal to enable Jython to use native CPython extensions like NumPy or SciPy. This way we aim to enable scientific Python code to run on Jython. Since Java is quite present in industry and Python is more present in science, JyNI will be an important step to lower the costs of using scientific code in industrial environments. Note that current JyNI requires Jython 2.7.1.

# 机器学习实战

## k-近邻算法 第二章

def classify0(inX, dataSet, labels, k):

dataSetSize = dataSet.shape[0]

diffMat = tile(inX, (dataSetSize,1)) #tile return array

print('diffMat is {0}'.format(diffMat))

diffMat = diffMat - dataSet

sqDiffMat = diffMat\*\*2

print('sqDiffMat is {0}'.format(sqDiffMat))

sqDistances = sqDiffMat.sum(axis=1) #按行求和 axis=1; 按列求和 axis=0

print('sqDistances is {0}'.format(sqDistances))

distances = sqDistances\*\*0.5 #样本点与其余四点的距离

print('distances is {0}'.format(distances))

sortedDistIndicies = distances.argsort() #按从小到达排序后的index numpy.ndarray.argsort

print('sortedDistIndicies is {0}'.format(sortedDistIndicies))

classCount={} #建立空字典

for i in range(k):

voteIlabel = labels[sortedDistIndicies[i]]

print('voteIlabel is {0}'.format(voteIlabel))

classCount[voteIlabel] = classCount.get(voteIlabel,0) + 1 #get(k,default=none) :字典中包括key=k,返回value;否则返回default

print('classCount is {0}'.format(classCount))

sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), reverse=True) #将dict类型classCount的列表，按第二个值value排序，后返回列表。

print('sortedClassCount type is {0} is {1}'.format(type(sortedClassCount),sortedClassCount))

return sortedClassCount[0][0]

## 决策树 第3章

## 朴素贝叶斯 第4章

## Logistic回归

### 多元线性回归

自总: 类似多元线性回归, 找出

### 单位阶跃函数与对数几率函数

线性回归预测值为x实值, 需转换为0/1值. (下图红色为单位阶跃函数,不连续)



### Sigmoid函数:

* 类似S形状的函数



对数几率函数(logistic function)为其

# python简明教程

## 基础函数

Dir(): you get a list of known objects

Help(): 返回def第一行注释docstrings：The documentation string is the first string immediately below the line def

## print为代表的基础

age = 20

name = 'Swaroop'

print('{0} was {1} years old when he wrote this book'.format(name, age))

print('Why is {0} playing with that python?'.format(name))

# 对于浮点数 '0.333' 保留小数点(.)后三位

print('{0:.3f}'.format(1.0/3))

# 使用下划线填充文本，并保持文字处于中间位置

# 使用 (^) 定义 '\_\_\_hello\_\_\_'字符串长度为 11

print('{0:\_^11}'.format('hello'))

# 基于关键词输出 'Swaroop wrote A Byte of Python'

print('{name} wrote {book}'.format(name='Swaroop', book='A Byte of Python'))

print('a', end=' ')

print('b', end=' ')

结果：ab

转义：\ ; 结尾使用\ 代表显式行连接（Explicit Line Joining）

如不需要转义，前面写r：r’\1’ == ‘\\1’

## 控制流

**If：**

number = 23

guess = int(input('Enter an integer : '))

if guess == number:

# 新块从这里开始

print('Congratulations, you guessed it.')

print('(but you do not win any prizes!)')

# 新块在这里结束

elif guess < number:

# 另一代码块

print('No, it is a little higher than that')

# 你可以在此做任何你希望在该代码块内进行的事情

else:

print('No, it is a little lower than that')

# 你必须通过猜测一个大于（>）设置数的数字来到达这里。

print('Done')

# 这最后一句语句将在

# if 语句执行完毕后执行。

**While：**

number = 23

running = True

while running:

guess = int(input('Enter an integer : '))

if guess == number:

print('Congratulations, you guessed it.')

# 这将导致 while 循环中止

running = False

elif guess < number:

print('No, it is a little higher than that.')

else:

print('No, it is a little lower than that.')

else:

print('The while loop is over.')

# 在这里你可以做你想做的任何事

print('Done')

**for**

for i in range(1, 5):

print(i)

else:

print('The for loop is over')

**break/continue用在for和while中**

## 函数def

def say\_hello():

# 该块属于这一函数

print('hello world')

# 函数结束

say\_hello() # 调用函数

**默认参数：**

def say(message, times=1):

print(message \* times)

say('Hello')

say('World', 5)

**关键字参数：**

def func(a, b=5, c=10):

print('a is', a, 'and b is', b, 'and c is', c)

func(3, 7)

func(25, c=24)

func(c=50, a=100)

**可变参数：**

def total(a=5, \*numbers, \*\*phonebook):

print('a', a)

#遍历元组中的所有项目

for single\_item in numbers:

print('single\_item', single\_item)

#遍历字典中的所有项目

for first\_part, second\_part in phonebook.items():

print(first\_part,second\_part)

文档字符串：DocStrings

print(print\_max.\_\_doc\_\_)

我强烈推荐你为你编写的所有重要的函数配以文档字符串。

## 模块

如果你希望直接将 argv 变量导入你的程序（为了避免每次都要输入 sys. ），那么你可以

通过使用 from sys import argv 语句来实现这一点。

from math import sqrt

print("Square root of 16 is", sqrt(16))

模块的 \_\_name\_\_

每个模块都有一个名称，而模块中的语句可以找到它们所处的模块的名称。这对于确定模块

是独立运行的还是被导入进来运行的这一特定目的来说大为有用。正如先前所提到的，当模

块第一次被导入时，它所包含的代码将被执行。我们可以通过这一特性来使模块以不同的方

式运行，这取决于它是为自己所用还是从其它从的模块中导入而来。这可以通过使用模块的

\_\_name\_\_ 属性来实现。

if \_\_name\_\_ == '\_\_main\_\_':

print('This program is being run by itself')

else:

print('I am being imported from another module')

**编写自己的module**

def say\_hi():

print('Hi, this is mymodule speaking.')

\_\_version\_\_ = '0.1'

import mymodule

mymodule.say\_hi()

print('Version', mymodule.\_\_version\_\_)

from mymodule import say\_hi, \_\_version\_\_

say\_hi()

print('Version', \_\_version\_\_)

我们大都推荐最好去使用 import 语句，尽管这会使你的程序变得稍微长一些。

from mymodule import \* #避免使用最好

这将导入诸如 say\_hi 等所有公共名称，但不会导入 \_\_version\_\_ 名称，因为后者以双下划

线开头。

**Dir（）**

**内置的 dir() 函数能够返回由对象所定义的名称列表。 如果这一对象是一个模块，则该列**

**表会包括函数内所定义的函数、类与变量。**

**包**

**包是指一个包含模块与一个特殊的 \_\_init\_\_.py 文件的文件夹，后者向 Python 表明这一文**

**件夹是特别的，因为其包含了 Python 模块。**

- <some folder present in the sys.path>/

- world/

- \_\_init\_\_.py

- asia/

- \_\_init\_\_.py

- india/

- \_\_init\_\_.py

- foo.py

- africa/

- \_\_init\_\_.py

- madagascar/

- \_\_init\_\_.py

- bar.py

## 555数据结构

### 列表List

shoplist = ['apple', 'mango', 'carrot', 'banana']

一个类也可以带有方法（Method）:在这里要注意到我们通过使用点号的方法来访问对象。

一个类同样也可以具有字段（Field）: 它是只为该类定义且只为该类所用的变量

# This is my shopping list

shoplist = ['apple', 'mango', 'carrot', 'banana']

print('I have', len(shoplist), 'items to purchase.')

print('These items are:', end=' ')

for item in shoplist:

print(item, end=' ')

print('\nI also have to buy rice.')

shoplist.append('rice')

print('My shopping list is now', shoplist)

print('I will sort my list now')

shoplist.sort()

print('Sorted shopping list is', shoplist)

print('The first item I will buy is', shoplist[0])

olditem = shoplist[0]

del shoplist[0]

print('I bought the', olditem)

print('My shopping list is now', shoplist)

.append增加对象

.sort排序（列表会变化）

del shoplise[0]: 删除

len()

append与extend的区别:

>>> li = ['a', 'b', 'c']

>>> li.extend(['d', 'e', 'f'])

>>> li

['a', 'b', 'c', 'd', 'e', 'f']

>>> len(li)

6

>>> li[-1]

'f'

>>> li = ['a', 'b', 'c']

>>> li.append(['d', 'e', 'f'])

>>> li

['a', 'b', 'c', ['d', 'e', 'f']]

>>> len(li)

4

>>> li[-1]

['d', 'e', 'f']

### 元组Tuple

元组的一大特征类似于字符串，它们是不可变的，也就是说，你不能编辑或更改元组。

# 我会推荐你总是使用括号

# 来指明元组的开始与结束

# 尽管括号是一个可选选项。

# 明了胜过晦涩，显式优于隐式。

zoo = ('python', 'elephant', 'penguin')

print('Number of animals in the zoo is', len(zoo))

new\_zoo = 'monkey', 'camel', zoo

print('Number of cages in the new zoo is', len(new\_zoo))

print('All animals in new zoo are', new\_zoo)

print('Animals brought from old zoo are', new\_zoo[2])

print('Last animal brought from old zoo is', new\_zoo[2][2])

print('Number of animals in the new zoo is',

len(new\_zoo)-1+len(new\_zoo[2]))

new\_zoo[2][2] 来指定

new\_zoo 元组中的第三个项目中的第三个项

### 字典dict

键值（Keys）（即姓名）与值（Values）（即地址等详细信息）联立到一起。在这里要注意到键值必须是唯一的

在字典中，你可以通过使用符号构成 d = {key : value1 , key2 : value2} 这样的形式，来成

对地指定键值与值。在这里要注意到成对的键值与值之间使用冒号分隔，而每一对键值与值

则使用逗号进行区分，它们全都由一对花括号括起。

ab = {

'Swaroop': 'swaroop@swaroopch.com',

'Larry': 'larry@wall.org',

'Matsumoto': 'matz@ruby-lang.org',

'Spammer': 'spammer@hotmail.com'

}

print("Swaroop's address is", ab['Swaroop'])

# 删除一对键值—值配对

del ab['Spammer']

print('\nThere are {} contacts in the address-book\n'.format(len(ab)))

for name, address in ab.items():

print('Contact {} at {}'.format(name, address))

# 添加一对键值—值配对

ab['Guido'] = 'guido@python.org'

if 'Guido' in ab:

print("\nGuido's address is", ab['Guido'])

#### .items

字典的.items方法:将字典转换为列表，包含为元组[('B', 2), ('A', 1)]

#### Sorted (operator. Itemgetter)[[8]](#footnote-8)

sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), reverse=True)

将dict classCount的列表，按第二个值value排序，后返回列表。

### 序列sequence

列表、元组和字符串可以看作序列（Sequence）的某种表现形式，可是究竟什么是序列，它

又有什么特别之处？

序列的主要功能是资格测试（Membership Test）（也就是 in 与 not in 表达式）和索引

操作（Indexing Operations），它们能够允许我们直接获取序列中的特定项目。

# Slicing on a list #

print('Item 1 to 3 is', shoplist[1:3])

print('Item 2 to end is', shoplist[2:])

print('Item 1 to -1 is', shoplist[1:-1])

print('Item start to end is', shoplist[:])

# 从某一字符串中切片 #

print('characters 1 to 3 is', name[1:3])

print('characters 2 to end is', name[2:])

print('characters 1 to -1 is', name[1:-1])

print('characters start to end is', name[:])

索引操作也可以使用负数，在这种情况下，位置计数将从队列的末尾开始。因

此， shoplist[-1] 指的是序列的最后一个项目， shoplist[-2] 将获取序列中倒数第二个项

目。

如果第一位数字没有指定，Python 将会从序列的起始

处开始操作。如果第二个数字留空，Python 将会在序列的末尾结束操作。要注意的是切片操

作会在开始处返回 start，并在 end 前面的位置结束工作。**也就是说，序列切片将包括起始位**

**置，但不包括结束位置。**

**切片步长：**

>>> shoplist = ['apple', 'mango', 'carrot', 'banana']

>>> shoplist[::1]

['apple', 'mango', 'carrot', 'banana']

>>> shoplist[::2]

['apple', 'carrot']

>>> shoplist[::3]

['apple', 'banana']

>>> shoplist[::-1]

['banana', 'carrot', 'mango', 'apple']

### 集合Set

当集合中的项目存在与否比起次序或其出现次数更加重要时，我们就会使用集合。

>>> bri = set(['brazil', 'russia', 'india'])

>>> 'india' in bri

True

>>> 'usa' in bri

False

**>>> bric = bri.copy()**

**>>> bric.add('china')**

>>> bric.issuperset(bri)

True

>>> bri.remove('russia')

>>> bri & bric # OR bri.intersection(bric)

{'brazil', 'india'}

### 通过切片方式可以修订副本

print('Copy by making a full slice')

# 通过生成一份完整的切片制作一份列表的副本

mylist = shoplist[:]

# 删除第一个项目

del mylist[0]

print('shoplist is', shoplist)

print('mylist is', mylist)

# 注意到现在两份列表已出现不同

### 555Tips：

**遍历字典：**

For item in jj:

Print item, jj[item]

**列表推导式：**

【item\*4 for item in a if item>2】

## txt文件读取

### Open and readlines

# 打开文件

fo = open("runoob.txt", "r")

print "文件名为: ", fo.name

for line in fo.readlines(): #依次读取每行

line = line.strip() #去掉每行头尾空白

print "读取的数据为: %s" % (line)

# 关闭文件

fo.close()

## 后续：

1 Help(str)命令查询各种标准库

2 from引入各种第三方库

3练习List，Tuple，Dict，Set等用法

# Python包http://docs.scipy.org/doc/

## Package - NumPy

NumPy是Python的一个矩阵类型，提供了大量矩阵处理的函数

NumPy实际上包含了两种基本的数据类型：数组和矩阵

### Array

Array([1,1,1])

多维数组

Array([[1,2,3],[1,1,1]])

### Mat

#### 列表转换为Numpy矩阵

dataMat.append([1.0, float(lineArr[0]), float(lineArr[1])])

dataMatrix = mat(dataMatIn)

#### Numpy矩阵转变为ndarray数组

matrix.getA():

Return self as an ndarray object.

Equivalent to np.asarray(self).

mrray([1,1,1])

mm\*ss.T:转置后相乘

shape(mm):矩阵的维度

matlab中的.\*: multiply(mm,ss)

mm.sort()

mm.argsort(): 得到index

### python中numpy与matlab的对应关系

http://blog.csdn.net/ly\_ysys629/article/details/61422455

## Package – pandas

### 文件读取

read\_csv函数：

my\_data = pd.read\_csv('\watermelon.csv', encoding='gbk') #注意绝对和相对路径问题

## Package - Scikit-learn

1. https://zhuanlan.zhihu.com/p/25198543 [↑](#footnote-ref-1)
2. https://git-scm.com/book/zh/v2/Git-%E5%9F%BA%E7%A1%80-%E8%8E%B7%E5%8F%96-Git-%E4%BB%93%E5%BA%93 [↑](#footnote-ref-2)
3. http://blog.csdn.net/jjf09/article/details/76100665 [↑](#footnote-ref-3)
4. https://zhuanlan.zhihu.com/p/23344403 [↑](#footnote-ref-4)
5. http://datartisan.com/article/detail/159.html [↑](#footnote-ref-5)
6. <https://wiki.python.org/jython/InstallationInstructions>；http://search.maven.org/#search%7Cga%7C1%7Corg.python%2Bjython [↑](#footnote-ref-6)
7. http://www.jyni.org/#download [↑](#footnote-ref-7)
8. https://www.cnblogs.com/zhoufankui/p/6274172.html [↑](#footnote-ref-8)