# Python语言学习笔记

[Python语言学习笔记 1](#_Toc482781109)

[1. 三个Python专用的科学计算扩展库： 2](#_Toc482781110)

[2. 常用的编写python程序有两种方式：交互提示模式和文本编辑器。 2](#_Toc482781111)

[3. 模块： 2](#_Toc482781112)

[4. 在CMD命令行中打开python文件的方法 2](#_Toc482781113)

[5. 在CMD命令行直接运行python文件的方法 3](#_Toc482781114)

[6 Windows系统中CMD命令行使用的几个常用技巧 3](#_Toc482781115)

[7. 双击打开python程序后执行窗口一闪而过的情形 5](#_Toc482781116)

[8. 在C/C++程序中嵌入python 6](#_Toc482781117)

[9. python 自定义模块如何导入 6](#_Toc482781118)

[10. 常用的两个函数：dir()和help() 8](#_Toc482781119)

[11. python的标准第三方图像处理库PIL 9](#_Toc482781120)

[12. 在Python 3.5中使用OpenCV （安装） 10](#_Toc482781121)

[13. Numpy简介 13](#_Toc482781122)

[（1）Numpy是什么？ 13](#_Toc482781123)

[（2）Python中为什么要使用Numpy库？ 13](#_Toc482781124)

[（3）ndarray对象的创建 13](#_Toc482781125)

[（4）ndarray对象的基本操作 16](#_Toc482781126)

[（5）numpy的文件存取 17](#_Toc482781127)

[14 Spyder编辑器的简单使用 19](#_Toc482781128)

[15 Python的另一个图像处理库skimage 19](#_Toc482781129)

## 1. 三个Python专用的科学计算扩展库：

NumPy、SciPy、Matplotlib。分别为Python提供了数值计算、科学计算以及绘图功能。（由于我安装的是Anaconda，这些扩展库已经在安装时包含进去了，这也是用Anaconda进行python开发的一个好处，很多第三方扩展库都不用另外安装。要是安装的标准的Python IDLE，则可利用pip命令在线安装这些第三方扩展库，比如安装Numpy，只需要打开cmd命令行，然后直接输入“pip install numpy”即可）

## 2. 常用的编写python程序有两种方式：交互提示模式和文本编辑器。

其中交互提示模式下，每输入一个表达式并按下“回车”，该表达式的结果就自动计算出来；当输入的是一个复合语句（比如for循环语句）时，需要在语句结束后连续按下两次“回车”，才表示该复合语句的结束，这时该复合语句的结果才会被执行并显示。

在Python的交互式命令行写程序，好处是一下就能得到结果，坏处是没法保存，下次还想运行的时候，还得再敲一遍。所以，实际开发的时候，我们总是使用一个文本编辑器来写代码，写完了，保存为一个文件（保存时需要加上后缀名“.py”），这样，程序就可以反复运行了。

直接输入python进入交互模式，相当于启动了Python解释器，但是等待你一行一行地输入源代码，每输入一行就执行一行。

直接运行.py文件相当于启动了Python解释器，然后一次性把.py文件的源代码给执行了，你是没有机会输入源代码的。

用Python开发程序，完全可以一边在文本编辑器里写代码，一边开一个交互式命令窗口，在写代码的过程中，把部分代码粘到命令行去验证，事半功倍！

## 3. 模块：

一个包含了python语句的简单文本文件，其后缀名一般是“.py”，也就是一个python源文件；

**脚本：**可以直接运行的**模块**文件。

## 4. 在CMD命令行中打开python文件的方法

假设我要打开位于“F:\pythonTest”文件夹中的“script1.py”文件。

（1）首先按下快捷键“win+R”，打开“运行”窗口；然后输入“cmd”并回车，打开“命令提示符”窗口；

（2）然后转到“script1.py”文件所在文件夹：在命令行输入“F:”并回车，转到F盘；然后输入“CD pythonTest”并回车，转到“F:\pythonTest”文件夹。

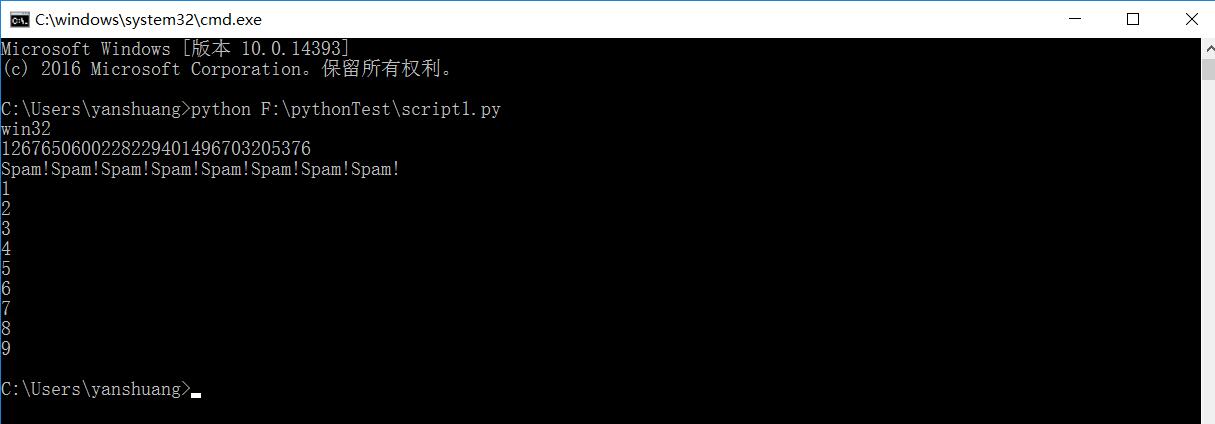
（3）直接输入“script1.py”并回车，命令行就会打开系统默认的打开“.py”文件的IDE并打开相应的python文件。如下图所示：



## 5. 在CMD命令行直接运行python文件的方法

同样，假设我要做在命令行中直接运行位于“F:\pythonTest”文件夹中的“script1.py”文件。

只需要在命令行中输入“python F:\pythonTest\script1.py”即可。如下图所示：



## 6 Windows系统中CMD命令行使用的几个常用技巧

利用Windows系统中的命令提示符窗口（cmd）可以做很多事情（目前还只是简单了解），比如之前python利用pip命令安装各种第三方库就是利用cmd命令行完成的（当然，也可以在Anaconda prompt中完成）。首先是如何打开cmd.exe。cmd程序在电脑上的位置一般是在“C:\Windows\System32”目录下。可以直接在该目录下找到cmd.exe双击打开。Win7以上系统可以使用快捷键“win+R”转到“运行”窗口，然后在该窗口中输入“cmd”，点击“确定”也可进入cmd窗口。下面整理一下最近学习到的cmd命令行使用的几个常用技巧：

（1）cmd命令行“属性”的使用

鼠标点击cmd命令行左上角，弹出菜单的最下面会有一个“Properties”选项，打开属性窗口如下图所示：



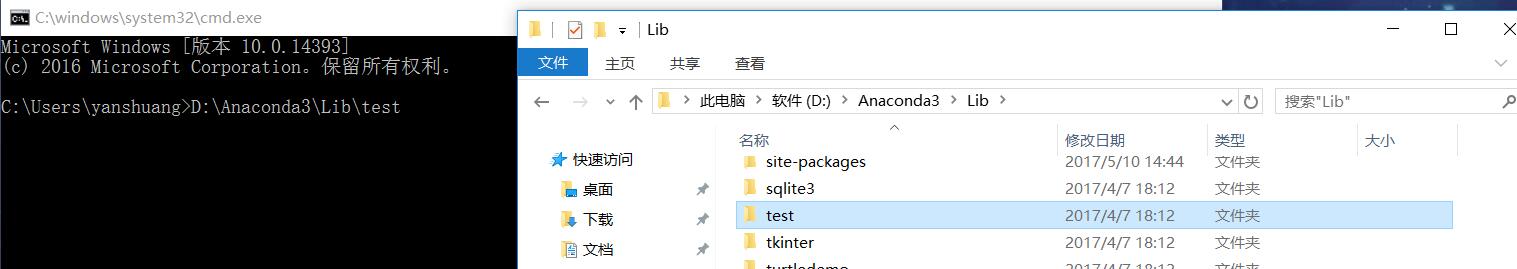
里面有关于cmd命令行的常规属性设置以及颜色、布局、字体等属性设置，可以根据自己的需求具体设置；

（2）使用拖拽方式快速输入文件路径

比如我需要CD到目录“D:\Anaconda3\Lib\test”，首先需要在cmd中输入“d:”，回车，转到D盘下，然后继续输入“cd D:\Anaconda3\Lib\test”，回车，就转到了“D:\Anaconda3\Lib\test”目录下了。这里输入路径名“D:\Anaconda3\Lib\test”的方式有3中：第一种方式是直接键盘输入，这种方式最慢而且容易输错；第二种方式是进入电脑中的该目录下，点击上面路径部分，右键复制/粘贴；图下图所示：

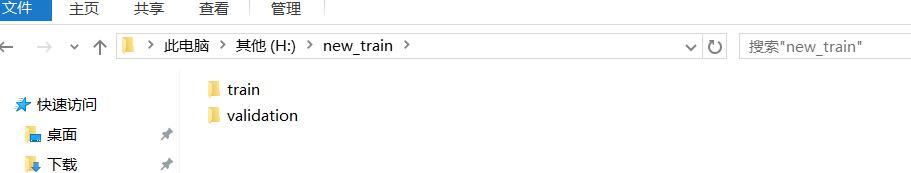


第三种方式是采用拖拽方式，同样进入该目录下，鼠标直接按住该文件夹，将其拖入cmd命令窗口，这种方式最方便，如下图所示：



（3）TAB键的自动填充功能

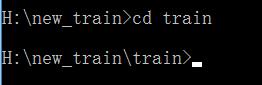
TAB键具有“自动填充”的功能，比图我们想要进入下图目录：



同时，如果我们已经CD转到了该目录的上一级目录：

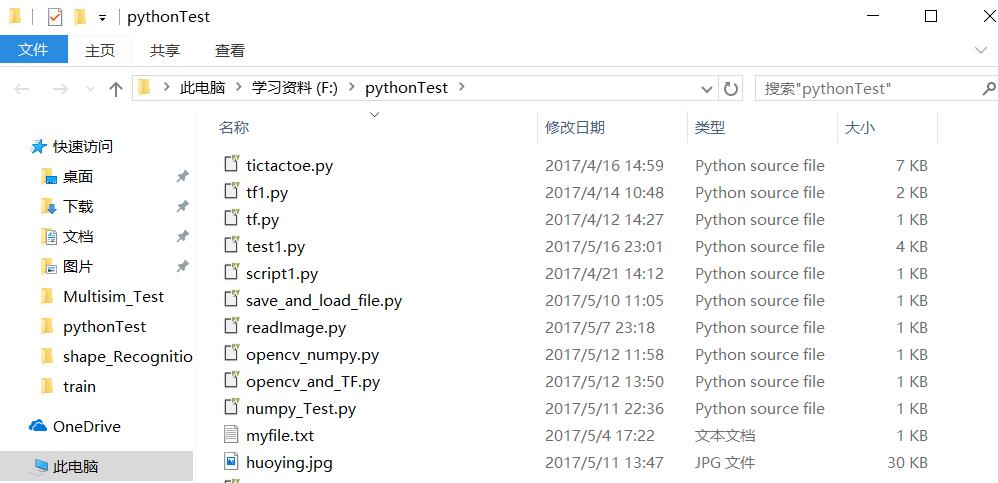


此时就可以用TAB键进行自动填充了，在上图中继续输入“cd ”，然后按TAB键，就会自动填充“new\_train”目录下的第一个子目录名称了，也就是“train”，继续按TAB键，自动填充内容就变成了“validation”目录，不断按TAB键，填充的内容就会在“new\_train”目录的子目录间不断切换，知道切换到你想要转到的目录，回车，如下图所示：



上图中“cd”后面的train是通过TAB键自动填充进去的。

当然，上面的TAB快捷例子不太好，另外，比如我们想在下面文件夹中的某个python脚本：



我们就可以先CD到“F:\pythonTest”，然后就可以使用TAB键自动填充/切换到我们想运行的python脚本名，然后回车，就可以在cmd中执行该程序了。

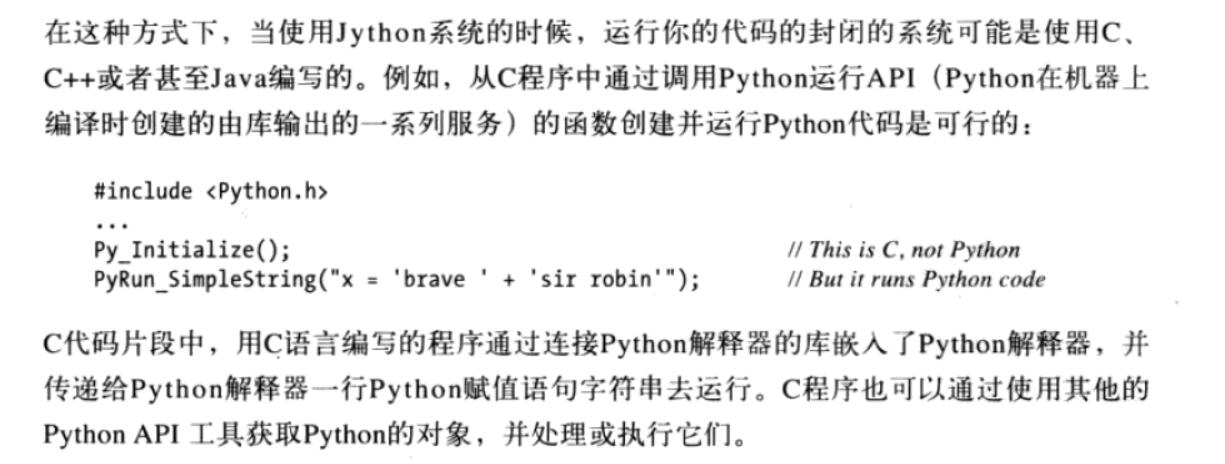
（4）快速查看/输入历史记录

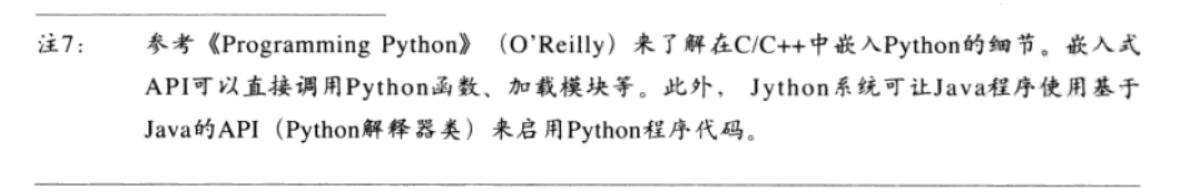
想要快速选择上一次输入的命令，只需要按下“↑”即可，随后下一个命令可以用“↓”来选择。完整的历史记录可以通过按下F7（或者CTRL+F7）来查看。

## 7. 双击打开python程序后执行窗口一闪而过的情形

如果电脑上只安装了python或Anaconda，当你双击打开一个python文件的时候，可能会看到执行窗口一闪而过，那是程序执行完之后自动退出的结果。如果想要看到程序的执行结果，一个比较方便的办法是：在程序代码的最后输入一条“input()”语句，我们知道，该语句是等待用户输入命令，这里就利用它来起到延迟功能。方便用户观察程序的执行结果。（但是如果程序有问题的话，输入“input()”就没用了，因为程序还没到最后就出错终止而退出了）

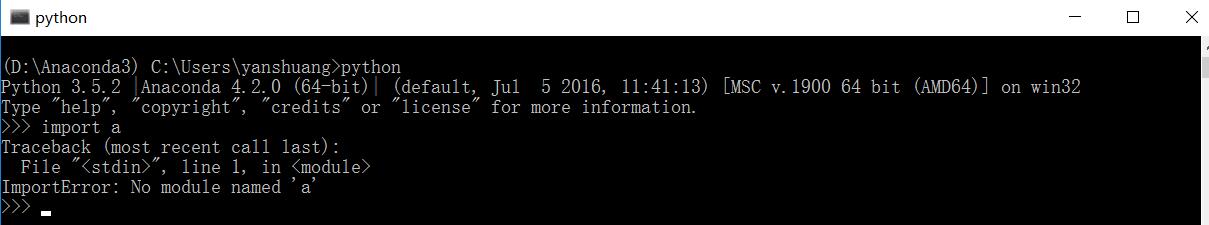
## 8. 在C/C++程序中嵌入python





## 9. python 自定义模块如何导入

我们自己定义的模块(比如a.py)，或者从网上下载的模块（比如用于下载MNIST训练集的模块input\_data,py），如果放在电脑中随便一个文件夹的话，直接在python交互提示窗口中输入“import input\_data”的话，程序会抛出错误，说是找不到该模块。如下图所示：



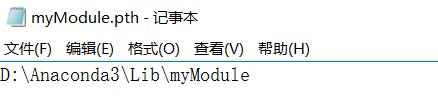
其原因大概是，Python 运行环境在查找库文件/模块时是对 sys.path 列表进行遍历。因为我们自己的模块所在路径还没有添加到sys.path 列表中，所以Python 运行环境找不到我们定义的模块。所以我们需要做的就是将自定义模块所在的文件夹路径添加到sys.path 列表中，我了解到的也不多，不过以下方法是可行的：

（1）为了便于区分，我们通常将自定义（或网上下载）的模块（或者package）和python自带的标准库中的模块/包分开管理。参照网上的做法，我是在路径“D:\Anaconda3\Lib”中建立了一个文件夹myModule，用于以后专门存放自定义的模块/包。比如，目前在该文件夹里只存放了一个用于下载MNIST训练集的模块“input\_data.py”，如下图所示：

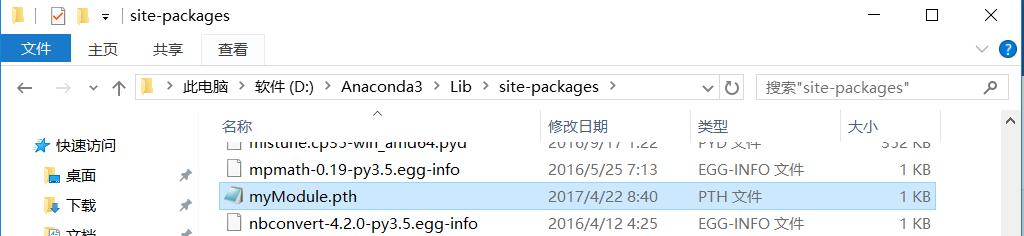


（2）然后，需要做的就是将自定义模块路径“D:\Anaconda3\Lib\myModule”添加到sys.path 列表中了。最简单的办法是用 .pth 文件来实现。Python 在遍历已知的库文件目录过程中，如果见到一个 .pth 文件，就会将文件中所记录的路径加入到 sys.path 设置中，这样 .pth 文件所指明的库也就可以被 Python 运行环境找到。

比如，其中一种做法是，建立一个名为myModule.pth的文件（如何建立？可以建立一个记事本文件，然后另存为…时，文件类型改为“所有文件”，文件名加上后缀.pth即可），该文件内容为路径“D:\Anaconda3\Lib\myModule”。如下图所示：

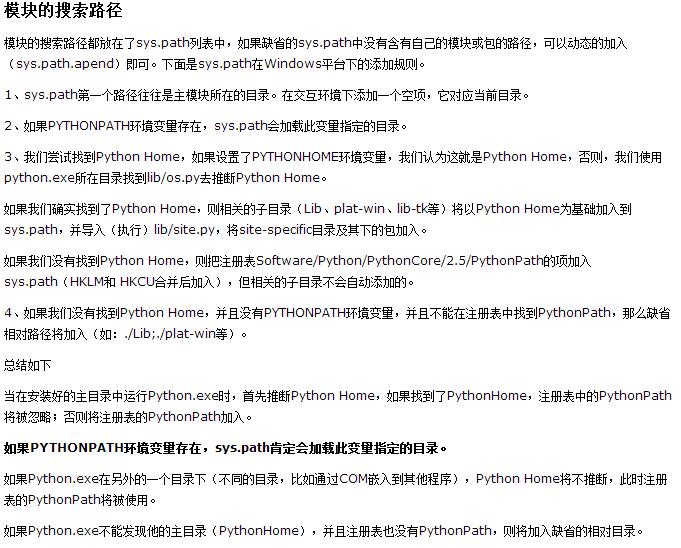


然后将myModule.pth文件放到“D:\Anaconda3\Lib\site-packages”文件夹中。如下图所示：



这样，在python交互提示窗口中输入“import input\_data”的话，Python 运行环境就能顺利找到该模块了，程序就不会报错。

（3）另外，网上还提供了了其他很多方法和原理解释，因为暂时只是为了模块能顺利导入就行，原理以后再深究，这里就把其他方法截图放在这里：



参考网址：

<http://www.cnblogs.com/essaylife/p/5144969.html>

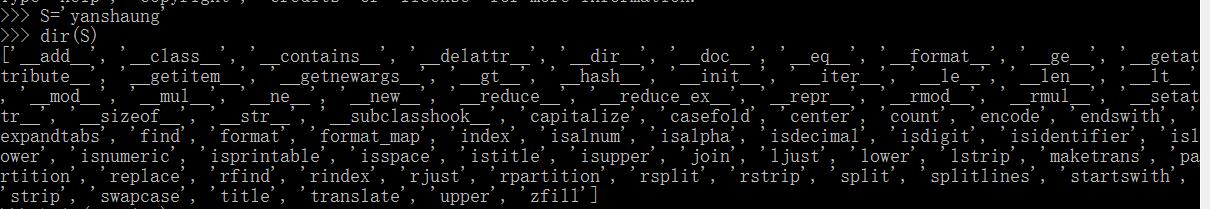
<http://blog.csdn.net/crylearner/article/details/7707446>

## 10. 常用的两个函数：dir()和help()

**dir()函数：**返回该对象类型的所有可用的方法。

比如：S=‘aass’。即S是一个字符串对象。

输入dir(S)，效果如下图：

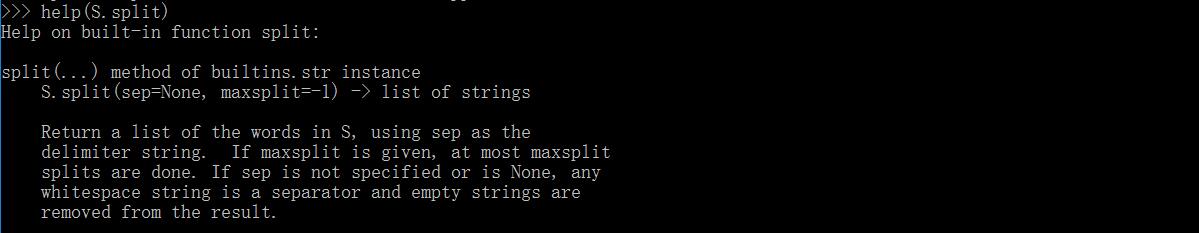


即显示了字符串对象能用的所有方法的名称；

help()函数：可以查看所有某种数据对象的某个方法的介绍。

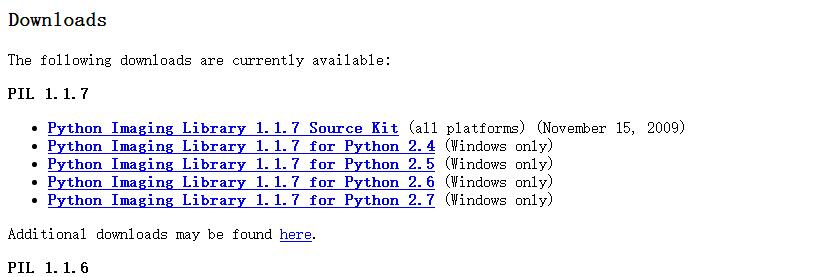
还是以字符串对象为例。假设已经创建了字符串对象S。

现在输入help(S.split)，则效果如下：



## 11. python的标准第三方图像处理库PIL

PIL (Python Imaging Library)是 Python 中最常用的图像处理库之一，可以在cmd直接使用pip命令进行在线安装。但是PIL在2009年推出1.1.7版本后就再没更新，而且当时的PIL最高支持的python版本是python2.7如下图所示：



所以，python3.x版本是无法在cmd中通过“pip install PIL”命令安装PIL的。因为没有支持Python3的PIL。

但是，官网推出了Pillow库用来替代PIL，可以说Pillow是PIL的2.0增强版本，是专门用于python3.x的图像处理库。下图是Pillow的官方介绍：



即Pillow是PIL的一个分支。

关于Pillow的安装，python3.x的用户直接在cmd中输入“pip install Pillow”，回车，就可以在线安装Pillow库了。

安装完成后，在python交互窗口输入“from PIL import Image”（导入Pillow中的Image模块），如果没报错，说明Pillow安装成功。（可以看到，虽然PIL改成了Pillow，但是导入时名称还是用的是“PIL”）。

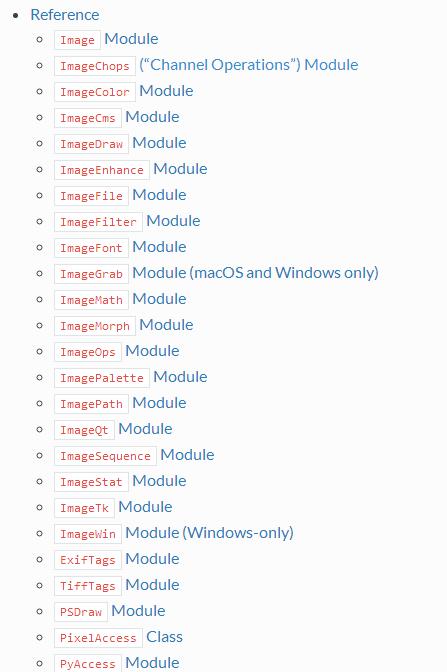
下面是调用Pillow进行简单图像读取并显示的代码示例：

>>>from PIL import Image

>>> a=Image.open('F:\\pythonTest\\22.jpg')

>>> a.rotate(45).show()

当然，Pillow库中除了Image模块，还有其他很多用于图像处理的模块，如下图所示：



关于学习PIL库（文中PIL和Pillow不分，因为都是同一个库，只是更新换代的区别）的参考网址：

<http://www.cnblogs.com/way_testlife/archive/2011/04/20/2022997.html>

<https://pillow.readthedocs.io/en/latest/index.html>

另外，Anconda已经在安装时自动安装了诸如Pillow、numpy、pip、scipy、matplotlib等比较常用的python第三方扩展库。所以安装Anconda进行python程序开发练习可以免去很多python扩展库的安装麻烦。

## 12. 在Python 3.5中使用OpenCV （安装）

之前安装的OpenCV库是针对VS2015中的C/C++程序的。由于要学习TensorFlow（下面简称TF）。打算将之前的基于opencv的双目视觉程序和TF机器学习算法结合起来应用。其中的问题是：TF目前只提供python语言的函数API，而前面写的opencv程序是c++程序。如何在同一个程序中使用TF和opencv函数？

还好的是，opencv作为一个十分强大的开源计算机视觉库，不仅提供了C/C++的函数接口，同时还提供的python语言的函数接口。因此，可以将之前的双目视觉程序用python语言重新编写。这样就可以同时导入opencv模块和TF模块，统一用python语言进行TF深度学习+opencv双目视觉开发了。

之前已经在python中成功安装了TF，由于TF for Windows目前只支持python3.5.x版本（我用的Anconda中的python版本是3.5.2），所以现在也需要python3.5.x中安装opencv。

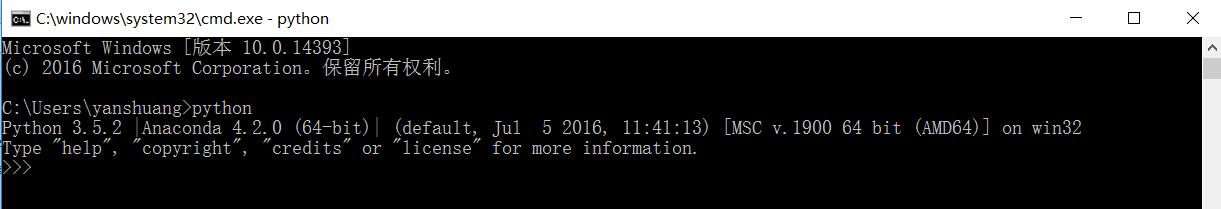
Opencv官网上介绍的“Install OpenCV-Python in Windows”的方法比较老。只提供了在python2.7中安装opencv3.0的方法。

参考网址：

<http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_setup/py_setup_in_windows/py_setup_in_windows.html#install-opencv-python-in-windows>

但是我们需要在python3.5中安装opencv。参考了网上的很多博文，终于发现一种十分便捷的方法，步骤如下：

（1）安装前请先确认自己电脑上已经安装了python3.5.x。检查方法：在cmd命令行中直接输入“python”，回车，如果已安装，就会出现python的版本信息，比如我的电脑上：



（2）在Python Extension Packages官网中下载opencv-for-python安装包（.whl文件）（该网站上提供了大量python语言的第三方扩展库的安装包下载，十分方便）。网址：

<http://www.lfd.uci.edu/~gohlke/pythonlibs/>

利用鼠标滑到网页中部左右，就能找到opencv安装包，如下图所示：



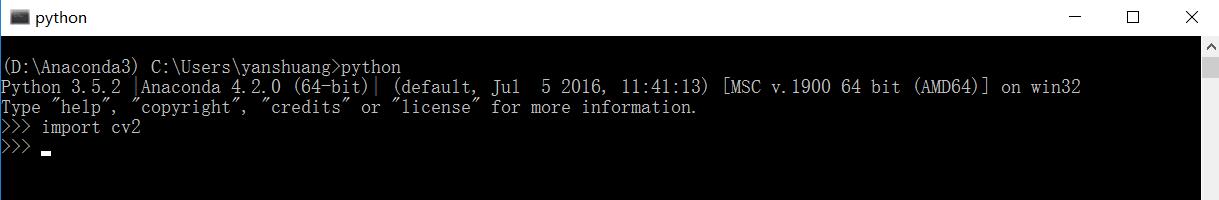
选择红色框线中的版本进行下载（cp35表示python3.5，win\_amd64表示电脑操作系统位数，我的是win10-64位，如果是32位操作系统就选择相邻上一个安装包；+contrib表示添加了contrib扩展库，据说很多opencv函数都需要依赖这个库，为了避免另外安装contrib库就选择这个版本吧）。

（3）将下载好了安装包放入电脑某个文件夹中，打开cmd命令行，CD到安装包所在文件夹，或者将安装包拷贝到cmd.exe的默认路径（比如我电脑上的默认路径是“C:\Users\yanshuang”），然后直接输入pip命令：

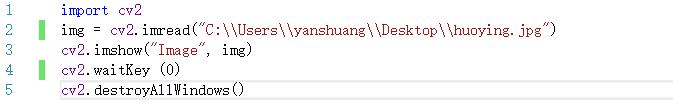
pip install opencv\_python-3.2.0+contrib-cp35-cp35m-win\_amd64.whl

顺利的话，几秒钟之内就能安装成功。

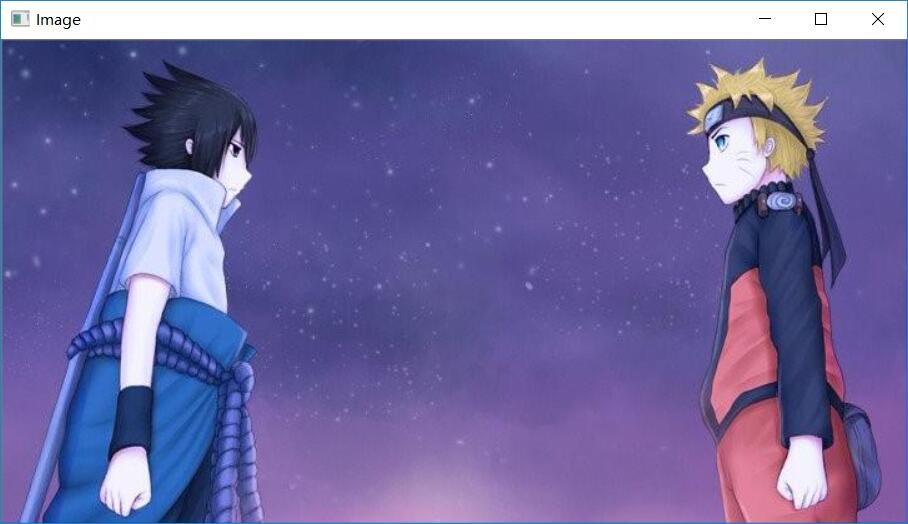
（4）测试，打开python交互窗口，输入“import cv2”。如果没有报错，表示opencv模块安装成功，如下图所示：



下面利用opencv库从文件夹载入图片并显示：



结果图：



安装参考网址：

<http://blog.csdn.net/qxconverse/article/details/59704906>

opencv for python的练习可参考网址：

<http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html>

<http://blog.csdn.net/sunny2038/article/category/904451>

国外的一本英文参考书：《OpenCV Computer Vision with Python》，免费下载网址（需要注册CSDN账号）：

<http://blog.csdn.net/sunny2038/article/details/8970266>

中文参考书：OpenCV官方教程中文版（For Python）.pdf。免费下载网址（需要注册CSDN账号）：

<http://download.csdn.net/detail/l1505624/9142455>

（在python-opencv中，利用cv2.imshow()函数显示图像时遇到了一个问题：当cv2.imread()载入一张比较大的图像时（我载入的是2.2M的图像），程序运行十分缓慢，而且加载了好久都没能准确加载显示图片，显示图像窗口一片黑，不清楚是什么原因，日后用到的时候再找原因吧；显示一张几百KB的图像就没问题）

## 13. Numpy简介

### （1）Numpy是什么？

很简单，Numpy是Python的一个用于科学计算的库，主要特点是提供了矩阵运算的大量函数，功能十分强大。其主要数据类型是多维数组：numpy.ndarray。

### （2）Python中为什么要使用Numpy库？

标准安装的Python中用列表(list)保存一组值，可以用来当作数组使用，不过由于列表的元素可以是任何对象，因此列表中所保存的是对象的指针。这样为了保存一个简单的[1,2,3]，需要有3个指针和三个整数对象。对于数值运算来说这种结构显然比较浪费内存和CPU计算时间。

此外Python还提供了一个array模块，array对象和列表不同，它直接保存数值，和C语言的一维数组比较类似。但是由于它不支持多维，也没有各种运算函数，因此也不适合做数值运算。

Numpy库的诞生弥补了这些不足，NumPy提供了两种基本的对象：**ndarray**（N-dimensional array object）和 **ufunc**（universal function object）。ndarray(下文统一称之为数组)是存储单一数据类型的多维数组，而ufunc则是能够对数组进行处理的函数。

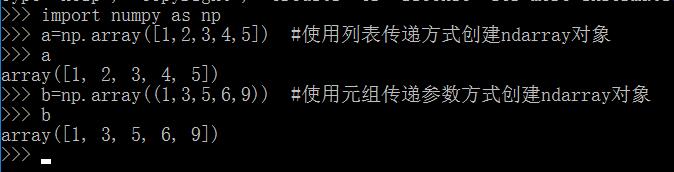
NumPy的主要对象是同种元素的多维数组。这是一个所有的元素都是一种类型、通过一个正整数元组索引的元素表格(通常是元素是数字)。在NumPy中维度(dimensions)叫做轴(axes)，轴的个数叫做秩(rank)。

### （3）ndarray对象的创建

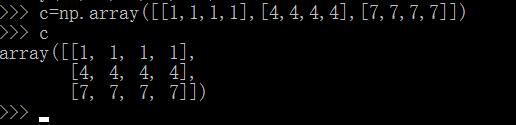
首先，需要使用import命令导入numpy模块：

import numpy as np

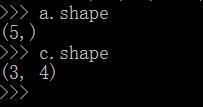
可以使用python的list（列表）或tuple（元组）来创建ndarray对象：



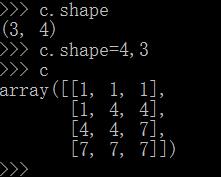
如果传递的是多层嵌套的序列，将创建多维数组：



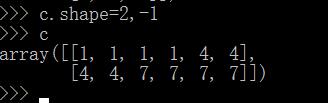
数组ndarray的大小可通过其shape属性获取：



其中，数组a的shape只有一个元素，因此它是一维数组。而数组c的shape有两个元素，因此它是二维数组，其中第0轴的长度为3，第1轴的长度为4。另外，在保证元素个数不变的情况下，数组的shape属性可以进行修改：

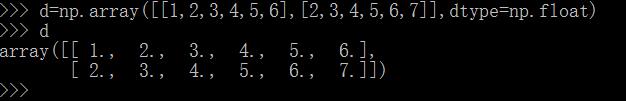


当某个轴的元素为-1时，将根据数组元素的个数自动计算此轴的长度，因此下面的程序将数组c的shape改为了(2,6)：

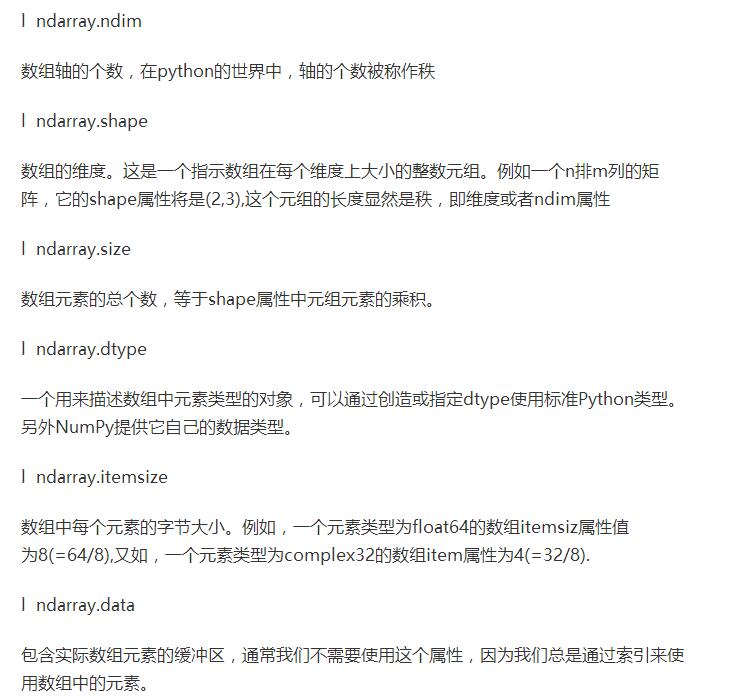


数组的元素类型可以通过dtype属性获得，同时也可以通过dtype参数在创建时指定元素类型（numpy.int32, numpy.int16, and numpy.float64等）：



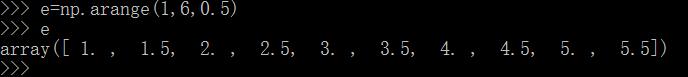


ndarray对象常用的其他属性如下图所示：

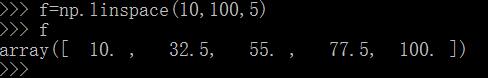


上面的例子都是先创建一个Python序列，然后通过array函数将其转换为数组，这样做显然效率不高。因此NumPy提供了很多专门用来创建数组的函数。

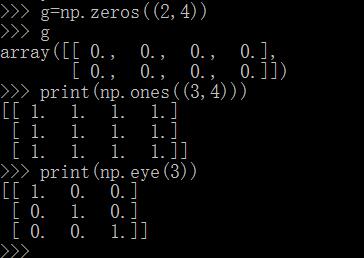
arange函数类似于python的range函数，通过指定开始值、终值和步长来创建一维数组，注意数组不包括终值:



linspace函数通过指定开始值、终值和元素个数来创建一维数组，可以通过endpoint关键字指定是否包括终值，缺省设置是包括终值:

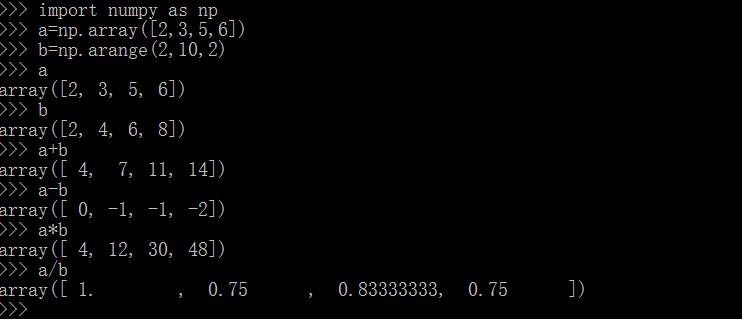


使用numpy.zeros，numpy.ones，numpy.eye等方法可以构造特定的矩阵：

****

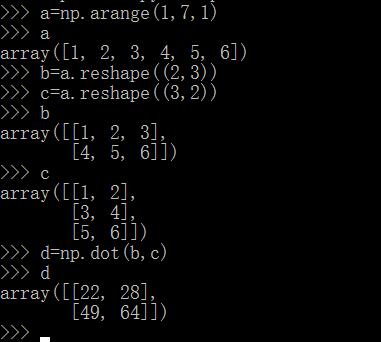
### （4）ndarray对象的基本操作

注意，ndarray对象的直接“加减乘除/+、-、\*、/”操作的对象是对应的每个元素，并不是常规的矩阵的加减乘除法则。如下图所示：



此时，进行加减乘除运算的两个ndarray对象的shape属性必须相同，否则会报错。

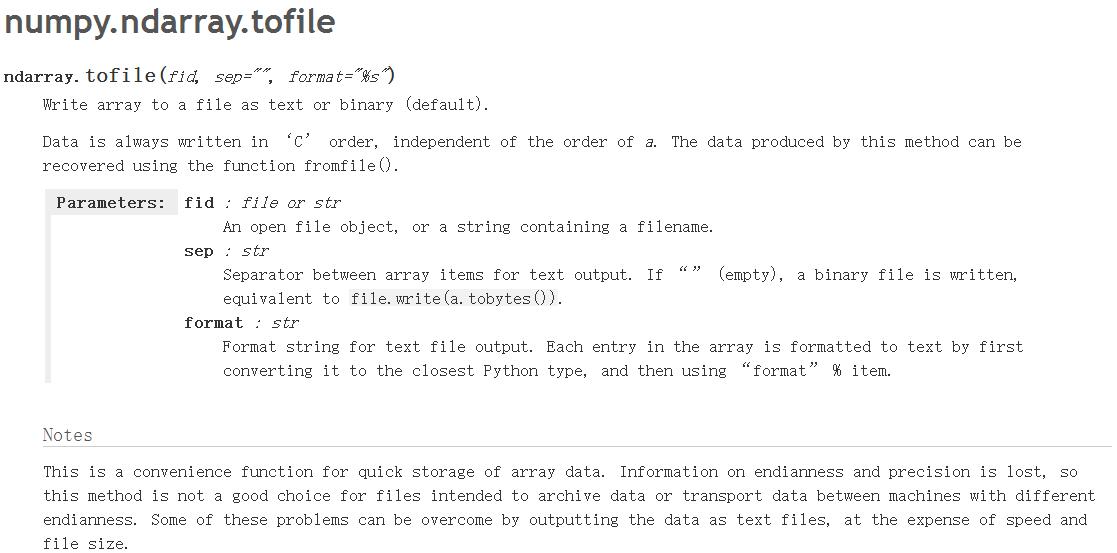
另外，如果想让两个数组进行矩阵相乘运算，可以用np.dot()函数：



### （5）numpy的文件存取

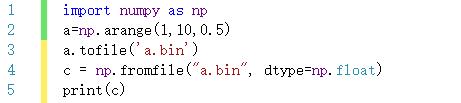
#### （5.1）使用tofile()和fromfile()进行文件存取

tofile()函数的使用说明如下：

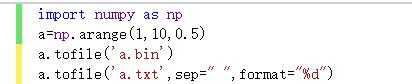


tofile()函数有3个参数，可以将ndarray对象保存为二进制文件（.bin）或者文本文件（.txt）。

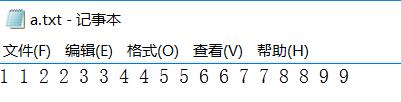
（a）当保存成二进制文件时，只用第一个参数；如下图所示：



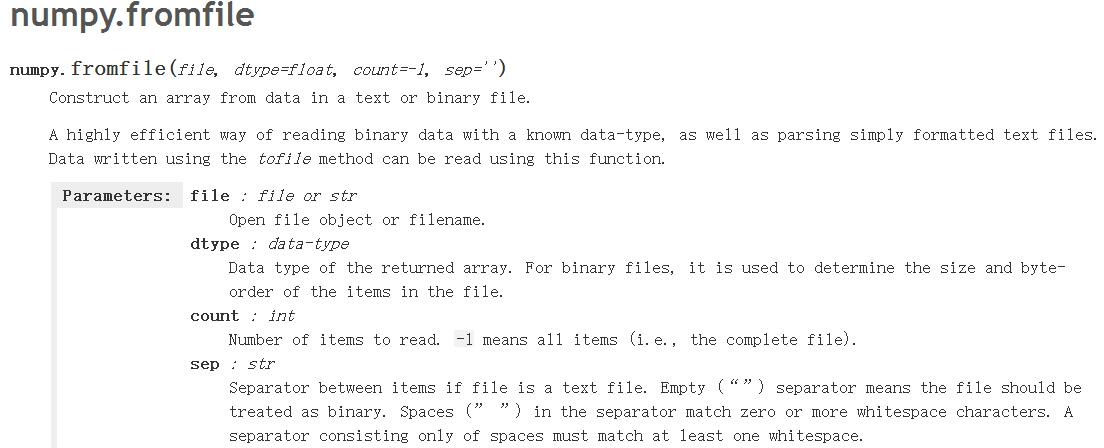
（b）当需要保存成文本文件（.txt）时，至少需要前面2个参数，其中第二个参数是指定保存文本中每个元素之间的间隔方式（” ”表示元素间以空格隔开，”,”表示元素间以逗号隔开）；第三个参数指定要保存文件中元素的数据类型，默认格式是”%s”，即字符串格式，还可以指定python的其他格式，比如”%d”——有符号整数(十进制)，”%f”——浮点数字(用小数点符号)。如下图所示：



得到的文本内容（因为保存格式是十进制整数，所以小数部分都没了）：

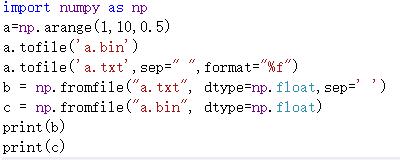


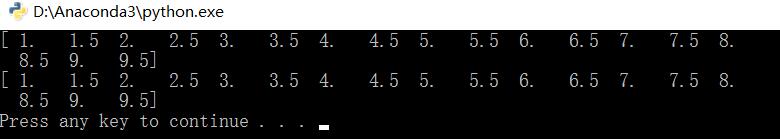
fromfile()函数的使用说明如下：



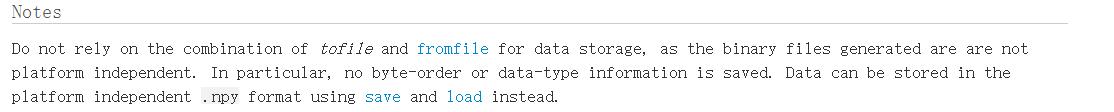
（a）当从二进制文件中载入ndarray对象时，一般只要前面2个参数，第一个是文件名，第二个是数据类型，默认是float，注意：fromfile()函数中的dtype必须和使用tofile()函数保存成二进制文件时ndarray对象的数据类型保持一致，否则会载入的数据会出错。

（b）当从文本文件中载入ndarray对象时，在（a）的基础上，必须加上最后一个参数“sep”，该参数同样指定的是元素间的间隔方式，注意必须和tofile()函数中的“sep”参数一致。代码如下：





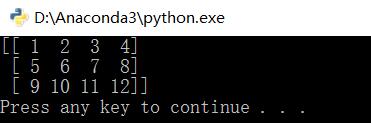
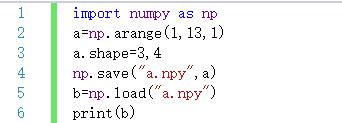
但是，官网看到下面的话：



上面说的是，不要依赖于tofile()函数和fromfile()函数进行数据存取，因为tofile()函数生成的二进制文件只适合与python平台，在其他平台上载入可能会出错。建议使用save()函数和load()函数进行文件存取。

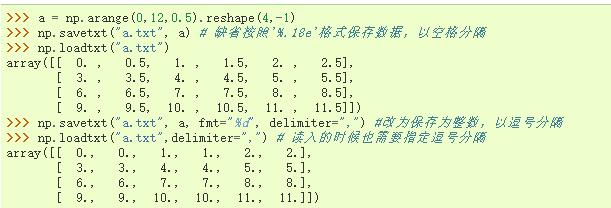
#### （5.2）使用save()和load()进行文件存取

numpy.load和numpy.save函数以NumPy专用的二进制类型保存数据，这两个函数会自动处理元素类型和shape等信息，使用它们读写数组就方便多了，但是numpy.save输出的文件很难和其它语言编写的程序读入：



#### （5.3）使用savetxt()和loadtxt()存取文件

使用numpy.savetxt和numpy.loadtxt可以读写1维和2维的数组：



## 14 Spyder编辑器的简单使用

## 15 Python的另一个图像处理库skimage

基于python脚本语言开发的数字图片处理包，其实很多，比如PIL,Pillow, opencv, scikit-image等。

对比这些包，PIL和Pillow只提供最基础的数字图像处理，功能有限；scikit-image是基于scipy的一款图像处理包，它将图片作为numpy数组进行处理，正好与matlab一样，因此，我们最终选择scikit-image进行数字图像处理。

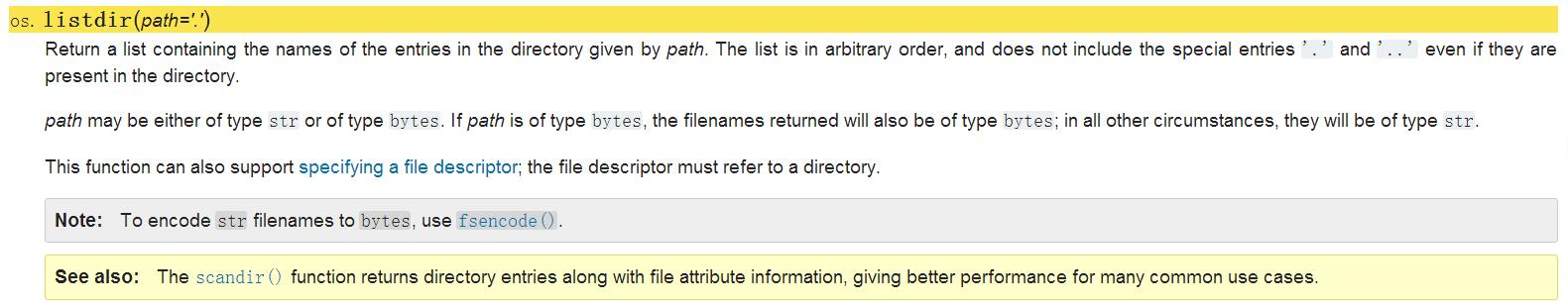
## 16 Python模块—os模块

os模块是Python标准库中的一个模块，它提供了通用的操作系统相关服务。本节主要记录一些os模块中的函数/方法的使用方法，遇到一个记录一个。

官方文档链接：

<https://docs.python.org/3.5/library/os.html>

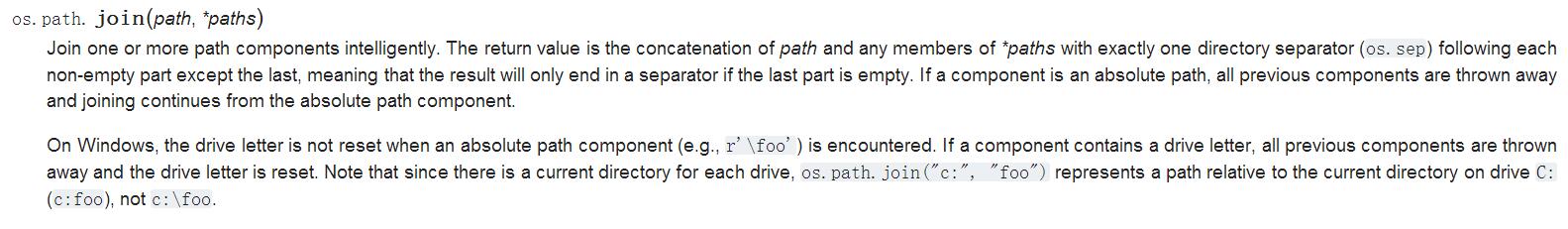
### os.listdir()



os.listdir()函数可以返回输入路径目录下所有文件名/子文件夹名组成的list列表。如果输入的路径是“str”类型，则返回的列表元素名也是“str”类型；但是需要注意，返回的列表元素的顺序是“arbitrary”任意的。

我们在利用tensorflow中就是使用了os.listdir()来获取训练图像文件夹中的所有图像名组成的列表，注意该列表的元素顺序是任意的。

### os.path.join()



os.path.join()就是起一个连接路径的作用，如下图所示：



## 17 在Python2中引入Python3中的特性

由于Python2和Python3在部分语法上的不兼容型性，在Python2中想要使用Python3中的一些特性都是使用“from \_\_future\_\_ import …”来实现。

### from \_\_future\_\_ import division

这句导入语句在一般的Github上下载的开源代码里都存在，其功能主要是为了解决Python2和Python3两个版本中的除法“/”规则不兼容问题。

在Python2中，如果分子分母都是整数，那么其结果也是整数（向下取整）。比如2/3=0。这和C/C++语法类似。

但是在Python3中，无论分子分母是整数还是浮点数，除法都是按浮点数规则相除，即2/3=1.5。

如果你的环境是Python2，但是想要使用Python3中的除法规则，则需要导入以上模块了。开源代码中一般都包含“from \_\_future\_\_ import division”的目的是为了增强代码的通用性，这样无论下载者使用Python2还是Python3都不会影响结果的准确性（虽然Python3环境不用添加这条语句）。

## 18 使用常见的图像处理库从文件读取图像时返回的shape及dtype

（1）cv2-cv2.imread()

返回值shape：[height, weight, channels]；

返回值dtype：uint8；

颜色通道顺序：BGR。

（2）skimage-skimage.io.imread()

返回值shape：[height, weight, channels]；

返回值dtype：uint8；

颜色通道顺序：RGB。

（3）PIL.Image-Image.open()

返回值shape：[height, weight, channels]；

返回值dtype：Image对象；需要使用img = np.asarray(img, dtype="uint8")，将其转成ndarray。

颜色通道顺序：RGB。