**[深度学习与计算机视觉系列(1)\_基础介绍](https://blog.csdn.net/yaoqiang2011/article/details/49876119)**

作者：[寒小阳](http://blog.csdn.net/han_xiaoyang?viewmode=contents" \t "_blank)   
时间：2015年11月。   
出处：[http://blog.csdn.net/han\_xiaoyang/article/details/49876119](http://blog.csdn.net/han_xiaoyang/article/details/49876119" \t "_blank)   
声明：版权所有，转载请注明出处，谢谢。

## 1.背景

[计算机视觉](http://library.kiwix.org/wikipedia_zh_all/A/html/%E8%AE%A1/%E7%AE%97/%E6%9C%BA/%E8%A7%86/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%A7%86%E8%A7%89.html)/[computer vision](https://en.wikipedia.org/wiki/Computer_vision)是一个火了N年的topic。持续化升温的原因也非常简单：在搜索/影像内容理解/医学应用/地图识别等等领域应用太多，大家都有一个愿景『让计算机能够像人一样去”看”一张图片，甚至”读懂”一张图片』。

有几个比较重要的计算机视觉任务，比如图片的分类,物体识别，物体定位于检测等等。而近年来的[神经网络/深度学习](https://en.wikipedia.org/wiki/Deep_learning" \t "_blank)使得上述任务的准确度有了非常大的提升。加之最近做了几个不大不小的计算机视觉上的项目，爱凑热闹的博主自然不打算放过此领域，也边学边做点笔记总结，写点东西，写的不正确的地方，欢迎大家提出和指正。

## 2.基础知识

为了简单易读易懂，这个课程笔记系列中绝大多数的代码都使用python完成。这里稍微介绍一下python和Numpy/Scipy(**python中的科学计算包**)的一些基础。

### 2.1 python基础

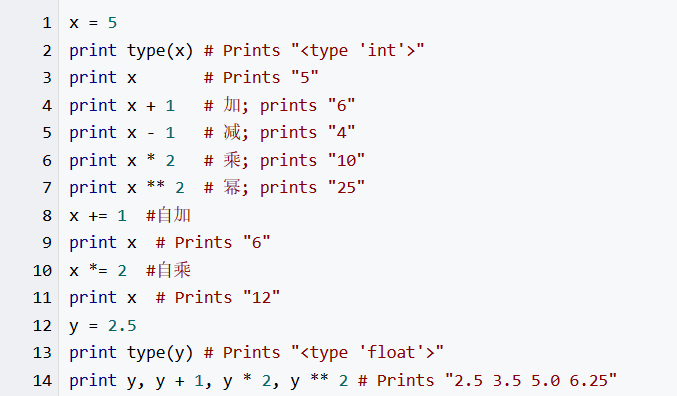
python是一种长得像伪代码，具备高可读性的编程语言。   
优点挺多：可读性相当好，写起来也简单，所想立马可以转为实现代码，且社区即为活跃，可用的package相当多；缺点：效率一般。

#### 2.1.1 基本数据类型

最常用的有数值型(Numbers),布尔型(Booleans)和字符串(String)三种。

* 数值型(Numbers)

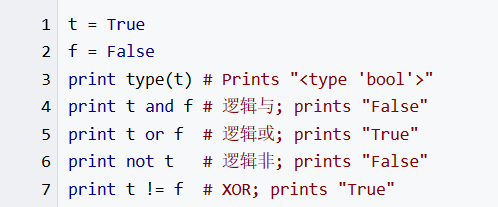
可进行简单的运算，如下：



PS：python中没有x++ 和 x– 操作

* 布尔型(Booleans)

包含True False和常见的与或非操作

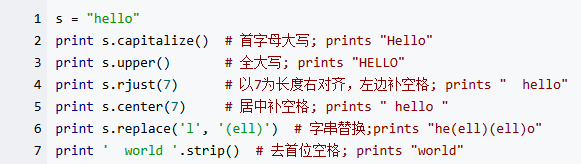


* 字符串型(String)

字符串可以用单引号/双引号/三引号声明



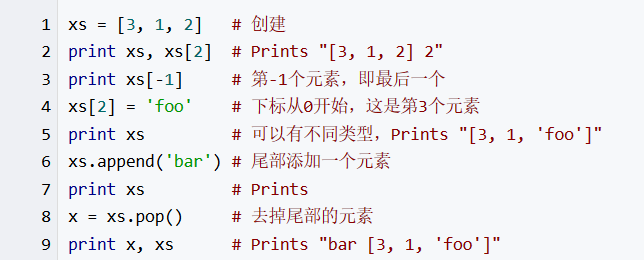
字符串对象有很有有用的函数：



#### 2.1.2 基本容器

* **列表/List**

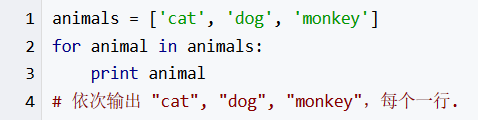
和数组类似的一个东东，不过可以包含不同类型的元素，同时大小也是可以调整的。



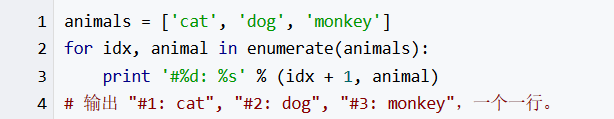
列表最常用的操作有：   
**切片/slicing**   
即取子序列/一部分元素，如下：



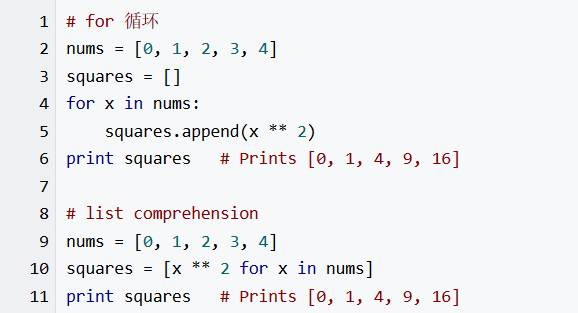
**循环/loops**   
即遍历整个list，做一些操作，如下：



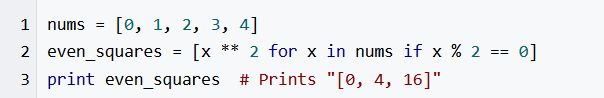
可以用enumerate取出元素的同时带出下标



**List comprehension**   
这个相当相当相当有用，在很长的list生成过程中，效率完胜for循环：



你猜怎么着，list comprehension也是可以加多重条件的：

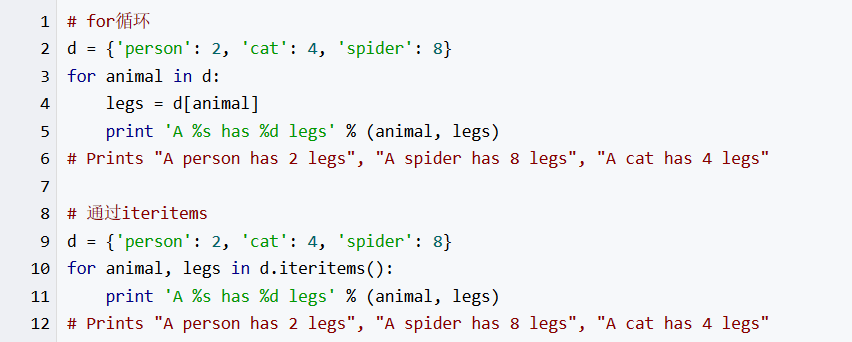


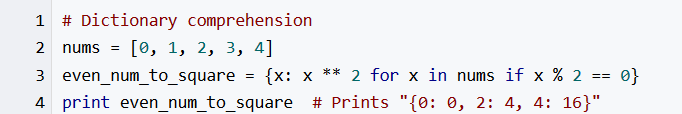
**字典/Dict**   
和Java中的Map一样的东东，用于存储key-value对：



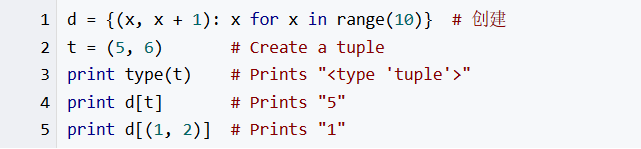
对应list的那些操作，你在dict里面也能找得到：

**循环/loops**



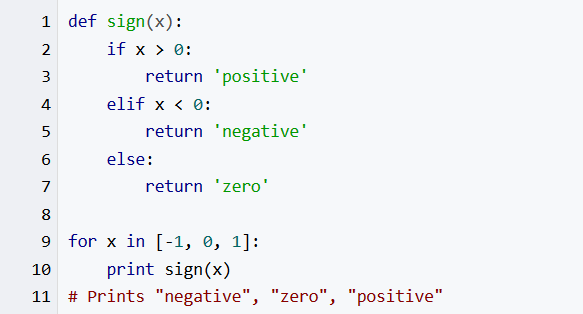


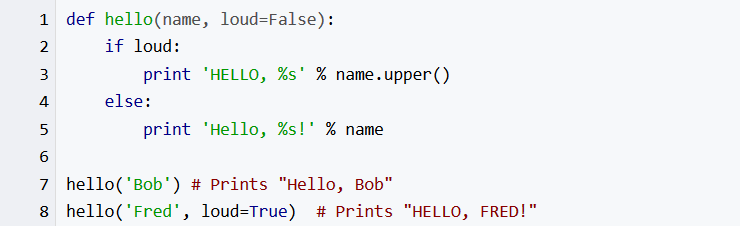
**元组/turple**   
本质上说，还是一个list，只不过里面的每个元素都是一个两元组对。



#### 2.1.3 函数

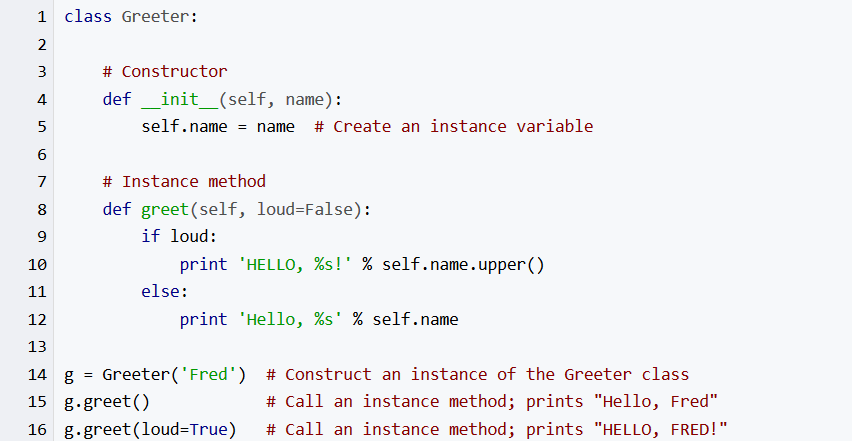
用def可以定义一个函数：





* 类/Class

python里面的类定义非常的直接和简洁：

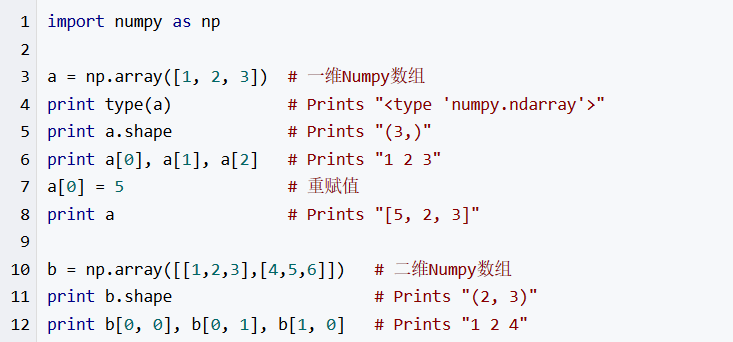


### 2.2.NumPy基础

NumPy是Python的科学计算的一个核心库。它提供了一个高性能的多维数组(矩阵)对象，可以完成在其之上的很多操作。很多机器学习中的计算问题，把数据vectorize之后可以进行非常高效的运算。

#### 2.2.1 数组

一个NumPy数组是一些类型相同的元素组成的类矩阵数据。用list或者层叠的list可以初始化：

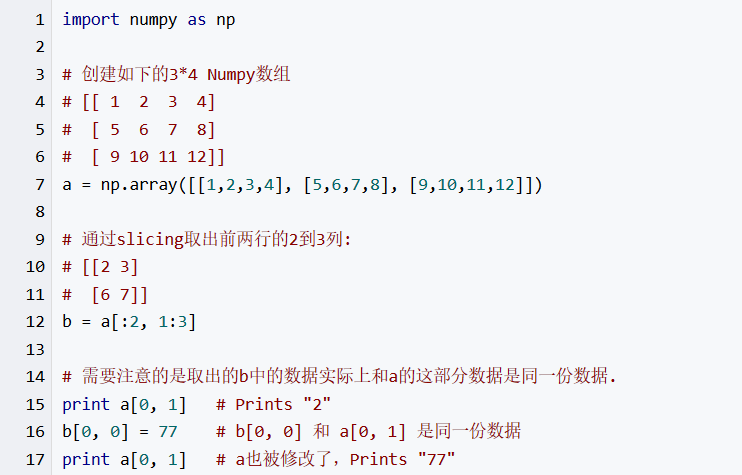


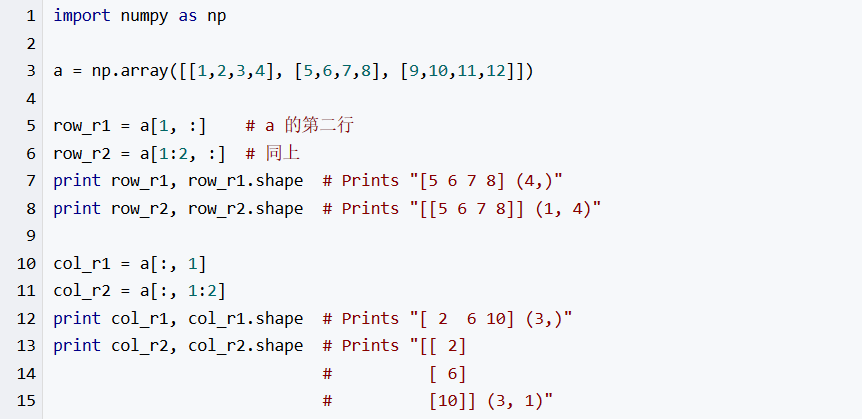
生成一些特殊的Numpy数组(矩阵)时，我们有特定的函数可以调用：



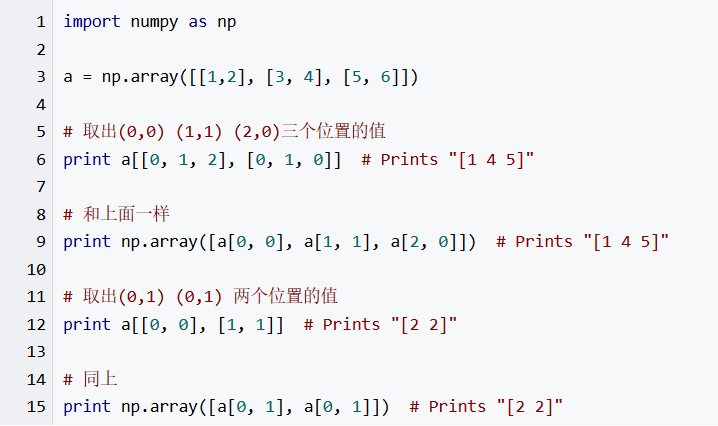
#### 2.2.2 Numpy数组索引与取值

可以通过像list一样的分片/slicing操作取出需要的数值部分。

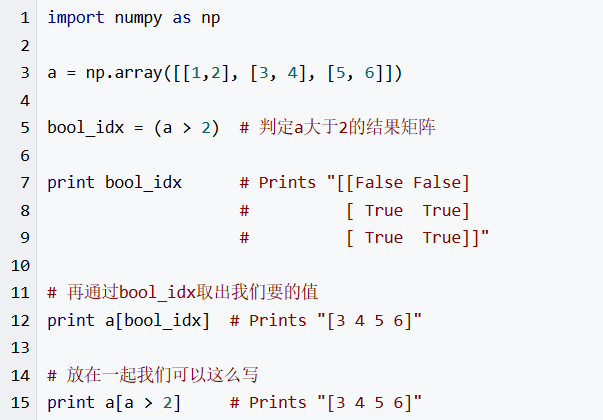




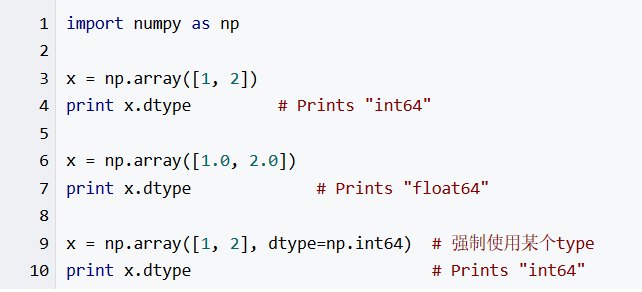
还可以这么着取：



我们还可以通过条件得到bool型的Numpy数组结果，再通过这个数组取出符合条件的值，如下：

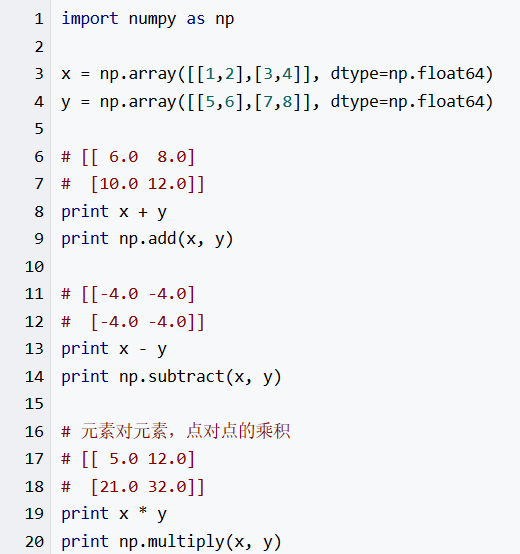


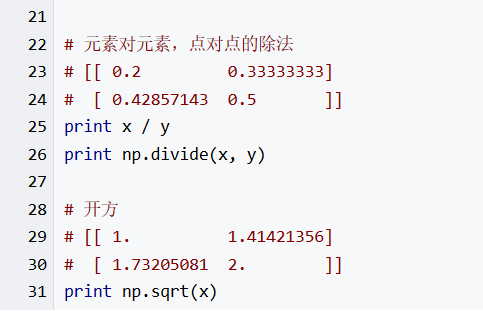
#### Numpy数组的类型



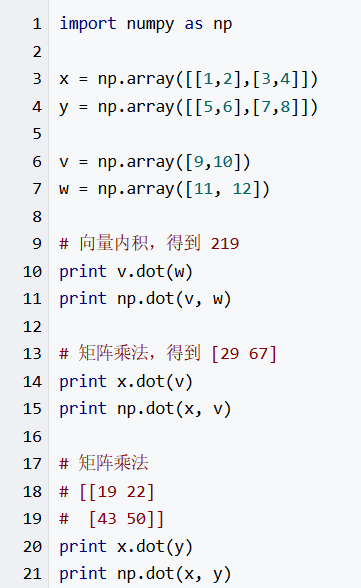
#### 2.2.3 Numpy数组的运算

矩阵的加减开方和(元素对元素)乘除如下：

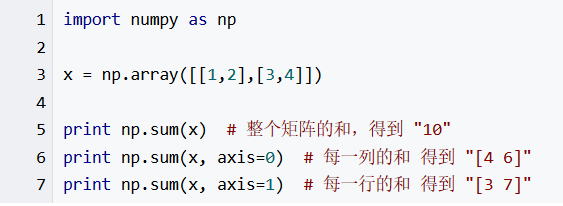




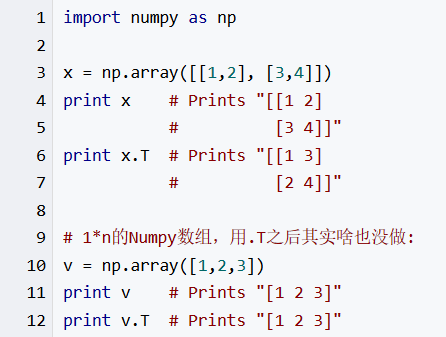
矩阵的内积是通过下列方法计算的：



特别特别有用的一个操作是，sum/求和(对某个维度)：



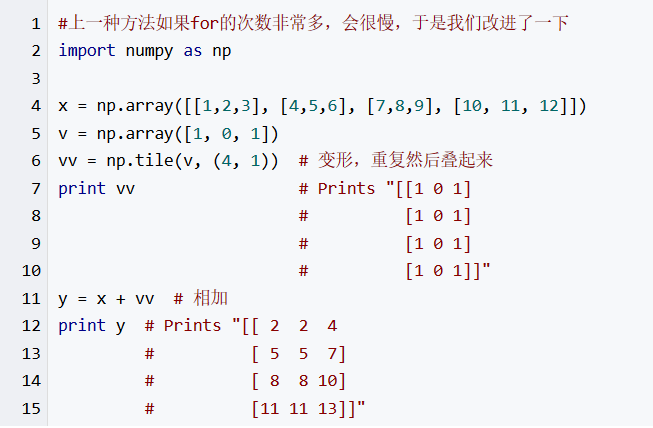
还有一个经常会用到操作是矩阵的转置，在Numpy数组里用.T实现：

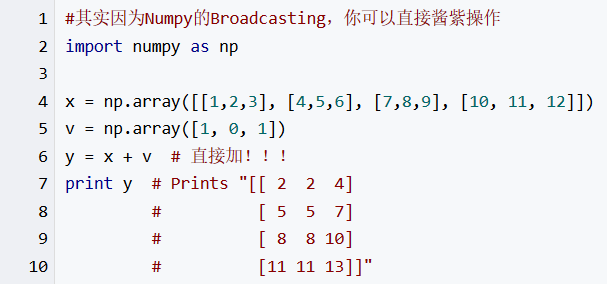


#### 2.2.4 Broadcasting

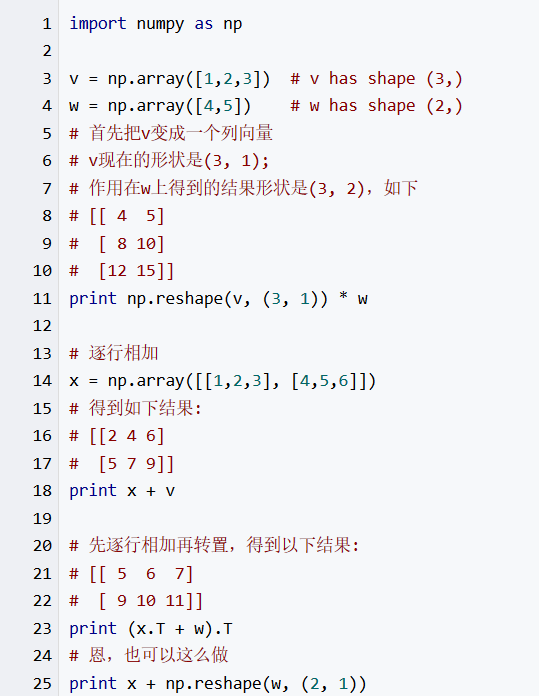
Numpy还有一个非常牛逼的机制，你想想，如果你现在有一大一小俩矩阵，你想使用小矩阵在大矩阵上做多次操作。额，举个例子好了，假如你想将一个1\*n的矩阵，加到m\*n的矩阵的每一行上：







更多Broadcasting的例子请看下面：

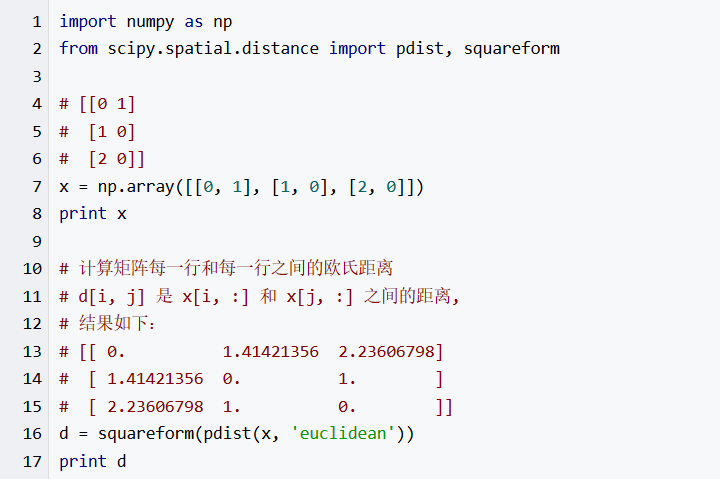


### 2.3 SciPy

Numpy提供了一个非常方便操作和计算的高维向量对象，并提供基本的操作方法，而Scipy是在Numpy的基础上，提供很多很多的函数和方法去直接完成你需要的矩阵操作。有兴趣可以浏览[Scipy方法索引](http://docs.scipy.org/doc/scipy/reference/index.html)查看具体的方法，函数略多，要都记下来有点困难，随用随查吧。

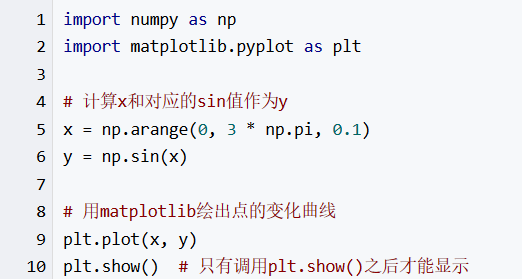
#### 向量距离计算

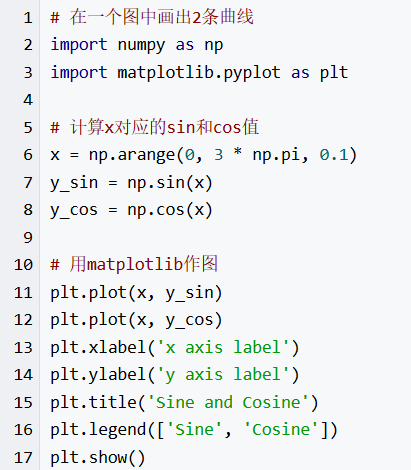
需要特别拎出来说一下的是，向量之间的距离计算，这个Scipy提供了很好的接口[scipy.spatial.distance.pdist](http://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html" \l "scipy.spatial.distance.pdist" \t "_blank)：



### 2.4 Matplotlib

这是python中的一个作图工具包。如果你熟悉matlab的语法的话，应该会用得挺顺手。可以通过[matplotlib.pyplot.plot](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.plot" \t "_blank)了解更多绘图相关的设置和参数。

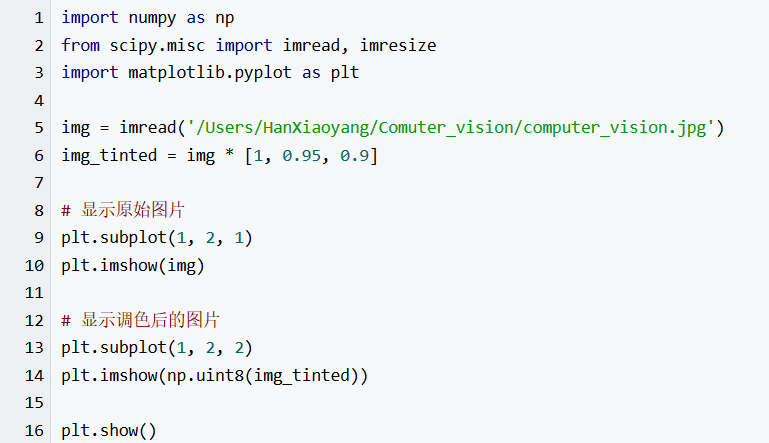






### 2.5 简单图片读写

可以使用imshow来显示图片。



## 参考资料与原文

[cs231n python/Numpy指南](http://cs231n.github.io/python-numpy-tutorial/)