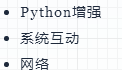
对Python标准库的认识

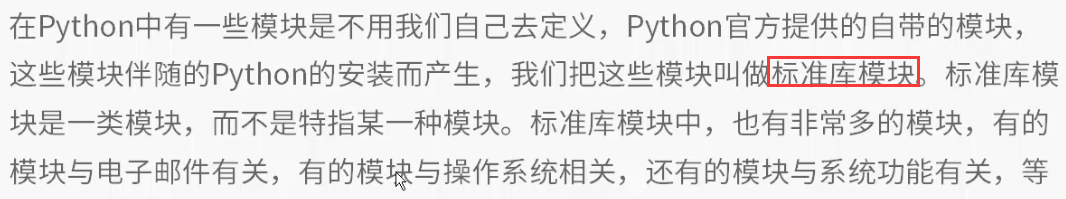
# 对Python标准库的认识

正如那句 **Python 社区**中很有名的话所说的：“**battery included**”，Python 的一大好处在于它有一套很有用的标准库(**standard library**)。标准库是随着 Python 一起安装在你的电脑中的，是 Python 的一部分 (当然也有特殊情况。有些场合会因为系统安全性的要求，不使用全部的标准库，比如说Google App Engine)。

**利用已有的类(class)和函数(function)进行开发，可以省去你从头写所有程序的苦恼**。这些标准库就是盖房子已经烧好的砖，要比你自己去烧砖来得便捷得多。

我将根据我个人的使用经验中，先挑选出**标准库下面三个方面的包(package)**介绍，以说明标准库的强大功能：





# Python增强

Python 自身的已有的一些功能可以随着**标准库的使用**而得到增强。

## 文字处理

**Python的string类提供了对字符串进行处理的方法**。但Python并不止步于此。**通过标准库中的re包，Python实现了对正则表达式(regular expression)的支持。**Python的正则表达式可以和Perl以及Linux bash的正则表达相媲美。**正则表达式通过自定义的模板在文本中搜索或替换符合该模板的字符串**。比如你可以搜索一个文本中所有的数字。**正则表达式的关键在于根据自己的需要构成模板**。

此外，Python 标准库还为字符串的输出提供更加丰富的格式， 比如： **string包**，**textwrap包**。

## 数据对象

我们之前的快速教程介绍了**表(list), 字典(dictionary)**等数据对象。它们各自有不同的特征，适用于不同场合的对数据的组织和管理。Python的标准库定义了更多的数据对象，比如说**数组(array)，队列(Queue)**。这些数据对象也分别有各自的特点和功能。**一个熟悉数据结构(data structure)的Python用户可以在这些包中找到自己需要的数据结构。**

此外，我们也会经常使**用copy包**，以复制对象。

## 日期和时间

日期和时间的管理并不复杂，但容易犯错。Python的标准库中对日期和时间的管理颇为完善（**利用time包管理时间，利用datetime包管理日期和时间**），你不仅可以进行**日期时间的查询和变换**（比如：2012年7月18日对应的是星期几），还可以**对日期时间进行运算**(比如2000.1.1 13:00的378小时之后是什么日期，什么时间)。通过这些标准库，还可以根据需要控制日期时间输出的文本格式(比如：输出’2012-7-18‘还是'18 Jul 2012')

## 数学运算

标准库中，Python定义了一些新的数字类型(**decimal包, fractions包**), 以弥补之前的数字类型(integer, float)可能的不足。**标准库还包含了random包，用于处理随机数相关的功能（产生随机数，随机取样等）**。math包补充了一些重要的数学常数和数学函数，比如pi，三角函数等等。

**尽管numpy并不是标准库中的包，但它的数组运算的良好支持，让它在基于Python的科研和计算方面得到相当广泛的应用，可以适当关注**。

## 存储

之前我们的快速教程中，只提及了**文本的输入和输出**。实际上，Python可以输入或输出任意的对象。这些对象可以通过标准库中的pickle包转换成为二进制格式(binary)，然后存储于文件之中，也可以反向从二进制文件中读取对象。

此外，标准库中还支持基本的数据库功能(sqlite3包)**。XML和csv格式的文件也有相应的处理包**。

# 系统互动

**系统互动**，主要指Python和操作系统(operate system)、文件系统(file system)的互动。Python可以实现一个操作系统的许多功能。**它能够像bash脚本那样管理操作系统，这也是Python有时被成为脚本语言的原因**。

## Python运行控制

**sys包被用于管理Python自身的运行环境**。Python是一个**解释器(interpreter)**, 也是一个运行在操作系统上的程序。我们可以用sys包来控制这一程序运行的许多参数，比如说Python运行所能占据的内存和CPU， Python所要扫描的路径等。另一个重要功能是和Python自己的命令行互动，从命令行读取命令和参数。

## 操作系统

如果说Python构成了一个小的世界，那么操作系统就是包围这个小世界的大世界。Python与操作系统的互动可以让Python在自己的小世界里管理整个大世界。

**os包**是Python与操作系统的接口。我们可以用os包来实现操作系统的许多功能，比如管理系统进程，改变当前路径(相当于’cd‘)，改变文件权限等，建立。但要注意，os包是建立在操作系统的平台上的，许多功能在Windows系统上是无法实现的。另外，在使用os包中，要注意其中的有些功能已经被其他的包取代。

我们通过文件系统来管理磁盘上储存的文件。查找、删除，复制文件，以及列出文件列表等都是常见的文件操作。这些功能经常可以在操作系统中看到（比如ls, mv, cp等Linux命令），但现在可以通过Python标准库中的glob包、shutil包、os.path包、以及os包的一些函数等，在Python内部实现。

**subprocess包**被用于执行外部命令，其功能相当于我们在操作系统的命令行中输入命令以执行，比如常见的系统命令'ls'或者'cd'，还可以是任意可以在命令行中执行的程序。

## 线程与进程

Python支持**多线程(threading包)**运行和**多进程(multiprocessing包**)运行。通过多线程和多进程，可以提高系统资源的利用率，提高计算机的处理速度。Python在这些包中，附带有相关的通信和内存管理工具。此外，Python还支持类似于UNIX的signal系统，以实现进程之间的粗糙的信号通信。

# 网络

现在，**网络功能的强弱很大程度上决定了一个语言的成功与否**。从Ruby, JavaScript, php身上都可以感受到这一点。Python的标准库对互联网开发的支持并不充分，这也是Django等基于Python的项目的出发点: **增强Python在网络方面的应用功能**。这些项目取得了很大的成功，也是许多人愿意来学习Python的一大原因。但应注意到，这些基于Python的项目也是建立在Python标准库的基础上的。

## 基于socket层的网络应用

**socket是网络可编程部分的底层**。通过**socket包**，我们可以直接管理socket，比如说将socket赋予给某个端口(port)，连接远程端口，以及通过连接传输数据。我们也可以利用SocketServer包更方便地建立服务器。

通过与多线程和多进程配合，建立多线程或者多进程的服务器，可以有效提高服务器的工作能力。此外，通过**asyncore包**实现异步处理，也是改善服务器性能的一个方案。

## 互联网应用

在实际应用中，网络的很多底层细节（比如socket）都是被高层的协议隐藏起来的。**建立在socket之上的http协议实际上更容易也更经常被使用**。http通过request/responce的模式建立连接并进行通信，其信息内容也更容易理解。**Python标准库中有http的服务器端和客户端的应用支持(BaseHTTPServer包; urllib包, urllib2包), 并且可以通过urlparse包对URL（URL实际上说明了网络资源所在的位置）进行理解和操作**。

这些内容可以说非常粗糙，只希望能为大家提供一个了解标准库的入口。欢迎大家一起分享标准库的使用经验。

# sys模块

import sys  
print(sys.version)  
#3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)]  
print(sys.copyright)#Copyright (c) 2000 BeOpen.com.All Rights Reserved.  
print(sys.flags)#sys.flags(debug=0, inspect=0, interactive=0  
print(sys.executable)#D:\Python\Python36\python.exe  
print(sys.getwindowsversion())  
#sys.getwindowsversion(major=10, minor=0, build=14393, platform=2, service\_pack='')

还有很多，用到的时候可以查官方文档。