讲义 7：python 语言进阶

没有见过有语言基础的同学 step by step 学习 python 这类语言的。本课程将继续介绍 python 的一些基础知识，以帮助你能够快速源代码中学习编程技巧。并建立高质量的程序。

课程目标：
1）掌握 python 基本数据结构，使用 tornado 建立一个小网站。

使用工具：
Notepad++ or sublime;
chrome browser;
python2.7; Tornado 3.1;

课程资源与要点：
1）python 官方教程（需要 C 基础）：***python tutorial***
http://docs.python.org/2/tutorial/
--- 5. Data Structures：list , stack , tuple , dict
--- 7. Input and Output：str.format() , file
--- 6. Modules：module
--- 9. Classes：class
2）python 标准库文档；***python library***
http://docs.python.org/2/library/index.html
--- 2. Built-in Functions：filter() , map() , reduce() , repr() , eval() 等
--- 5. Built-in Types：str 对象方法，split() , partition() , join() , strip() 等
--- 18.2. json — JSON encoder and decoder: dumps(), Loads(), dump(), load()
3）tornado 官方文档（中文）：***tornado doc***
http://www.tornadoweb.cn/documentation
--- Tornado 攻略－请求处理程序和请求参数：第一个案例，理解处理 URL;
--- Tornado 攻略－模板；模板结构；模板的基本使用；模板输出与转义(escape);
--- Tornado 攻略－静态文件和主动式文件缓存
--- 生产环境下的部署

任务 1：python 结构类型数据
List 数据对象是一种结构类型，使用“[...]”包含数据对象。实际程序中，还有“{...}”和“(...)”包含数据对象的案例。我们进一步学习这些内容
进入文档“***python tutorial*** -> 5. Data Structures”
任务 1.1：List 类型的常用函数
- 5.1. More on Lists

快速阅读后，使用 python 解释器测试以下函数的效果。
index(), sort(), reverse()
- 5.1.1. Using Lists as Stacks

自己用 append() 和 pop() 函数模拟堆栈操作。并自己实现以下函数

```
def reverse(alist):
```

```
    """
    input a list
    return the reverse list with list.pop()
    """
    pass # write your code here and replacing pass
   #注意使用使用 alist[:]，why?
```

- 5.1.2. Using Lists as Queues

练习队列操作

任务 1.2：List 类型与函数式编程

Python 支持一些简单函数式编程功能。

- 5.1.3. Functional Programming Tools

练习 filter, map, reduce 的所有练习。

- 5.1.4. List Comprehensions

练习 1, 2, 3, 6 案例。

到目前为止，你大概可以体会到如果没有性能需求，为什么不使用 C，C++编程的理由了。

任务 1.3：del 语句

练习 5.2. The del statement。清除 list 中的项或一个变量

任务 1.4：元组（tuple）

tuple 用于表示多个对象之间的关系。例如：（用户名，密码，邮箱）。在 python 中用这样的语句保存这样的关系。

```
userinfo = (“sun”,”12345”,”sun@qq.com”)
name, pass, email = userinfo
print pass
```

练习 5.3. Tuples and Sequences 的练习 1.

注意：元组类型是 *immutable*

任务 1.5：字典（dict）类型

字典类型是除 List 类型外，在 python 程序中最常用的数据类型。*dict* 用于处理（key : value）的 hash 表。注：二年级学生不要纠结什么是 Hash Table

练习 5.5. Dictionaries 的 1，4 案例

任务 1.6：相关的一些编程要点

- 5.6. Looping Techniques

完成练习 1, 2, 5

- 4.7. More on Defining Functions

这里主要讨论了

1）函数的动态参数实现方法，如 c 语言的 printf

如果你有动态参数需要，请参考这里

2）lambda expression

例如元组列表的排序技术（非常有用）。

```
>>> pairs = [(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')]
>>> pairs.sort(key=lambda pair: pair[1])
>>> pairs
[(4, 'four'), (1, 'one'), (3, 'three'), (2, 'two')]
```

特别强调：（在有语言和数据结构基础后）任何计算机语言不是给你看（学）的，

是给你用的。在知道一些基本的语法和函数后，只有在使用中才能发现问题，然后返回教程和参考手册，才能学好。

任务 2：输出与文件读写

文档“***python tutorial*** -> 7. Input and Output”

任务 2.1：基本的格式化输出

练习 ***python tutorial*** -> 7.1. Fancier Output Formatting ：

- 函数 repr()，将任意对象输出为解释器可识别的字符串，例如：

```
>>> # The repr() of a string adds string quotes and backslashes:
... hello = 'hello, world\n'
>>> hellos = repr(hello)
>>> print hellos
'hello, world\n'
>>> # The argument to repr() may be any Python object:
... repr((x, y, ('spam', 'eggs')))
"(32.5, 40000, ('spam', 'eggs'))"
```

使用 eval() 函数计算 repr() 输出字符串或字符串表达式

- 输出参数与格式化

```
>>> print '{0} and {1}'.format('spam', 'eggs')
spam and eggs
>>> print '{1} and {0}'.format('spam', 'eggs')
eggs and spam
```

```
>>> print 'This {food} is {adjective}.'.format(
...       food='spam', adjective='absolutely horrible')
This spam is absolutely horrible.
```

```
>>> print 'The story of {0}, {1}, and {other}.'.format('Bill', 'Manfred',
...                                                    other='Georg')
The story of Bill, Manfred, and Georg.
```

练习 ***python tutorial*** -> 7.1.1. Old string formatting :

```
>>> import math
>>> print 'The value of PI is approximately %5.3f.' % math.pi
The value of PI is approximately 3.142.
```

任务 2.2：文件读写

python 文件类似 C 语言，例如：打开文件 open(filename, mode)。

练习 ***python tutorial*** -> 7.2. Reading and Writing Files

- 读文本文件

练习 ***python tutorial*** -> 7.2.1. Methods of File Objects 前 3 个读文本练习。

- 关闭文件

后面有案例

任务 2.3：一些常用的字符串处理函数

为了有效解析文件中的文本，最有用的函数是：split() , partition() , join(), strip()

例如：文件部分内容如下

```
STARRING:Cary Elwes, Robin Wright, Andre the Giant, Mandy Patinkin
DIRECTOR:Rob Reiner
PRODUCER:Arnold Scheinman, Rob Reiner
SCREENWRITER:William Goldman
```

在处理每一行时，我们可以使用

```
fname , fvalue = s.partition(':')[::2]
```

从每一行得到（属性名，属性值）的元组。

在处理 Starring 时，我们可以使用 stars = fvalue.split(‘,’) 得到列表

任务 2.4：使用 json 格式文本

在 internet 编程时，一个更通用的表示复杂内容的方法就是 JSON 格式文本。

例如：我们把前面的文本写成如下格式：

```
{
    “STARRING”:
        [“Cary Elwes”, “Robin Wright”, “Andre the Giant”, “Mandy Patinkin”],
    “DIRECTOR”:
        ”Rob Reiner”,
    “PRODUCER”:
        [”Arnold Scheinman”, “Rob Reiner”],
}
```

那么我们可以用这样的语句，从文件直接加载内容到变量

```
import json
f = open(‘myfile.txt’ , ’r’)
filminfo = json.load(f)
f.close()
```

同样，我们可以把复杂的内容，使用 json.dump 写入文件。

具体内容参见 http://docs.python.org/2/library/json.html

任务 3：python 的模块化，结构化技术

为了提供可服用，可扩展，高质量的程序，python 提供了模块、对象和包技术。

一个应用程序通常有若干功能模块，每个模块有若干对象，公用程序可以打成程序包。这里仅讨论模块与对象。

任务 3.1：模块定义

按 ***python tutorial*** -> 6. Modules 所述，将 *Fibonacci numbers module* 的代码保存在当前目录的 fibo.py 中。

Import fibo 就可以使用其中定义的函数、类

任务 3.2：执行模块脚本（初始化）

当 import 模块时，所有非类和函数内部的代码都将被执行。

其中，在如下条件快中的语句仅当模块作为主程序时，被执行

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

这个特点，通常用于程序分模块调试。

任务 3.3：快速学习类的定义与使用

参见：***python tutorial*** -> 9.3.2. Class Objects

- 定义类

```
class MyClass:
    """A simple example class"""
    i = 12345
    def f(self):
        return 'hello world'
```

- 定义构造函数与属性

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

注意，所有属性必须在构造函数 __init__(self，参数) 中初始化。
属性申明，self.属性名 = value

- 函数的继承

class DerivedClassName(BaseClassName):

任务 4；案例，使用 tornado 实现“Lab 3 PHP”要求
大家可以参考：http://www.cnblogs.com/joyeecheung/p/3390475.html
任务 4.1：ex1, The first task is to convert music.html into a tornado template
修改的方法是：假设文件列表 items 以准备好。我们仅需要重复生成<li>元素就 OK 了。

```
{% for item in files %}
    <li class="mp3-item">
        <a href="songs/{{ item }}">{{ item }}</a>
       <span class="mp3-size">99b<span>
   </li>
{% end %}
```

注意：任何程序都是迭代开发的，一步完成是不科学的。
将文件另存为 template.html。运行“Tornado 攻略一模板”程序，任务完成 50%。
下一步按提示，可以找到很多代码，将“./songs”目录下文件列表获得。任务完成
任务 4.2：解决 txt 和 mp3 文件无法下载问题。
1）根据上接课介绍，谷歌“python tornado download files”
显然，stackoverflow 的第一答案就可以解决这个问题
2）或者，阅读 tornado 文档
在“Tornado 攻略一静态文件和主动式文件缓存”找到官方的答案。
tornado.web.Application 类需要两个参数（在 tonado 目录，
C:\Python27\Lib\site-packages\tornado
下找 web.py，用 notepad++打开，搜“Class Application”就找到类的说明和__init__函数）看到如下结果：

```
1403    def __init__(self, handlers=None, default_host="", transforms=None,
1404                 wsgi=False, **settings):
1405        if transforms is None:
1406            self.transforms = []
1407            if settings.get("gzip"):
1408                self.transforms.append(GZipContentEncoding)
1409            self.transforms.append(ChunkedTransferEncoding)
1410        else:
1411            self.transforms = transforms
1412        self.handlers = []
```

这里面有丰富的信息，如对 gzip 传输的支持、wsgi 部署的设置等。我们仅关心 Setting 是一个 dict 类型。只要 setting 中包含"static_path" key，程序会做相关处理，并正确处理/robots.txt 和 /favicon.ico 文件（给网络爬虫和浏览器的网页图标）。

handler 是一个 list，元素是 tuple（url path, handler class, parameter）。需要作为静态文件供用户下载，需要使用 tornado.web.StaticFileHandler 处理器及静态文件在服务上的绝对路径参数。

任务 4.3：解决 txt 和 mp3 文件下载可能的高级方案。

“在生产环境下，你可能会使用 nginx 这样的更有利于静态文件伺服的服务器”，这会引起我们对生产环境部署的兴趣。

转入“***tornado*** -> 生产环境下的部署”内容。

注意：如果你运用代理，实现了 2－4 个 tornado 应用实例进程的部署，证明你掌握了部署高性能 HTTP 服务的基本能力。请邮件告知你的技术博客，或将技术报告邮件发给我。

任务 4.4：浏览器不知道.txt 是附件，直接打开怎么办？

这时一个新的问题，需要你对 HTTP 协议 header 有一定的了解。

谷歌“http header”

打开链接，找到 Responses 部分表格，在向浏览器发送 message-body 之前，先在 message-header 中添加 Content-Disposition 告知浏览器下载文件。例如：

self.set_header("Content-Disposition", 'attachment; filename="fname.ext"')

self.write("You wrote " + self.get_argument("message"))

参考：http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

任务 4.4：模块化程序

为了尽可能保持主程序的简单，我们将文件操作与处理放在 musicfile.py 中定义相关的类与函数，并测试正确（测试方法，主程序代码）

改写现有程序，似乎程序结构好了许多。

任务 4.5：迭代开发程序

完成 lab3 其他的要求，见参考实现就 OK 了。

作业与练习：
1、简答以下问题
1）编一个小程序，将 Fibonacci 数的前 10 个数存入 List，并按倒序打印
2）设计 stack 类
3）编序一个模块（ini.py），处理 .ini 格式的配置文件
4）编写一个小程序，显示自己的代码。备注：使用模块”__file__”属性

二、操作与实践
1、Homework：Homework 3 - Movie Review Part Deux

三、下一次测试的内容与要求
下一次测试是做一个博客网站的一些页面。
1）会告知你某博客网站的网页
2）首先，你需要将指定部分的网页，变成模板
3）然后，将用试题指定的数据生成博客页面
4）功能包括：登陆，发布我的博客，评论
5）提交 Homework 可以保证本次测试成绩 60％以上。