Python Good Coder考试指南

有兴趣的同学，可以学习以下课程Engineering Excellence系列之如何做好Python代码的Code Review

1. 编写说明

在做Python Good Coder二审的review过程中，经常发现：已经经过一审review的代码中，仍然存在很多问题。

其中一些问题是非常基础的，而且这些问题往往在多个同学的cooder重复出现。为了提高review的效率，写了这个指南。

这个指南供一审的reviewer参考，同时也供参加考试的同学参考（包括在还没有开始写考试代码的同学）。

在我的理解中，参加Good Coder的考试更像是一个training，希望所有人（参加考试的同学，负责review的同学）都能有

一个实质性的提高，而不只是为了应付一个流程。

2. 考题说明

Python Good Coder的考试需要写一个mini spider。这个程序虽然很小，但是“麻雀虽小，五脏俱全”，已经是一个相对完整

的Daemon程序。 从小的方面讲，这只是一个demo，是一个运行在单机上的程序；从大的方面讲，如果设计的合理，这个程序 中的框架设计完全

可以轻松的扩展到一个大型的分布式系统中。

在这个考题中，希望考察的要点如下：

- 编程规范（http://styleguide.baidu.com/style/python/index.html）中所列出的各规范
    - 目前编程规范定义了一些程序格式和风格方面的要求
- 多线程程序的编写
    - 多线程的编程模式在后台开发中是非常常用的，这对软件工程师来说应该是一个基本的技能
- 软件构架的能力
    - 包括：模块的切分，函数的切分，数据的封装等
- 一些细节的处理

    - 包括：配置的读取，日志的打印方式，程序参数的读取等

应该说，以上这些考察点还是非常基本的，并不是很高的要求。但是非常遗憾（以及非常坦率）的说，相当大比例的同学

在初次提交代码时（包括经过反复修改后），仍然完全无法满足以上的要求。由于reviewer的资源和精力所限，根本无法

让每个同学在考试结束时都写出完全符合要求的代码，所以很多情况下在“满足基本要求”的情况下就放过了。

这也是写这个指南的出发点，就是希望各位同学能本着“对自己负责”的态度，对照这份指南，找到自己代码中的不足，从而提高。

下面会针对上面列出的这些考察要点做详细的说明。

3. 要点解读

3.1 编程规范的内容

3.1.1 import

编程规范中的以下两条 经常被忽略：

- [强制] 禁止使用from xxx import yyy语法直接导入类或函数(即yyy只能是module或package，不能是类或函数)。（第2.1部分）
- [强制] 必须按如下顺序排列import，每部分之间留一个空行（第3.8部分）
    - 标准库
    - 第三方库
    - 应用程序自有库

3.1.2 程序的段落

有些同学非常不习惯在代码中使用“空格”或“空行”来进行内容的分隔。这样的代码可读性非常差。

有些同学可能认为这只是一种形式。但是请记住，形式代表的“逻辑”，而逻辑对程序来说是非常重要的。

对于这样的代码，我也没有精力一一详细的标记出来，只能靠大家自己注意了。

3.2 多线程程序的编写

3.2.1 数据的互斥访问

- mini-spider中，在涉及到“url的去重”时，一般会使用一个内存中哈希表。由于这个哈希表会有多个线程同时访问的情况，所以必须做互斥保护。
    - 有些同学忽略了这方面的考虑，对公共访问的数据没有任何保护
    - 这个地方，请大家看一下python中threading.Lock()的说明。
- 另外一个问题是，很多同学在使用lock的时候，没有“模块化”和“数据封装”的思路，将lock.acquire()和lock.release()调用随意的放在程序的各个地方
    - lock的调用，应该作为“数据模块”的一个“内部实现”，被封装在数据模块的接口之内（见《代码的艺术》（http://wiki.baidu.com/pages/viewpage.action?pageId=81003267）中的详解）

3.2.2 线程间的同步

在mini-spider中，涉及到两处线程间同步的处理：

- 抓取任务被分发到多个抓取线程（crawler）
- 在所有抓取任务都被执行完成后，整个程序结束退出

很多同学在实现这两个功能时，使用的机制非常复杂，可读性也不好。其中这两个问题在多线程编程中都是经常遇到的，也有非常成熟而简单的解

决方案。

请参考系统库提供的Queue（尤其注意get()，put()，task_done()，join()这几个函数）

3.2.3 临界区内的处理

被锁保护起来的区域，被称为临界区（也就是在lock.acquire()和lock.release()之间的区域）

在临界区内的处理要非常注意，有很多禁忌。如：

- 尽量不要执行time-consuming的操作，更不要执行可能阻塞的操作（如IO读写）
- 不能出现无捕捉的exception。如果发生这样的情况，会引起lock.release()没有调用，从而导致死锁的发生

以上这些问题在以往的review中都发生过。

3.3 软件构架方面

3.3.1 模块的切分

很多提交的代码中，代码都写在一个文件中（如：mini_spider.py）。有些同学会把代码分在3-4个文件中，但也还是做的不够。

我感觉很多同学对如何“分文件”是没有考虑的，分文件是比较随意的。在C/Python/Go里面，一个文件都代表了一个“模块”，而“分模块”

在软件开发中是非常重要的工作，模块分的不好，这个代码的可维护性会受到很大的影响。

关于软件开发中怎么分模块，见《代码的艺术》（http://wiki.baidu.com/pages/viewpage.action?pageId=81003267）中的详解

这里给出一个建议的模块切分方法（注：不是标准答案。请想一下这么切分模块的原因）：

- mini_spider.py：主程序

- crawl_thread.py: 实现抓取线程

- webpage_parse.py: 对抓取网页的解析

- webpage_save.py: 将网页保存到磁盘

- seedfile_load.py: 读取种子文件

- config_load.py: 读取配置文件

- url_table.py: 保存已抓取的url列表，用于去重

3.3.2 函数的切分

这也是软件编写中一个非常需要重视的地方。规划一个函数时，最重要是它功能的单一，及控制它的规模。在review中，经常出现的问题包括：

- 在一个函数中，平铺了好几段功能完全无关的代码
    - 其实这几段代码是应该分别写为函数的
- 函数的名字和其实现的内容无法完全对应起来
    - 这种情况下很容易让人误解，比如看着一个crawl_webpage()的函数，其中又包含着save to file的功能
- 函数的规模太大
    - 在python中，大部分函数都应该控制在一屏以内（大约是40行）

- 很多同学的代码中出现非常长的函数，给程序的可维护性带来很大的隐患，在功能切分和程序组织方面的能力还需要提高

3.4 一些细节的处理

3.4.1 对exception的捕捉

常见问题包括：

- 对某些IO操作没有捕捉异常
  - 编程时应该有这样的意识，只要是IO操作，都有可能发生异常。
  - 比如，对os.makedir()这个调用，很多同学都没有意识到这里是可能出现异常的
- 异常捕捉的区域过大。应注意规范中的这条建议：
  - [建议] 建议try中的代码尽可能少。避免catch住未预期的异常，掩藏掉真正的错误（第2.2部分）

3.4.2 构造函数的编写

一些同学在类的构造函数（也就是__init__()）中执行了可能出错的操作。

这应该是一个常识（不仅仅是python）：在构造函数中不能执行可能出错的操作（因为根本无法获得这个错误的返回值，从而进行错误处理）

3.4.3 exit()的调用

在程序初始化过程中，一些同学在某个函数内部执行失败后，就直接调用os.exit()强行退出。

这种用法是非常不推荐的。想象一下，如果这样的调用很多，当程序退出的时候，往往很难找到程序退出的原因。

推荐的方式是，将os.exit()的调用集中到main函数中

3.4.4 正则的使用

在做正则匹配的时候，一些同学直接使用re.match(pattern, string)这种调用方式。

由于这个操作会反复执行，推荐的方式是

```
# 只执行一次
```

```
prog = re.compile(pattern)
```

```
# 可能被调用多次 result = prog.match(string)
```

相比前一种方式，这种方式的性能会有很大的提升（尤其是在吞吐量比较大的时候）

3.4.5 注释的使用

常见问题：

- 很少写注释
- docstring（用"""包围）和普通注释（以#开头）不做区分
    - 搜索和思考一下：什么时候用docstring，什么时候用#

-- Main.zhangmiao02 - 2015-08-20