# Google Python Style Guide

1. 背景

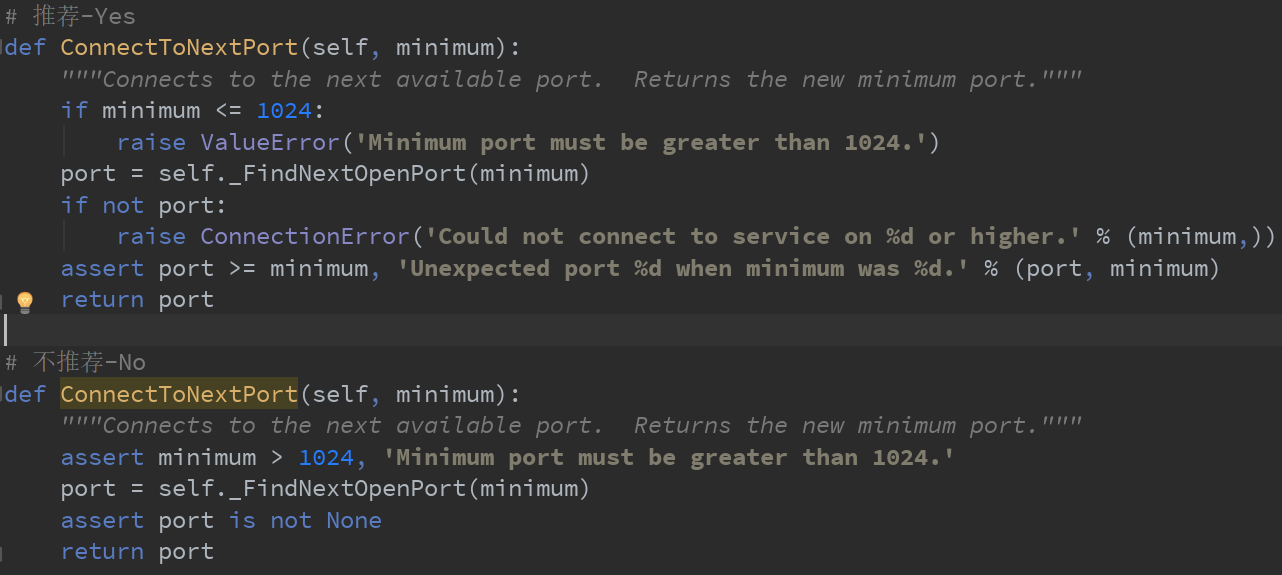
二、Python语法规范

* 1. Lint
     1. 定义：Python代码风格检查工具；
     2. 其他：(略)
  2. Import

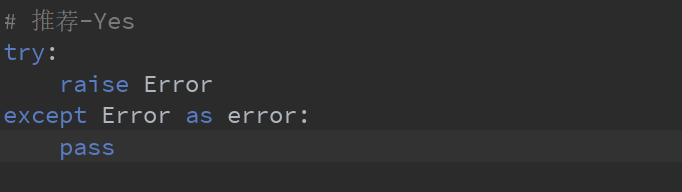
用于导入软件包或者模块

* + 1. 定义：
    2. 其他：
  1. Packages
  2. 异常(Exceptions)
     1. 定义：异常是用于处理代码错误或其他异常条件发生时打破代码块正常操作流的方法。python可以自动触发异常，raise（内置函数）的定义为显示的抛出异常，用户可以使用raise进行判断，显式的引发异常，raise执行后程序将不再向下执行。assert语句用于检测某个条件表达式是否为真。assert语句又称为断言语句，即assert认为检测的表达式永远为真。断言用来指定某一操作必须为真。
     2. 优缺点：
     3. 使用：

1. 异常必须要有确定的条件，引发异常的关键字为raise。
2. 异常的抛出最好使用一阶段形式：如raise MyError(‘Error message’)或raise MyError()。不要使用两参数形式：如raise MyError, ‘Error message’。
3. 最好使用内置的异常函数，如ValueError、ConnectionError等。

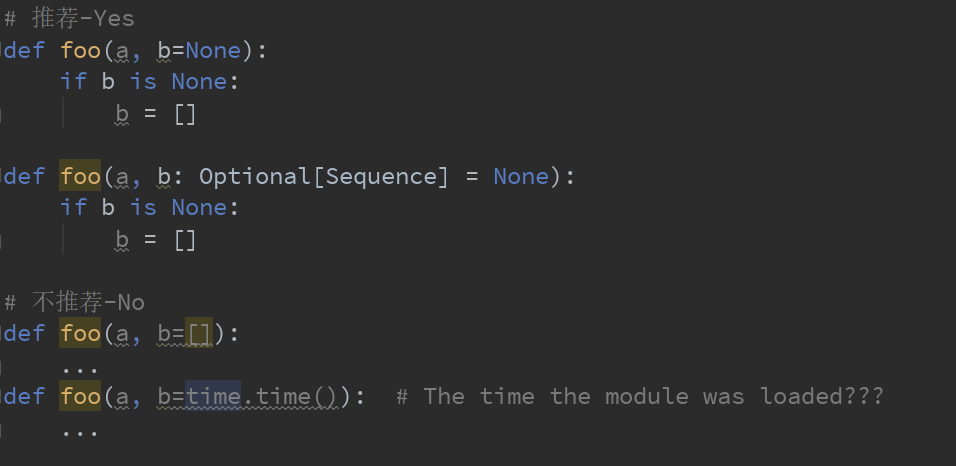


1. 库或包可以定义它们自己的异常。这样做时，它们必须从现有的异常类继承。异常名应该以错误(Error)结尾，如foo.FooError。
2. 除非特殊情况的情况下，最好不要使用-all来捕获所有的异常。
3. 尽量减少try/except块中的代码量。尝试的主体越大，异常被一行代码引发的可能性就越大，而你没有料到会引发异常。在这些情况下，try/except块隐藏了一个真正的错误。
4. 使用finally子句执行代码，这样不管try块中是否引发异常，都会执行finally子句中的代码，这对于清理代码现场非常有用。



* 1. 全局变量(Global variables)
  2. 嵌套/局部/内部类和函数表达式
  3. Comprehensions & Generator Expressions
  4. Default Iterators and Operators
  5. Generators
  6. Lambda函数
  7. 条件表达式
  8. 默认参数值
     1. 定义：可以在函数参数列表的末尾为变量指定值。
     2. 优缺点：
     3. 使用：

1. 不要在函数或方法定义中使用可变对象作为默认值。
2. 建议的格式如下：

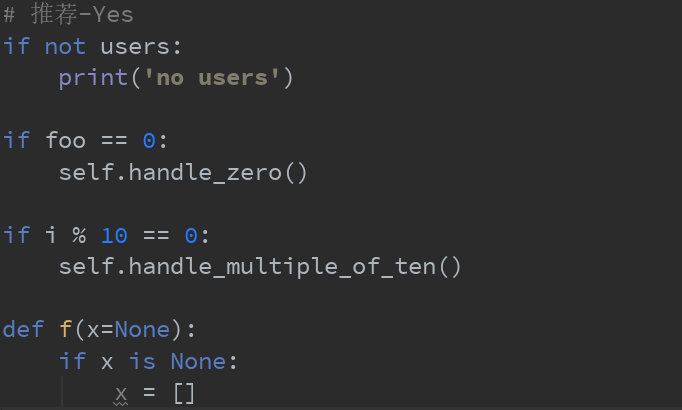


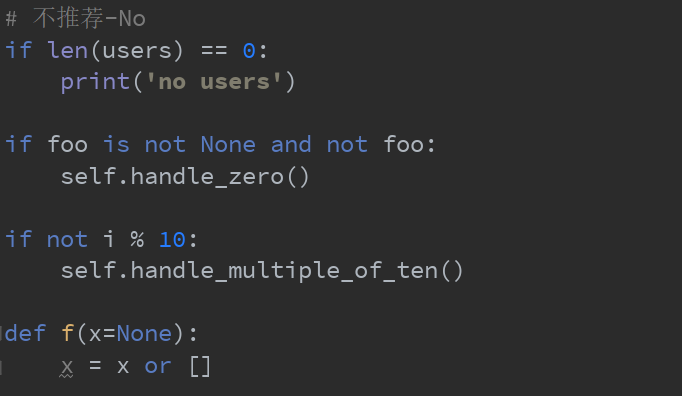
* 1. 属性(Properties)
  2. True/False的评估

如果可能最好使用隐式的false形式。

* + 1. 定义：在布尔形式的上下文中，Python将所有的“空”值认定为false。例如0、None、[]、{}会被判断为false，但’0’是string，结果为True。
    2. 优缺点：
    3. 使用：

1. 尽可能使用隐式的false用法，如if foo:的效果要比if foo !=[]好。
2. 对于‘空’类型的布尔变量，建议使用is或is not来进行判断。
3. 最好不要用‘==’对布尔类型的变量和False进行比较，可以使用if not x的形式代替。
4. 对于序列(字符串、列表、元组)，若被用来当作空序列操作，最好不要使用len()函数来操作判断。





* 1. 弃用语言特性
  2. Lexical Scoping
  3. 装饰器(Decorators)
  4. Threading
  5. Power Features
  6. Modern Python
  7. Type Annotated Code

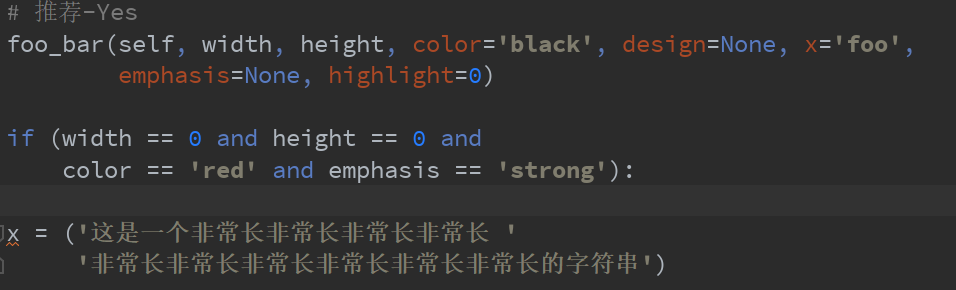
三、Python Style Rules

3.1 分号

* + 1. Python语法中没有分号的概念。
    2. 不要在行尾加分号, 也不要用分号将两条命令放在同一行。

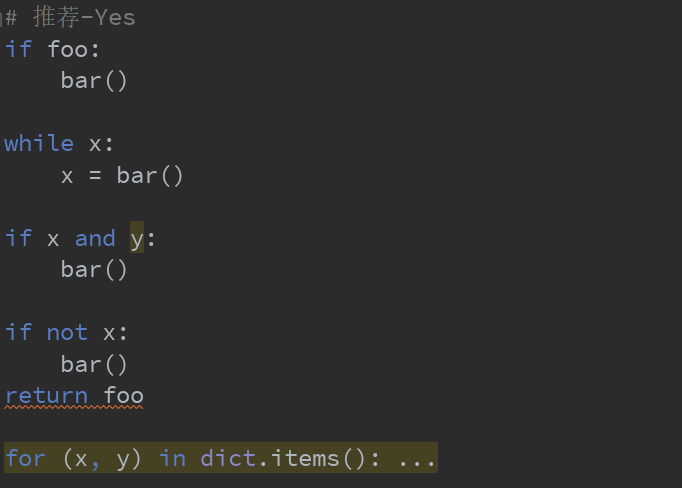
3.2 行长

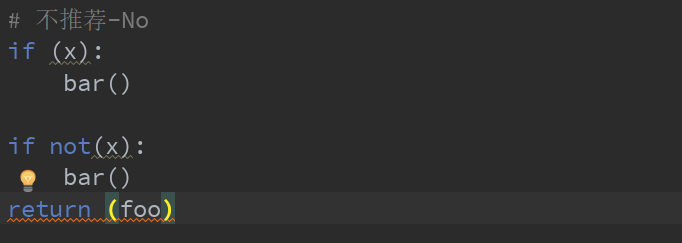
* + 1. 每行不要超过80个字符。
    2. 特殊情况除外：(1)长的导入模块语句。(2)注释行。
    3. Python会将圆括号、中括号和花括号中的行隐式的连起来；
    4. 注释中，如果必要，将长的URL放在一行上；



3.3 括号

* + 1. 宁缺毋滥的使用括号。
    2. 除非是用于实现连接，否则不要在返回语句或者条件语句中使用括号。不过在元组两边可以使用括号。

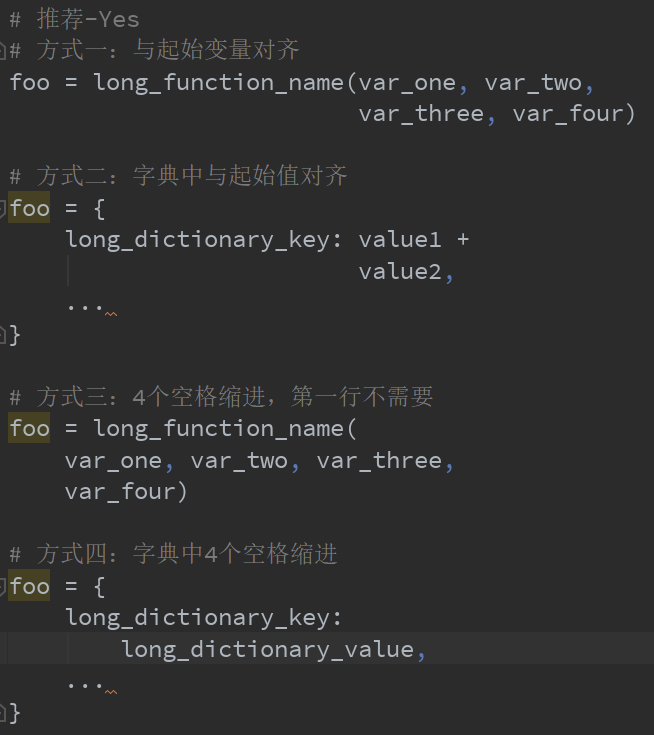




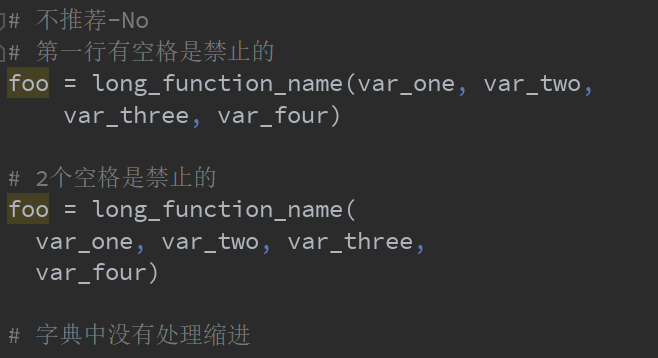
3.4 缩进

* + 1. 用4个空格来缩进代码。
    2. 绝对不要用tab键，也不要tab和空格键混用。
    3. 对于行连接的情况, 要么垂直对齐换行的元素，或者使用4空格的悬挂式缩进(这时第一行不应该有参数)。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



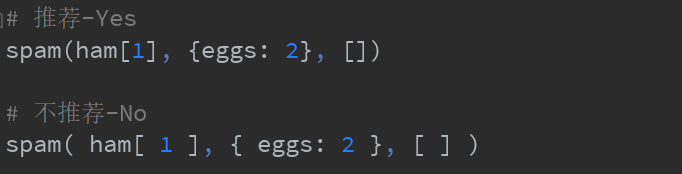
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3.5 空白行

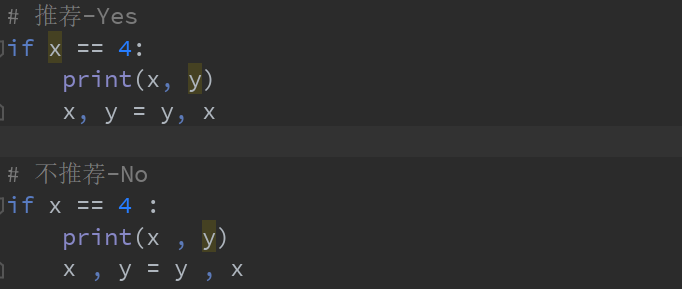
* + 1. 顶级定义之间空两行，比如函数或者类定义。
    2. 方法定义之间空一行。

3.6 空格

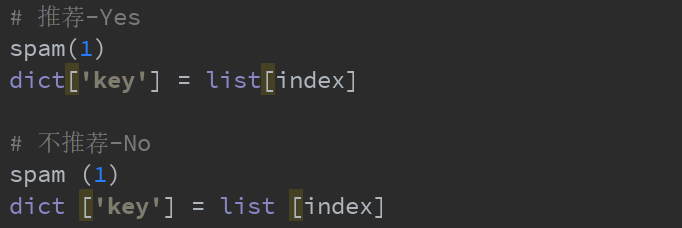
* + 1. 括号内不要有空格。



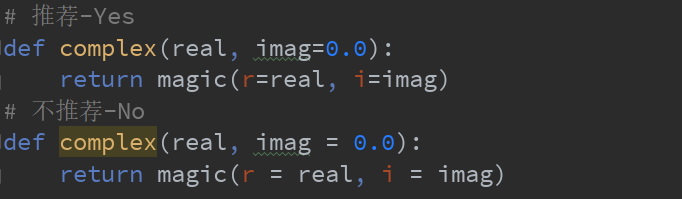
* + 1. 不要在逗号, 分号, 冒号前面加空格, 但应该在它们后面加(除了在行尾)。



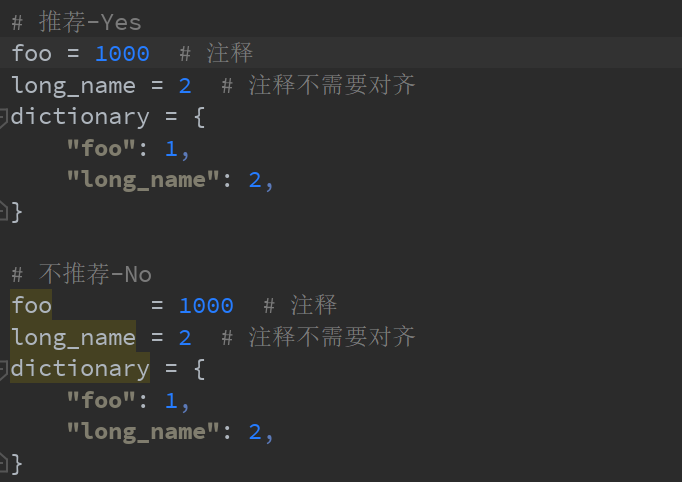
* + 1. 参数列表, 索引或切片的左括号前不应加空格。



* + 1. 二元操作符两边都加上一个空格, 比如赋值(=), 比较(==, <, >, !=, <>, <=, >=, in, not in, is, is not), 布尔(and, or, not)。
    2. 至于算术操作符两边的空格该如何使用, 需要你自己好好判断，不过两侧务必要保持一致。
    3. 当'='用于指示关键字参数或默认参数值时, 不要在其两侧使用空格。



* + 1. 不要用空格来垂直对齐多行间的标记, 因为这会成为维护的负担。



3.7 Shebang Line(#!)

* + 1. 根据 [PEP-394](http://www.python.org/dev/peps/pep-0394/) , 程序的main文件应该以 #!/usr/bin/python2或者 #!/usr/bin/python3开始；
    2. 相互调用的子文件可以省略；(?)

3.8 注释及文档字符串

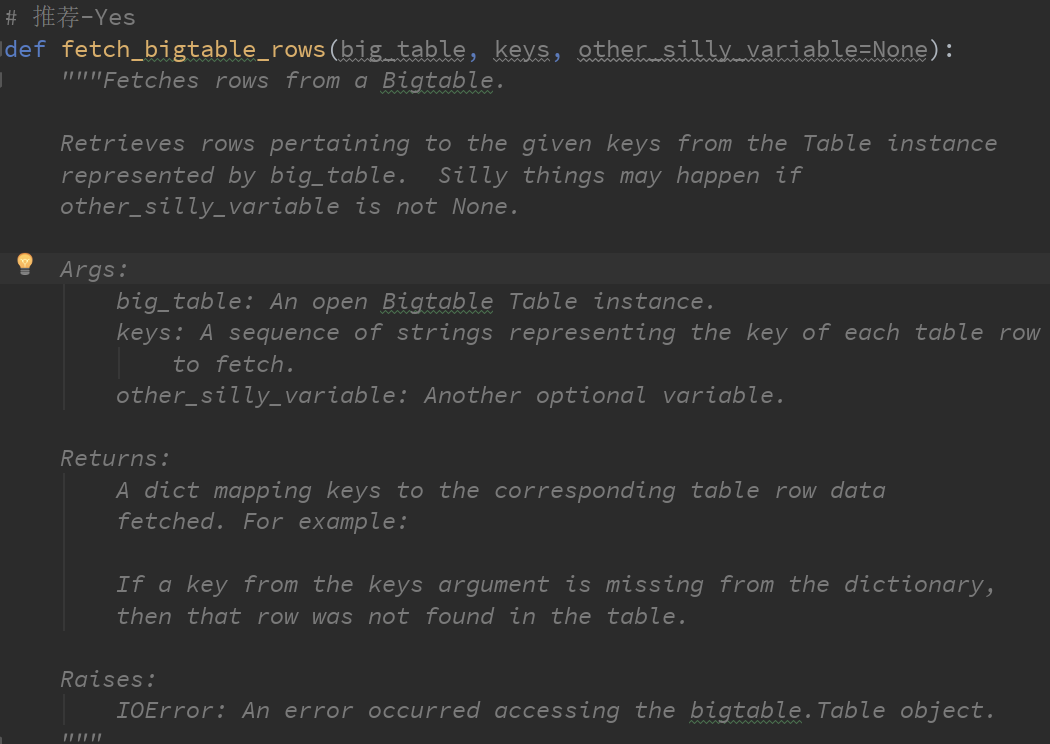
文档字符串定义：

* 文档字符串是包、模块、类或函数里的第一个语句;
* 可以通过对象的\_\_doc\_\_成员被自动提取；
* 使用三重双引号声明；
* 格式化规范：首先是一行以句号, 问号或惊叹号结尾的概述(或者该文档字符串单纯只有一行)。接着是一个空行。接着是文档字符串剩下的部分, 它应该与文档字符串的第一行的第一个引号对齐；
  + 1. 模块：

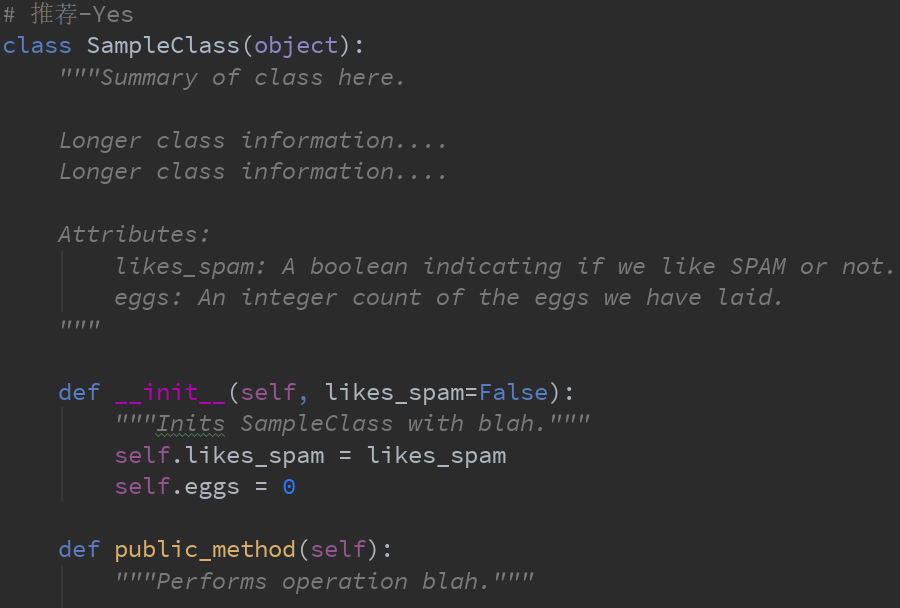
格式：每个文件应该包含一个许可样板，根据项目使用的许可，选择合适的模板；

* + 1. 函数和方法：

1. 一个函数必须要有文档字符串, 除非它满足以下条件：外部不可见、非常短小、简单明了；
2. 文档字符串应该包含函数做什么, 以及输入和输出的详细描述。 通常, 不应该描述"怎么做", 除非是一些复杂的算法。文档字符串应该提供足够的信息, 当别人编写代码调用该函数时, 他不需要看一行代码, 只要看文档字符串就可以了。 对于复杂的代码, 在代码旁边加注释会比使用文档字符串更有意义。
3. 主要格式内容：Args—列出每个参数的名字, 并在名字后使用一个冒号和一个空格, 分隔对该参数的描述。Returns—描述返回值的类型和语义。Raises—列出与接口有关的所有异常。(样式一)



* + 1. 类：
       1. 用于描述类的字符串，如果类中有公共属性(Attributes)，那么文档中应该有一个属性(Attributes)段。

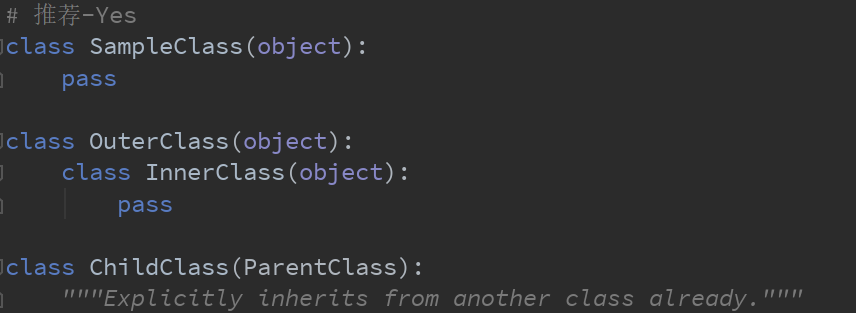


* + 1. 块注释和行注释：
       1. 如果你在下次[代码审查](http://en.wikipedia.org/wiki/Code_review)的时候必须解释一下, 那么你应该现在就给它写注释。
       2. 对于复杂的操作, 应该在其操作开始前写上若干行注释。
       3. 对于不是一目了然的代码, 应在其行尾添加注释。

建议：绝对不要描述代码。

3.9 类

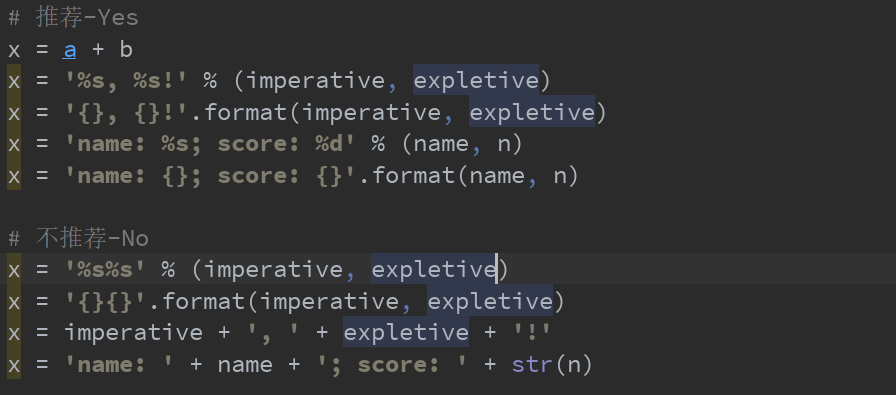
* + 1. 如果一个类不继承自其他类，就显式的从object继承，嵌套类也一样。(这样做的好处可以避免Python潜在的不兼容性问题，而且默认定义了一些特殊的方法，包括\_\_new\_\_, \_\_init\_\_, ......)





3.10 字符串

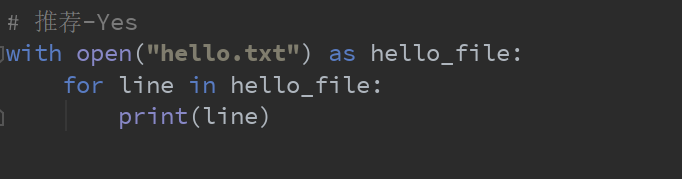
* + 1. 字符串要适当使用’+’、空格或者format函数；



* + 1. 避免在循环中用+和+=操作符来累加字符串；
    2. 同一文件中，保持使用字符串引号的一致性。在字符串内可以使用另外一种引号，以避免在字符串中使用。
    3. 其他

3.11 文件和Sockets

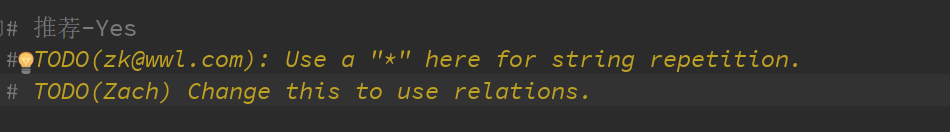
* + 1. 在文件和sockets结束时，显示的关闭它。



* + 1. 对于不支持使用with语句的类似文件的对象，使用contextlib.closing().

3.12 TODO注释

* + 1. 为临时代码使用TODO注释，它是一种短期解决方案。
    2. TODO注释应该在所有开头处包含"TODO"字符串，紧跟着是用括号括起来的你的名字，email地址或其它标识符。然后是一个可选的冒号。接着必须有一行注释，解释要做什么。主要目的是为了有一个统一的TODO格式，这样添加注释的人就可以搜索到(并可以按需提供更多细节)。写了TODO注释并不保证写的人会亲自解决问题。当你写了一个TODO，请注上你的名字。
    3. 如果你的TODO是为了提醒“将来做某事”的形式，那么最好指定日期或者一个特定的事件。



3.13 导入格式

* + 1. 每个导入应该独占一行。



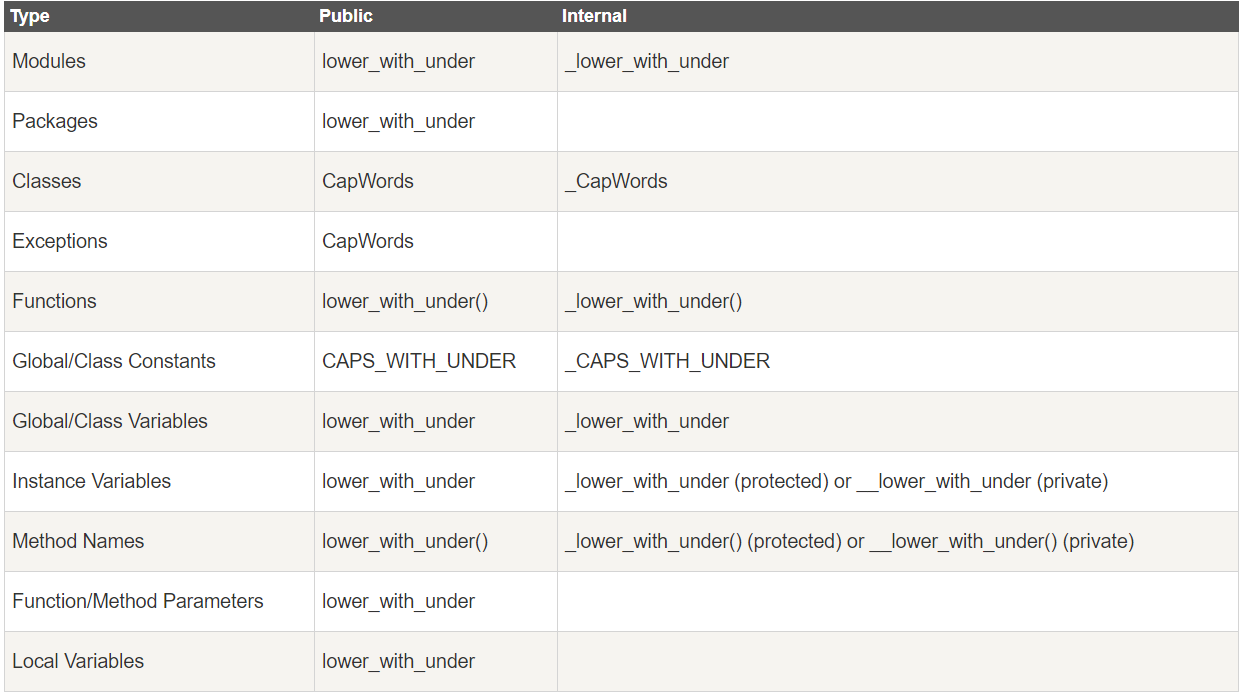
* + 1. 导入应该放在文件顶部，位于模块注释和文档字符串之后，模块全局变量和常量之前。
    2. 导入应该按照从最通用到最不通用的顺序分组：(1)标准库导入，(2)第三方库导入，(3)应用程序指定导入。

3.14 声明

3.15 接入控制

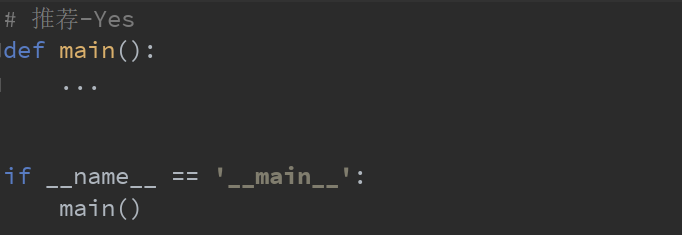
3.16 命名规则

* + 1. 应该避免的名称：
       - 1. 单字符名称，除了计数器和迭代器；
         2. 包/模块名中的连字符(-)，要避免出现;
         3. 双下划线开头并结尾的名称，这些都是Python保留；
    2. 命名约定：
       - 1. 所谓"内部(Internal)"表示仅模块内可用, 或者, 在类内是保护或私有的；
         2. 用单下划线(\_)开头表示模块变量或函数是protected的(使用import \* from时不会包含)；
         3. 用双下划线(\_ \_)开头的实例变量或方法表示类内私有;
         4. 对类名使用大写字母开头的单词(如CapWords, 即Pascal风格);
         5. 模块名应该用小写加下划线的方式(如lower\_with\_under.py)
    3. Python之父Guido推荐的命名规范：



3.17 main函数

* + 1. 即使一个打算被作为脚本的python文件，也应该是可导入的。并且简单的导入不应该导致该脚本的主功能被执行，主功能应该放在一个main()函数中；
    2. 所有的顶级代码在模块导入时都会被执行. 要小心不要去调用函数, 创建对象, 或者执行那些不应该在使用pydoc时执行的操作；



3.18 函数长度

3.19 类型注释

3.20 括号

1. Parting Words

参考资料：

1. <https://github.com/google/styleguide/blob/gh-pages/pyguide.md>
2. [http://www.runoob.com/w3cnote/google-python-styleguide.html](3.http:/www.runoob.com/w3cnote/google-python-styleguide.html)