python代码风格-PEP8

转载自http://www.douban.com/note/134971609/

Python 的代码风格由 PEP 8 描述。这个文档描述了 Python 编程风格的方方面面。在遵守这个文档的条件下，不同程序员编写的 Python 代码可以保持最大程度的相似风格。这样就易于阅读，易于在程序员之间交流。

1. 命名风格

1. 总体原则，新编代码必须按下面命名风格进行，现有库的编码尽量保持风格。
2. 尽量以免单独使用小写字母'l'，大写字母'O'，以及大写字母'I'等容易混淆的字母。
3. 模块命名尽量短小，使用全部小写的方式，可以使用下划线。
4. 包命名尽量短小，使用全部小写的方式，不可以使用下划线。
5. 类的命名使用CapWords的方式，模块内部使用的类采用_CapWords的方式。
6. 异常命名使用CapWords+Error后缀的方式。
7. 全局变量尽量只在模块内有效，类似C语言中的static。实现方法有两种，一是__all__机制;二是前缀一个下划线。对于不会发生改变的全局变量，使用大写加下划线。
8. 函数命名使用全部小写的方式，可以使用下划线。
9. 常量命名使用全部大写的方式，可以使用下划线。
10. 使用 has 或 is 前缀命名布尔元素，如: is_connect = True; has_member = False
11. 用复数形式命名序列。如：members = ['user_1', 'user_2']
12. 用显式名称命名字典，如：person_address = {'user_1':'10 road WD', 'user_2' : '20 street huafu'}
13. 避免通用名称。诸如 list, dict, sequence 或者 element 这样的名称应该避免。又如os, sys 这种系统已经存在的名称应该避免。
14. 类的属性（方法和变量）命名使用全部小写的方式，可以使用下划线。
15. 对于基类而言，可以使用一个 Base 或者 Abstract 前缀。如BaseCookie、AbstractGroup
16. 内部使用的类、方法或变量前，需加前缀'_'表明此为内部使用的。虽然如此，但这只是程序员之间的约定而非语法规定，用于警告说明这是一个私有变量，外部类不要去访问它。但实际上，外部类还是可以访问到这个变量。import不会导入以下划线开头的对象。
17. 类的属性若与关键字名字冲突，后缀一下划线，尽量不要使用缩略等其他方式。
18. 双前导下划线用于命名class属性时，会触发名字重整；双前导和后置下划线存在于用户控制的名字空间的"magic"对象或属性。

19. 为避免与子类属性命名冲突，在类的一些属性前，前缀两条下划线。比如：类Foo中声明__a,访问时，只能通过Foo._Foo__a，避免歧义。如果子类也叫Foo，那就无能为力了。
20. 类的方法第一个参数必须是self，而静态方法第一个参数必须是cls。
21. 一般的方法、函数、变量需注意，如非必要，不要连用两个前导和后置的下线线。两个前导下划线会导致变量在解释期间被更名。两个前导下划线会导致函数被理解为特殊函数，比如操作符重载等。

2 关于参数

1. 要用断言来实现静态类型检测。断言可以用于检查参数，但不应仅仅是进行静态类型检测。 Python 是动态类型语言，静态类型检测违背了其设计思想。断言应该用于避免函数不被毫无意义的调用。
2. 不要滥用 *args 和 **kwargs。*args 和 **kwargs 参数可能会破坏函数的健壮性。它们使签名变得模糊，而且代码常常开始在不应该的地方构建小的参数解析器

3 代码编排

1. 缩进。优先使用4个空格的缩进（编辑器都可以完成此功能），其次可使用Tap，但坚决不能混合使用Tap和空格。
2. 每行最大长度79，换行可以使用反斜杠，最好使用圆括号。换行点要在操作符的后边敲回车。
3. 类和top-level函数定义之间空两行；类中的方法定义之间空一行；函数内逻辑无关段落之间空一行；其他地方尽量不要再空行。
4. 一个函数：不要超过 **30 行**代码, 即可显示在一个屏幕类，可以不使用垂直游标即可看到整个函数；一个类：不要超过 **200 行**代码，不要有超过 **10 个**方法；一个模块 不要超过 **500 行**。

4. 文档编排

1. 模块内容的顺序：模块说明和docstring—import—globals&constants—其他定义。其中import部分，又按标准、三方和自己编写顺序依次排放，之间空一行。
2. 不要在一句import中多个库，比如import os, sys不推荐。
3. 如果采用from XX import XX引用库，可以省略‘module.’。若是可能出现命名冲突，这时就要采用import XX。

5. 空格的使用

1. 总体原则，避免不必要的空格。
2. 各种右括号前不要加空格。
3. 函数的左括号前不要加空格。如Func(1)。

4. 序列的左括号前不要加空格。如list[2]。
5. 逗号、冒号、分号前不要加空格。
6. 操作符（=/+=/-+/==/</>/!=/<>/<=/>=/in/not in/is/is not/and/or/not)左右各加一个空格，不要为了对齐增加空格。如果操作符有优先级的区别，可考虑在低优先级的操作符两边添加空格。如：hypot2 = x*x + y*y； c = (a+b) * (a-b)
7. 函数默认参数使用的赋值符左右省略空格。
8. 不要将多句语句写在同一行，尽管使用‘；’允许。
9. if/for/while语句中，即使执行语句只有一句，也必须另起一行。

6. 注释

1. 总体原则，错误的注释不如没有注释。所以当一段代码发生变化时，第一件事就是要修改注释！避免无谓的注释
2. 注释必须使用英文，最好是完整的句子，首字母大写，句后要有结束符，结束符后跟两个空格，开始下一句。如果是短语，可以省略结束符。
3. 行注释：在一句代码后加注释，但是这种方式尽量少使用。。比如：x = x + 1 # Increment
4. 块注释：在一段代码前增加的注释。在‘#’后加一空格。段落之间以只有‘#’的行间隔。比如：

```
# Description : Module config.
#
# Input : None
#
# Output : None
```

7. 文档描述

1. 为所有的共有模块、函数、类、方法写docstrings；非共有的没有必要，但是可以写注释（在def的下一行）。
2. 如果docstring要换行，参考如下例子,详见PEP 257

```
"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""
```

8. 编码建议

1. 编码中考虑到其他python实现的效率等问题，比如运算符‘+’在CPython（Python）中效率很高，都是Jython中却非常低，所以应该采用.join()的方式。
2. 与None之类的单件比较，尽可能使用‘is’is not’，绝对不要使用‘==’，比如if x is not None 要优于if x。
3. 使用startswith() and endswith()代替切片进行序列前缀或后缀的检查。比如：建议使用if foo.startswith('bar'): 而非if foo[:3] == 'bar':
4. 使用isinstance()比较对象的类型。比如：建议使用if isinstance(obj, int): 而非if type(obj) is type(1):
5. 判断序列空或不空，有如下规则：建议使用if [not] seq: 而非if [not] len(seq)
6. 字符串不要以空格收尾。
7. 二进制数据判断使用 if boolvalue的方式。
8. 使用基于类的异常，每个模块或包都有自己的异常类，此异常类继承自Exception。错误型的异常类应添加"Error"后缀，非错误型的异常类无需添加。
9. 异常中不要使用裸露的except，except后跟具体的exceptions。
10. 异常中try的代码尽可能少。比如：

```
try:
value = collection[key]
except KeyError:
return key_not_found(key)
else:
return handle_value(value)
```