**一 代码编排**

1 缩进。4个空格的缩进（编辑器都可以完成此功能），不使用Tab，更不能混合使用Tab和空格。

2 每行最大长度120，换行可以使用反斜杠，最好使用圆括号。

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。

**二 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.’，但是可能出现命名冲突，这时就要采用import XX。

**三 空格的使用**

总体原则，避免不必要的空格。

1 各种右括号前不要加空格。

2 逗号、冒号、分号前不要加空格。

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 操作符左右各加一个空格，不要为了对齐增加空格。

6 函数默认参数使用的赋值符左右省略空格。

7 不要将多个语句写在同一行，尽管使用‘；’，如果允许。

8 if/for/while语句中，即使执行语句只有一句，也必须另起一行。

**四 注释**

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！

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。

1 块注释，在一段代码前增加的注释。

# xxxxxxxxx

# xxxxxxxxx

2 行注释，在一句代码后加注释。比如：x = x + 1 # Increment x

但是这种方式尽量少使用。

3 避免无谓的注释。

**五 文档描述**

1 为模块写描述：简述模块作用，作者和日期，参考下面的例子：

#!/usr/bin/env python

#coding: utf-8

”””

Module: xxxxxx

Author: xxxxxxx

Date：xxxxxxx

”””

2 为类和外部接口写描述：简述类和外部接口的作用：

def StartProcess(self, processName)

”””

xxxxxx

”””

**六 命名规范**

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。

1 尽量单独使用小写字母‘l’，大写字母‘O’等容易混淆的字母。

2 模块命名尽量短小，使用Pascal命名法（每一个单词的首字母都采用大写字母的命名格式）的方式。

3 包命名尽量短小，使用全部小写的方式，不可以使用下划线。

4 类的命名使用Pascal命名法（每一个单词的首字母都采用大写字母的命名格式）的方式，内部使用的类采用加下划线前缀的方式。

5 全局变量尽量只在模块内有效，类似C语言中的static，加前缀一个下划线。

6 函数命名使用Pascal命名法（每一个单词的首字母都采用大写字母的命名格式）的方式，可以使用下划线。

7 常量命名使用全部大写的方式，可以使用下划线。

8 类的属性方法使用Pascal命名法（每一个单词的首字母都采用大写字母的命名格式）。

9 类的变量命名使用第一个单词全小写 + 后续单词Pascal命名法（每一个单词的首字母都采用大写字母的命名格式）。

10 类的属性有3种作用域public、non-public和subclass API，可以理解成C++中的public、private、protected，non-public属性前，前缀一条下划线。

11 类的属性若与关键字字冲突，后缀一下划线，尽量不要使用缩略等其他方式。

12 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。

13 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。

**七 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，但是Jython中却非常低，所以应该采用.join()的方式。

2 尽可能使用‘is’‘is not’取代‘==’，比如if x is not None 要优于if x。

3 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。

4 异常中不要使用裸露的except，except后跟具体的exceptions。

5 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)

要优于

try:

# Too broad!

return handle\_value(collection[key])

except KeyError:

# Will also catch KeyError raised by handle\_value()

return key\_not\_found(key)

6 使用startswith() and endswith()代替切片进行序列前缀或后缀的检查。比如

Yes: if foo.startswith(‘bar’):优于

No: if foo[:3] == ‘bar’:

7 使用isinstance()比较对象的类型。比如

Yes: if isinstance(obj, int): 优于

No: if type(obj) is type(1):

8 判断序列空或不空，有如下规则

Yes: if not seq:

if seq:

优于

No: if len(seq)

if not len(seq)

9 字符串不要以空格收尾。

10 二进制数据判断使用 if boolvalue的方式。