**Python编程规范v2**

**编码**

* 所有的 Python 脚本文件都应在文件头标上如下标识或其兼容格式的标识：



* 设置编辑器，默认保存为 utf-8 格式。

**注释**

* 业界普遍认同 Python 的注释分为两种的概念，一种是由 # 开头的“真正的”注释，另一种是 docstrings。前者表明为何选择当前实现以及这种实现的原理和难点，后者表明如何使用这个包、模块、类、函数（方法），甚至包括使用示例和单元测试。
* 坚持适当注释原则。对不存在技术难点的代码不需要注释，对存在技术难点的代码必须注释。
* 每一个包、模块、类、函数（方法）必须在前面使用注释进行功能描述说明。每一个超过50行的算法，必须用简单注释阐述。
* 注释建议使用中文， 除非自认为英语表述能力较强。

**格式**

**缩进**

* Python 依赖缩进来确定代码块的层次，行首空白符主要有两种：tab 和空格，但严禁两者混用。
* 公司内部强制使用 4 个空格进行缩进。
* 所有编辑器，ide 需要自行设定，对tab 进行4个空格的转化。

**空格**

* 空格在 Python 代码中是有意义的，因为 Python 的语法依赖于缩进，在行首的空格称为前导空格。在这一节不讨论前导空格相关的内容，只讨论非前导空格。非前导空格在 Python 代码中没有意义，但适当地加入非前导空格可以增进代码的可读性。
* 在二元算术、逻辑运算符前后加空格，如：



* “:”用在行尾时前后皆不加空格，如分枝、循环、函数和类定义语言；用在非行尾时两端加空格，如 dict 对象的定义：



* 括号（含圆括号、方括号和花括号）前后不加空格，如：



而不是



* 逗号后面加一个空格，前面不加空格；

**空行**

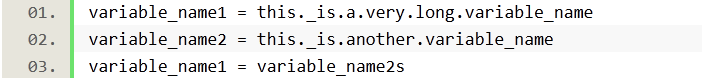
* 适当的空行有利于增加代码的可读性，加空行可以参考如下几个准则：
* 在类、函数的定义间加空行；
* 在 import 不同种类的模块间加空行；
* 在函数中的逻辑段落间加空行，即把相关的代码紧凑写在一起，作为一个逻辑段落，段落间以空行分隔；

**断行**

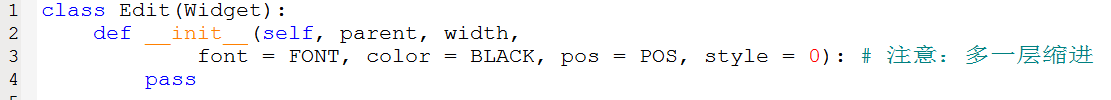
* 尽管现在的宽屏显示器已经可以单屏显示超过 256 列字符，但本规范仍然坚持行的最大长度不得超过 78 个字符的标准。折叠长行的方法有以下几种方法：
* 为长变量名换一个短名，如：

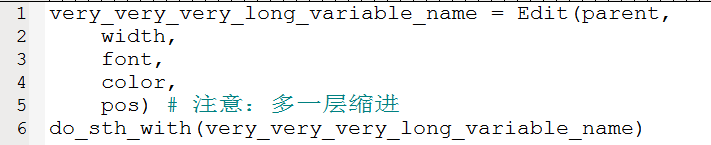


应改为：

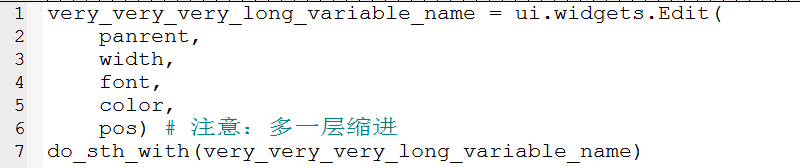


* 在括号（包括圆括号、方括号和花括号）内换行，如：

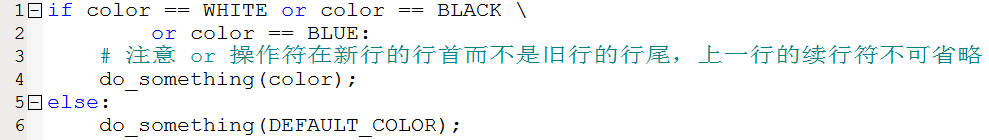
 或：



* 如果行长到连第一个括号内的参数都放不下，则每个元素都单独占一行：



* 在长行加入续行符强行断行，断行的位置应在操作符前，且换行后多一个缩进，以使维护人员看代码的时候看到代码行首即可判定这里存在换行，如：



**命名**

* 一致的命名可以给开发人员减少许多麻烦，而恰如其分的命名则可以大幅提高代码的可读性，降低维护成本。

**常量**

* 常量名所有字母大写，由下划线连接各个单词，如：



**变量**

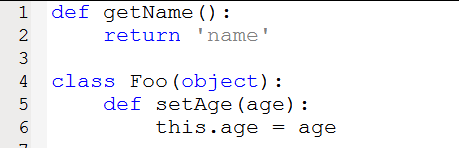
* 变量名全部小写，由下划线连接各个单词，如：



* 不论是类成员变量还是全局变量，均不使用 m 或 g 前缀。私有类成员使用单一下划线前缀标识，多定义公开成员，少定义私有成员。
* 变量名不应带有类型信息，因为 Python 是动态类型语言。如 iValue、names\_list、dict\_obj 等都是不好的命名。

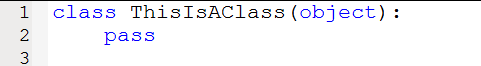
**函数**

* 函数名的命名规则采用首写动词小写，后续首写字母大写的方式，如：



**类**

* 类名单词首字母大写，不使用下划线连接单词，也不加入 C、T 等前缀。如：



**模块**

* 模块的命名规范与类相同。

**包**

* 包的命名规范与模块相同。

**缩写**

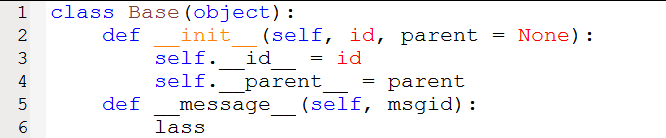
* 命名应当尽量使用全拼写的单词，缩写的情况有如下两种：
* 常用的缩写，如 XML、ID等，在命名时也应只大写首字母，如：



* 命名中含有长单词，对某个单词进行缩写。这时应使用约定成俗的缩写方式，如去除元音、包含辅音的首字符等方式，例如：
* function 缩写为 fn
* text 缩写为 txt
* object 缩写为 obj
* count 缩写为 cnt
* number 缩写为 num，等。

**特定命名方式**

* 主要是指 \_\_xxx\_\_ 形式的系统保留字命名法。项目中也可以使用这种命名，它的意义在于这种形式的变量是只读的，这种形式的类成员函数尽量不要重载。如：



其中 \_\_id\_\_、\_\_parent\_\_ 和 \_\_message\_\_ 都采用了系统保留字命名法。

**语句**

**import**

* import 语句有以下几个原则需要遵守：
* import 的次序，先 import Python 内置模块，再 import 第三方模块，最后 import 自己开发的项目中的其它模块；这几种模块中用空行分隔开来。
* 一条 import 语句 import 一个模块。
* 当从模块中 import 多个对象且超过一行时，使用如下断行法（此语法 py2.5 以上版本才支持）：



* 不要使用 from module import \*，除非是 import 常量定义模块或其它你确保不会出现命名空间冲突的模块。

**分枝和循环**

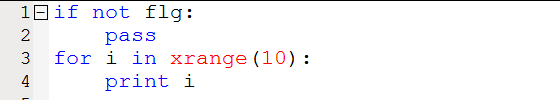
* 对于分枝和循环，有如下几点需要注意的：
* 不要写成一行，如：

****

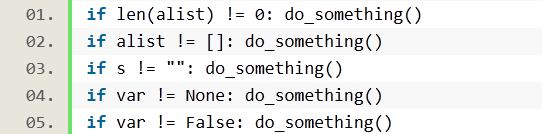
和



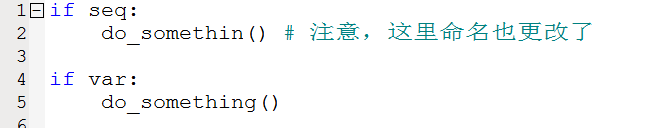
都不是好代码，应写成



* 条件表达式的编写应该足够 pythonic，如以下形式的条件表达式是拙劣的：



上面的语句应该写成：



* 用得着的时候多使用循环语句的 else 分句，以简化代码。

**已有代码**

* 对于项目中已有的代码，可能因为历史遗留原因不符合本规范，应当看作可以容忍的特例，允许存在；但不应在新的代码中延续旧的风格。
* 对于第三方模块，可能不符合本规范，也应看作可以容忍的特例，允许存在；但不应在新的代码中使用第三方模块的风格。
* tab 与空格混用的缩进是**'''不可容忍'''**的，在运行项目时应使用 -t 或 -tt 选项排查这种可能性存在。出现混用的情况时，如果是公司开发的基础类库代码，应当通知类库维护人员修改；第三方模块则可以通过提交 patch 等方式敦促开发者修正问题。

**已有风格**

* 开发人员往往在加入项目之前已经形成自有的编码风格，加入项目后应以本规范为准编写代码。特别是匈牙利命名法，因为带有类型信息，并不适合 Python 编程，不应在 Python 项目中应用。