Python 语言代码风格

1、概览

1.1 代码布局

1.1.1 缩进

a. 每级缩进用 4 个空格。

b. 为避免与旧代码混淆，可继续采用 8 个空格宽的 tab 缩进。

c. 绝不要混合使用 tab 和空格。

d. 调用 Python 命令行解释器时使用-t 选项，可对代码中不合法的混用制表符和空格发出警告 (warnings)，使用-tt 时警告将变成错误。

e. 对新项目，强烈推荐只用空格，而不是用 tabs。

1.1.2 最大行宽

a. 限制所有行的最大行宽为 79 个字符。

b. 对顺序排放的大块文本(文档字符 串或注释)，将长度限制在 72 个字符。

c. 折叠长行的首选方法是使用 Python 支持的圆括号、方括号和花括号内行延续。如果需要，可以在表达式周围增加一对额外的圆括号，但是，有时使用反斜杠看起来更好。

```
with open('/path/to/some/file/you/want/to/read') as file_1, \
     open('/path/to/some/file/being/written', 'w') as file_2:
    file_2.write(file_1.read())
```

1.1.3 空行

a. 用两行空行分割顶层函数和类的定义。

b. 类内方法的定义用单个空行分割。

c. 额外的空行可被用于分割相关函数群。在一组相关的单句中间可以省略空行

d. 在函数中使用空行时，请谨慎的用于表示一个逻辑段落 (logical sections)。

e. 接受 contol-L(即^L)换页符作为空白符；许多工具视这些字符为分页符，因此在你的文件中，可以用它们来为相关代码片段分页。

1.1.4 编码

a. 核心 Python 发布中的代码应该始终使用 UTF-8 (Python3) 或 ASCII (Python2)

b. 使用 UTF-8 (Python3) 或 ASCII (Python2) 的文件不需要译码

c. 仅当注释或文档字符串涉及作者名字需要 Latin-1 时，Latin-1 才被使用；使用\x 转义字符是在字符串中包含非 ASCII 数据的首选方法。

1.1.5 导入（Imports）

a. 一行中应该只有一个导入。但允许这样形式的导入：

```
from subprocess import Popen, PIPE
```

b. 导入应放置在文件的顶部，仅在模块注释和文档字符串之后，在模块的全局变量和常量之前。

c. 导入顺序：标准库的导入，相关的第三方包的导入，本地应用/库的特定导入；每个导入间有一个空行；把任何相关__all__说明的放在 imports 之后。

d. 对所有导入总是使用包的绝对路径

1.2 引号字符

单引号字符和双引号字符是一样的，如果一个字符串包含单引号或双引号，使用另一个引号来标示字符串，以避免转义问题

1.3 在表达式和语句中的空格

1.3.1 避免在以下情况下使用不必要的空格

a. 紧挨着圆括号、方括号和花括号
b. 紧贴在逗号、分号或冒号前
c. 紧贴着函数调用的参数列表前的开式括号
d. 紧贴在索引或切片 (indexing or slicing) 开始的开式括号前
e. 在用于指定关键字参数或默认参数值的'='号周围

1.3.2 其它推荐

a. 始终在这些二元运算符两边放置一个空格:
assignment (=), augmented assignment (+=, -= etc.), comparisons (==, <, >, !=, <>, <=, >=, in, not in, is, is not), Booleans (and, or, not).

b. 在数学运算符两边使用空格，如果表达式中有多个运算符，应只在最低优先级的运算符两边使用空格

c. 空格不应连续使用两个及以上

1.4 注释

注释应该是完整的句子。如果注释是一个短语或句子，首字母应该大写，除非它是一个以小写字母开头的标识符 (永远不要修改标识符的大小写)。

a. 如果注释很短，可以省略末尾的句号。

b. 每个句子应该以句号结尾。

c. 应该在结束语句的句点后使用两个空格。

d. 用英语书写时，断词和空格是可用的。

e. 请用英语书写你的注释。

f. 注释块通常应用于跟随其后的一些 (或者全部) 代码，并和这些代码有着相同的缩进层次。注释块中每行以'#'和一个空格开始 (除非它是注释内的缩进文本)。

g. 注释块内的段落以仅含单个'#'的行分割。

h. 一个行内注释是和语句在同一行的注释。行内注释应该至少用两个空格和语句分开。它们应该以一个'#'和单个空格开始。

1.5 命名规则

1.5.1 应该避免的名称

a. 单字符名称, 除了计数器和迭代器.

b. 包/模块名中的连字符(-)

c. 双下划线开头并结尾的名称(Python 保留，例如__init__)

1.5.2 命名约定

a. 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.

b. 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).

c. 用双下划线(__)开头的实例变量或方法表示类内私有.

d. 将相关的类和顶级函数放在同一个模块里。 不像 Jav，没必要限制一个类一个模块。

e. 对类名使用大写字母开头的单词(如 CapWords, 即 Pascal 风格)，但是模块名应该用小写加下划线的方式(如 lower_with_under.py)。尽管已经有很多现存的模块使用类似于 CapWords.py 这样的命名，但现在已经不鼓励这样做, 因为如果模块名碰巧和类名一致,，这会让人困扰。

1.5.3 Python 之父 Guido 推荐的规范

Type	Public	Internal
Modules	lower_with_under	_lower_with_under
Packages	lower_with_under	
Classes	CapWords	_CapWords
Exceptions	CapWords	
Functions	lower_with_under()	_lower_with_under()
Global/Class Constants	CAPS_WITH_UNDER	_CAPS_WITH_UNDER
Global/Class Variables	lower_with_under	_lower_with_under
Instance Variables	lower_with_under	_lower_with_under (protected) or __lower_with_under (private)
Method Names	lower_with_under()	_lower_with_under() (protected) or __lower_with_under() (private)