**1 变量  
  
常量** : 大写加下划线

USER\_CONSTANT

对于不会发生改变的全局变量，使用大写加下划线。  
 **私有变量** : 小写和一个前导下划线

\_private\_value

Python 中不存在私有变量一说，若是遇到需要保护的变量，使用小写和一个前导下划线。但这只是程序员之间的一个约定，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。  
  
**内置变量** : 小写，两个前导下划线和两个后置下划线

\_\_class\_\_

两个前导下划线会导致变量在解释期间被更名。这是为了避免内置变量和其他变量产生冲突。用户定义的变量要严格避免这种风格。以免导致混乱。  
  
  
**2 函数和方法**  
总体而言应该使用，小写和下划线。但有些比较老的库使用的是混合大小写，即首单词小写，之后每个单词第一个字母大写，其余小写。但现在，小写和下划线已成为规范。  
  
**私有方法** ： 小写和一个前导下划线

def \_secrete(self):  
    print "don't test me."

这里和私有变量一样，并不是真正的私有访问权限。同时也应该注意一般函数不要使用两个前导下划线(当遇到两个前导下划线时，Python 的名称改编特性将发挥作用)。特殊函数后面会提及。  
  
**特殊方法** ： 小写和两个前导下划线，两个后置下划线

def \_\_add\_\_(self, other):  
    return int.\_\_add\_\_(other)

这种风格只应用于特殊函数，比如操作符重载等。  
  
**函数参数** : 小写和下划线，缺省值等号两边无空格

def connect(self, user=None):  
    self.\_user = user

**3 类**

类总是使用驼峰格式命名，即所有单词首字母大写其余字母小写。类名应该简明，精确，并足以从中理解类所完成的工作。常见的一个方法是使用表示其类型或者特性的后缀，例如:  
SQLEngine  
MimeTypes  
  
对于基类而言，可以使用一个 Base 或者 Abstract 前缀  
BaseCookie  
AbstractGroup

class UserProfile(object):  
    def \_\_init\_\_(self, profile):  
        return self.\_profile = profile  
  
    def profile(self):  
        return self.\_profile

**4 模块和包**

除特殊模块 \_\_init\_\_ 之外，模块名称都使用不带下划线的小写字母。  
若是它们实现一个协议，那么通常使用lib为后缀，例如:  
import smtplib

import os  
import sys

**5 关于参数**  
5.1 不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。  
  
5.2 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。  
  
  
**6 其他**  
6.1 使用 has 或 is 前缀命名布尔元素

is\_connect = True  
has\_member = False

6.2 用复数形式命名序列

members = ['user\_1', 'user\_2']

6.3 用显式名称命名字典

person\_address = {'user\_1':'10 road WD', 'user\_2' : '20 street huafu'}

6.4 避免通用名称  
诸如 list, dict, sequence 或者 element 这样的名称应该避免。  
  
6.5 避免现有名称  
诸如 os, sys 这种系统已经存在的名称应该避免。  
  
  
**7 一些数字**  
一行列数 : PEP 8 规定为 **79 列**，这有些苛刻了。根据自己的情况，比如不要超过满屏时编辑器的显示列数。这样就可以在不动水平游标的情况下，方便的查看代码。

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**。

**8 验证脚本**

可以安装一个 pep8 脚本用于验证你的代码风格是否符合 PEP8。

http://pypi.python.org/pypi/pep8/#downloads

>>easy\_install pep8

>>pep8 -r --ignore E501 Test.py

这个命令行的意思是，重复打出错误，并且忽略 501 错误(代码超过 79 行)。

Ubuntu上的安装可以用命令

sudo apt-get install pep8