来自：http://blog.csdn.net/kellyseeme/article/details/50644893

Python的pep8-代码规范

1、    代码布局设计

1.1    缩进

A、   使用四个空格来进行缩进

B、   换行的时候可以使用反斜杠，最好的方法是使用园括号，在使用反斜杠的时候，在反斜杠的后直接回车，不能有任何空格存在

比较好的做法如下：

对准开始的分隔符：

|  |
| --- |
| # Aligned with opening delimiter.  foo = long\_function\_name(var\_one, var\_two,               var\_three, var\_four) |

包含更多的缩进表示是剩余部分：

|  |
| --- |
| # More indentation included to distinguish this from the rest.  def long\_function\_name(          var\_one, var\_two, var\_three,          var\_four):      print(var\_one) |

         悬挂缩进应该添加一个级别：

|  |
| --- |
| # Hanging indents should add a level.  foo = long\_function\_name(      var\_one, var\_two,      var\_three, var\_four) |

比较差的做法如下：（代码同样是可以运行的）

|  |
| --- |
| # Arguments on first line forbidden when not using vertical alignment.—未使用垂直对齐  foo = long\_function\_name(var\_one, var\_two,      var\_three, var\_four)    # Further indentation required as indentation is not distinguishable.（未使用缩进来表示每一层级）  def long\_function\_name(      var\_one, var\_two, var\_three,      var\_four):      print(var\_one) |

对于续行来说，四个空格的缩进是可选的。

         可选的如下：

|  |
| --- |
| # Hanging indents \*may\* be indented to other than 4 spaces.悬挂缩进的时候可以不是四个空格  foo = long\_function\_name(    var\_one, var\_two,    var\_three, var\_four) |

         当使用if语句的时候，如果条件恰好的缩进为四个空格空格，那么导致后面的语句的缩进也是四个空格，那么这种情况下是可以接受的，如下所示：

         没有额外的缩进：

|  |
| --- |
| # No extra indentation.  if (this\_is\_one\_thing and      that\_is\_another\_thing):      do\_something() |

         添加一个注释来进行分割缩进，做到语法高亮显示：

|  |
| --- |
| # Add a comment, which will provide some distinction in editors  # supporting syntax highlighting.  if (this\_is\_one\_thing and      that\_is\_another\_thing):      # Since both conditions are true, we can frobnicate.      do\_something() |

         在续行中添加额外的缩进：

|  |
| --- |
| # Add some extra indentation on the conditional continuation line.  if (this\_is\_one\_thing          and that\_is\_another\_thing):      do\_something() |

         成对的小括号，中括号在多行的结构中可以写成多行，然后括号在第一个不为空白的位置结束。如下：

|  |
| --- |
| my\_list = [      1, 2, 3,      4, 5, 6,      ]  result = some\_function\_that\_takes\_arguments(      'a', 'b', 'c',      'd', 'e', 'f',      ) |

         或者对齐第一个字符的位置结束，如下：

|  |
| --- |
| my\_list = [      1, 2, 3,      4, 5, 6,  ]  result = some\_function\_that\_takes\_arguments(      'a', 'b', 'c',      'd', 'e', 'f',  ) |

1.2 tab和空格的选择

关于tab的空格的选择，在python2中是可以混用的，但是在python3中，只能用一种风格。

1.3 最大行长度

行的最大长度为79个字符

在书写文档或者是注释的时候，行长度应该控制在72个字符。

反斜杠在有的时候是适用的，例如在参数很长，但是不能隐式的使用多行的时候，如下反斜杠的使用：

|  |
| --- |
| 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()) |

确保在合适的时候将连续的行进行分开，最好的位置是操作符之后，而不是在操作符之前，如下：

|  |
| --- |
| class Rectangle(Blob):        def \_\_init\_\_(self, width, height,                   color='black', emphasis=None, highlight=0):          if (width == 0 and height == 0 and                  color == 'red' and emphasis == 'strong' or                  highlight > 100):              raise ValueError("sorry, you lose")          if width == 0 and height == 0 and (color == 'red' or                                             emphasis is None):              raise ValueError("values are %s, %s" %                               (width, height))          Blob.\_\_init\_\_(self, width, height,                        color, emphasis, highlight) |

1.4 空行

Top level函数和类的定义的时候，空两行。

类中方法的定义空一行。

在函数中谨慎使用空行来表示相关的逻辑段。

无关的函数之间用一个空行进行分割。

1.5 源文件编码

在源文件中一直使用utf-8编码，在python2中使用ascll编码。

文件，在python2 中使用ascll编码，在python3中使用utf-8编码

1.6 导入

Import经常使用单独的行，如下：

|  |
| --- |
| import os  import sys |

或者使用如下的方式：

|  |
| --- |
| § from subprocess import Popen, PIPE |

Import总是在文件的最上行，在模块的注释和docstring之后，在模块的全局变量之前。

Import可以按照以下顺序进行组织：

         A标准类库import

         B第三方import

         C本地类库import

在每个组导入之后，可以用空行进行分割

把所有\_\_all\_\_相关类型的声明放在import之后

推荐使用绝对导入，可读性强，如下：

|  |
| --- |
| import mypkg.sibling  from mypkg import sibling  from mypkg.sibling import example |

对于复杂的封装布局来说，相对导入也是可以接受的，主要是使用绝对导入的时候路径太长，如下：

|  |
| --- |
| from . import sibling  from .sibling import example |

当导入一个类的时候，可以使用如下的方式：

|  |
| --- |
| from myclass import MyClass  from foo.bar.yourclass import YourClass |

当以上的写法导致本地名称冲突，可以写成如下：

|  |
| --- |
| import myclass  import foo.bar.yourclass |

并且使用"myclass.MyClass" and"foo.bar.yourclass.YourClass"。

在导入模块的时候，应该避免通配符的存在，如下：

|  |
| --- |
| from <module> import \* |

2、    字符串引号

在对于字符串的标示中，使用双引号还是单引号是没有区别的，主要就是两者混合使用从而避免反斜杠的出现。

3、    在表达式和语句中使用空格

3.1 避免使用空格情况

A． 在小括号，中括号，大括号中避免使用空格

|  |
| --- |
| Yes: spam(ham[1], {eggs: 2})  No:  spam( ham[ 1 ], { eggs: 2 } ) |

B． 在逗号，分好，冒号之前不需要空格

|  |
| --- |
| Yes: if x == 4: print x, y; x, y = y, x  No:  if x == 4 : print x , y ; x , y = y , x |

C． 在切片的时候，避免使用空格，在扩展的切片中，必须使用相同的空格个数，如下所示：

|  |
| --- |
| Yes:  ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]  ham[lower:upper], ham[lower:upper:], ham[lower::step]  ham[lower+offset : upper+offset]  ham[: upper\_fn(x) : step\_fn(x)], ham[:: step\_fn(x)]  ham[lower + offset : upper + offset]  No:  ham[lower + offset:upper + offset]  ham[1: 9], ham[1 :9], ham[1:9 :3]  ham[lower : : upper]  ham[ : upper] |

D．函数的左括号前不要添加空格：

|  |
| --- |
| Yes: spam(1)  No:  spam (1) |

E． 中括号前不要添加空格

|  |
| --- |
| Yes: dct['key'] = lst[index]  No:  dct ['key'] = lst [index] |

F． 操作符左右各一个空格，不要为了追求一致从而添加空格个数

|  |
| --- |
| Yes:  x = 1  y = 2  long\_variable = 3  No:  x             = 1  y             = 2  long\_variable = 3 |

3.2 其他建议

A． 避免在任何结尾添加空白。

B． 在下列操作符中左右各留空白

|  |
| --- |
| assignment ( = ), augmented assignment ( += , -= etc.), comparisons ( == , < , > , != , <> , <= , >= , in , not in , is , is not ), Booleans ( and , or , not ) |

C． 如果操作符优先级不同，注意在操作符左右留空白，特别是高优先级和低优先级的

|  |
| --- |
| i = i + 1  submitted += 1  x = x\*2 - 1  hypot2 = x\*x + y\*y  c = (a+b) \* (a-b)  No:  i=i+1  submitted +=1  x = x \* 2 - 1  hypot2 = x \* x + y \* y  c = (a + b) \* (a - b) |

D． 在使用函数的时候，赋值和默认值之间不需要空格

|  |
| --- |
| Yes:  def complex(real, imag=0.0):      return magic(r=real, i=imag)  No:  def complex(real, imag = 0.0):      return magic(r = real, i = imag) |

E． 不要将多语句写在同一行

|  |
| --- |
| Rather not:  if foo == 'blah': do\_blah\_thing()  for x in lst: total += x  while t < 10: t = delay()  Definitely not:  if foo == 'blah': do\_blah\_thing()  else: do\_non\_blah\_thing()    try: something()  finally: cleanup()    do\_one(); do\_two(); do\_three(long, argument,                               list, like, this)    if foo == 'blah': one(); two(); three() |

4、    注释

在修改的代码的时候，务必修改注释。

注释必须是英文，最好是完整的句子，首字母大写

4.1 块注释

在一段代码前增加注释，在#后添加一个空格，段落之间只有一个#作为行间隔

|  |
| --- |
| # Description : Module config.  #  # Input : None  #  # Output : None |

4.2 行注释

在使用行注释的时候，在代码句子结束之后至少两个空格，然后用#开头后跟一个空格

|  |
| --- |
| x = x + 1                 # Increment x  But sometimes, this is useful:  x = x + 1                 # Compensate for border |

在上面例子中，表示不要使用无效注释，主要是说明其目的

4.3 文档注释

在所有的公共模块，函数，类，方法中加入文档注释，这些注释写在def之后。

在进行多行注释的时候，注意“”“结束的时候，必须独占一行，如下：

|  |
| --- |
| """Return a foobang    Optional plotz says to frobnicate the bizbaz first.  """ |

当文档注释是一行的时候，确保开始的““”和“”“在同一行中。

5、    命名规范

使用单独的小写字母（b）

使用单独的大写字母（B）

使用小写字母（lowercase）

使用小写字母和下划线（lower\_case\_with\_underscores）

使用大写字母（UPPERCASE）

使用大写字母和下划线（UPPER\_CASE\_WITH\_UPPERCASE）

驼峰式写法（CamelCase）：在使用缩写的时候，大写优于小写例如HTTPServer优于HttpServer

首字母大写，然后使用下划线是一种丑陋的写法

5.1 避免使用的名称

在写变量的时候，尽量避免小写的l和大写字母O和大写字母I，主要原因是容易和数字中1,0相混淆

5.2 包和模块名称

模块尽量使用简短的全部小写的名称，如果可以增加可读性那么可以使用下划线，python的包不推荐使用下划线，但是在引用其他语言写的扩展包中可以使用下划线来表示区分

5.3 类名称

类名称主要遵循为CapWords约定，表示为首字母大写

5.4 异常名称

异常归于类，从而也要遵循类名的规范，主要是在后缀上必须添加“Error“

5.4 全局变量名

全局变量只在模块类有效，和function命名相同

5.5 方法名称

方法名称全部为小写，下划线是可选的（在增加可读性的基础上使用）

5.6 方法变量

类的方法第一个参数总是self

类方法的静态变量总是为crs

如果一个方法的参数和保留字相冲突，那么在后面添加下划线进行区分

5.7 常量

常量命名全部使用大写，可以使用下划线进行分割

6、    编码建议

单独比较的时候使用is或者is not，不要使用==进行比较。

当实现比较的方法的时候，最好全部实现

\_\_eq\_\_ , \_\_ne\_\_ ,\_\_lt\_\_ , \_\_le\_\_ , \_\_gt\_\_ , \_\_ge\_\_ )，而不要单独实现一个。

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):

来自http://blog.sae.sina.com.cn/archives/4781

## PEP8 Python 编码规范整理

* 07/17. 2014

决定开始Python之路了，利用业余时间，争取更深入学习Python。编程语言不是艺术，而是工作或者说是工具，所以整理并遵循一套编码规范是十分必要的。所以今天下午我根据PEP 8整理了一份，以后都照此编码了，还会持续更新。

**PEP8 Python 编码规范**

**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Tap，更不能混合使用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。

**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.’，都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 不要将多句语句写在同一行，尽管使用‘；’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。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

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.

"""

**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，使用全部小写的方式，可以使用下划线。  
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CapWords的方式，模块内部使用的类采用\_CapWords的方式。  
5 异常命名使用CapWords+Error后缀的方式。  
6 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。  
7 函数命名使用全部小写的方式，可以使用下划线。  
8 常量命名使用全部大写的方式，可以使用下划线。  
9 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。  
9 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名字冲突，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，都是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。