目录

[ 命名(naming) 2](#_Toc356508041)

[ 包(package) 2](#_Toc356508042)

[ 模块(module) 2](#_Toc356508043)

[ 类(class) 2](#_Toc356508044)

[ 类方法(method) 2](#_Toc356508045)

[ 函数(function) 2](#_Toc356508046)

[ 变量(variables) 2](#_Toc356508047)

[ 常量(constant) 2](#_Toc356508048)

[ 设计(design) 3](#_Toc356508049)

[ 横向(Horizontal) 3](#_Toc356508050)

[**空格(space)** 3](#_Toc356508051)

[换行(enter) 3](#_Toc356508052)

[缩进(indentation) 3](#_Toc356508053)

[ 纵向(vertical) 3](#_Toc356508054)

[空行(blank-lines) 3](#_Toc356508055)

[行数(lines) 3](#_Toc356508056)

[ 语句(sentence) 4](#_Toc356508057)

[ 类的继承 4](#_Toc356508058)

[ 访问控制 4](#_Toc356508059)

[public 4](#_Toc356508060)

[protected 4](#_Toc356508061)

[private 4](#_Toc356508062)

[ 字符串 4](#_Toc356508063)

[ main()方法 4](#_Toc356508064)

[ 注释 5](#_Toc356508065)

[ 文档字符串 5](#_Toc356508066)

[ 模块 5](#_Toc356508067)

[ 方法和函数 5](#_Toc356508068)

[ 块和行注释 5](#_Toc356508069)

[ TODO注解 5](#_Toc356508070)

[ Python解释器及编码注释 6](#_Toc356508071)

[ 导入(import) 6](#_Toc356508072)

Python 编码风格规范

# 命名(naming)

包含各个元素的命名规范, 一般使用Pascal 或者 caMel 两种类型.

Pascal: 单词首字母大写, 其余字母小写. 例如: PascalType.

caMel: 第一个单词首字母小写, 其余单词首字母大写, 余下字母小写. 例如: camelType.

在所有的命名中, 尽量避免使用下划线”\_”, 常量除外.

## 包(package)

使用caMel, 尽量使用单个单词

## 模块(module)

使用caMel, 尽量使用与模块中类或者方法不一样的单词或单词组合

## 类(class)

使用Pascal, 用名词语义构成, 使用1-3个单词组成, 突出类的作用

## 类方法(method)

使用caMel, 用动词语义构成, 可使用1个动词作为第1个单词, 之后伴随2-4个单词组成, 突出类方法的功能.

## 函数(function)

使用caMel, 用动词语义构成, 可使用1个动词作为第1个单词, 之后伴随2-4个单词组成, 突出函数的功能.

## 变量(variables)

使用caMel, 用名词语义构成, 可使用1-3个单词构成, 同时可以跟随类似, List, Dic, Tuple等等表明此变量的类型.

## 常量(constant)

常量不同于其他命名, 所有字母均用大写, 单词与单词之间可用下划线”\_”分隔.

# 设计(design)

## 横向(horizontal)

### 空格(space)

1. 圆括号, 方括号, 大括号里面不加空格, 例如:
   * + - 1. func = function(a, b, c)
2. 逗号, 分号, 冒号后面必须加空格 例如:
   * + - 1. func = function(a, b, c)
         2. print x, y
         3. x, y = y, x
3. 二元运算符, 赋值符, 比较符, 布尔运算符, 两边加上空格, 例如:
   1. x == 1:
   2. x is None and y is None

但是, 参数输入和默认参数时, 不加空格, 例如:

1. def function(a, b=1, c=2)

### 换行(enter)

1. 一行最多不超过80个字符(国际标准80个字符).
2. 换行遵循: 逗号之后换行, 换行后请于之前的元素对齐, 例如:
3. foo = longFunctionName(var\_one, var\_two,  
                           var\_three, var\_four)

### 缩进(indentation)

1. 缩进一律使用四个空格, 不使用制表符.

## 纵向(vertical)

### 空行(blank-lines)

1. 方法与方法之间, 使用2行空行.
2. 方法中变量声明与语句之间, 使用1行空行.
3. 方法中不同的逻辑块之间, 使用1行空行.
4. 注释与它注释的语句间, 不使用空行.

### 行数(lines)

一个模块不超过1000行.

# 语句(sentence)

## 类的继承

如果一个类不继承自其它类, 就显式的从object继承. 嵌套类也一样.

## 访问控制

### public

公有方法或者变量按普通命名, 使用caMel

### protected

保护方法或者变量按下划线”\_”+普通命名, 例如:

1. def \_protectedMethod(self, a, b):

pass

1. self.\_protectedVariables = ‘a’

### private

私有方法或者变量按双下划线”\_\_”+普通命名, 例如:

1. def \_\_privateMethod(self, a, b):

pass

1. self.\_\_privateVariables = ‘a’

## 字符串

1. 用”%”操作符格式化字符串, 即使参数都是字符串. 不过也不能一概而论, 你需要在+和%之间好好判定, 例如:
   1. x = a + b
   2. x = ‘%s, %s!’ % (a, b)
   3. x = ‘name: %s; score: %d’ % (name, n)
2. 避免在循环中用+和+=操作符来累加字符串. 造成创建不必要的临时对象, 可使用加入列表的方式, 在循环结束后用 .join连接列表.
3. 多行字符串使用三重双引号而非三重单引号, 但是必须注意多行字符串的格式问题

## main()方法

每一个模块, 无论是否需要单独运行, 都必须加入main方法和if \_\_name\_\_方式, 例如:

def main():

…

if \_\_name\_\_ == ‘\_\_main\_\_’:

main()

# 注释

## 文档字符串

用三重双引号, 详细格式参照pydoc文档

## 模块

每个文件应该包含下列项, 依次是:

1. 版权声明(例如, Copyright 2008 Google Inc.)
2. 一个许可样板. 根据项目使用的许可(例如, Apache 2.0, BSD, LGPL, GPL), 选择合适的样板
3. 作者声明, 标识文件的原作者

## 方法和函数

如果不是既显然又简短, 任何函数或方法都需要一个文档字符串. 而且, 任何外部可访问的函数或方法, 不管多短多简单, 都需要文档字符串. 文档字符串应该包含函数做什么, 以及输入和输出的详细描述. 通常, 不应该描述”怎么做”, 除非是一些复杂的算法. 对于技巧性的代码, 块注释或者行内注释是最重要的. 文档字符串应该提供足够的信息, 当别人编写代码调用该函数时, 他不需要看一行代码, 只要看文档字符串就可以了. 应该给参数单独写文档. 在冒号后跟上解释, 而且应该用统一的悬挂式2或4空格缩进. 文档字符串应该在需要特定类型的地方指定期望的类型. “Raise:”部分应该列出该函数可能触发的所有异常. 生成器函数的文档字符串应该用”Yields:”而非”Returns:”.

## 块和行注释

最需要写注释的是代码中那些技巧性的部分. 如果你在下次代码走查的时候必须解释一下, 那么你应该现在就给它写注释. 对于复杂的操作, 应该在其操作开始前写上若干行注释. 对于不是一目了然的代码, 应在其行尾添加注释.

为了提高可读性, 注释应该至少离开代码2个空格.

另一方面, 绝不要描述代码. 假设阅读代码的人比你更懂Python, 他只是不知道你的代码要做什么.

## TODO注解

为临时代码使用TODO注释, 它是一种短期解决方案.

TODO注释应该在所有开头处包含”TODO”字符串, 紧跟着是用括号括起来的你的名字, email地址或其它标识符. 然后是一个可选的冒号. 接着必须有一行注释, 解释要做什么. 主要目的是为了有一个统一的TODO格式, 这样添加注释的人就可以搜索到(并可以按需提供更多细节). 写了TODO注释并不保证写的人会亲自解决问题.

例如:

1. # TODO(kl@gmail.com): Drop the use of "has\_key".
2. # TODO(Zeke) change this to use relations.

# 导入(import)

1. 每个导入应该独占一行, 例如:
   1. import os

import sys

* 1. import os, sys 是错误的

1. 导入顺序, 如下:
   1. 标准库导入
   2. 第三方导入

应用程序指定导入

# Python解释器及编码注释

每个python文件须包含以下2行:

1. linux下 #!/usr/bin/env python2.x

windows下 #!C:/Python2.x/python.exe

1. # -\*- coding:uft-8 -\*-

或者 # coding=utf-8