Python语言编码规范

目录

[Python语言编码规范 1](#_Toc11525)

[一，简明概述 2](#_Toc31014)

[1.编码 2](#_Toc30131)

[2.代码格式 2](#_Toc15474)

[2.1.缩进 2](#_Toc21874)

[2.2.行宽 2](#_Toc7015)

[2.3.引号 3](#_Toc19610)

[2.4. 空行 3](#_Toc19965)

[2.5.编码 3](#_Toc28755)

[3.import语句 3](#_Toc9601)

[4.空格 4](#_Toc16985)

[5.换行 5](#_Toc17371)

[6.docstring 5](#_Toc20387)

[二，注释 6](#_Toc29202)

1.注释 6

[1.1块注释 6](#_Toc5475)

[1.2行注释 6](#_Toc14278)

2.文档注释（Docstring） 6

[3.命名规范 8](#_Toc26372)

3.[1.模块 8](#_Toc9156)

3.[2.类名 8](#_Toc9483)

3.[3.函数 8](#_Toc30165)

3.[4.变量名 9](#_Toc21508)

3.[5.常量 9](#_Toc7294)

# 一，简明概述

## 1.编码

* 如无特殊情况，文件一律使用utf-8编码
* 如无特殊情况，文件头部必须加入# -\*-coding:utf-8-\*-

## 2.代码格式

### 2.1.缩进

* 统一使用4个空格进行缩进

### 2.2.行宽

每行代码最好不要超过80个字符(在特殊情况下可以超过80个，但最好不要超过120个)  
理由:

* 这在查看 side-by-side 的 diff 时很有帮助
* 方便在控制台下查看代码
* 太长可能是设计缺陷

### 2.3.引号

简单的说，自然语言使用双引号，机器语言使用单引号，因此在**代码**里多数应该使用**单引号**

* **自然语言使用双引号**"..."  
  例如错误信息，很多时候还是unicode，使用u"你好，世界"
* **机器标识使用单引号**'...'  
  例如字典里的key
* **正则表达式使用原生的双引号**r"..."
* **文档字符串(docstring)使用三个双引号**"""..."""

### 2.4. 空行

* 模块级函数和类之间空两行
* 类成员函数之间空一行

class A:

def \_\_init\_\_(self): # 类成员函数

pass

def hello(self): # 类成员函数

pass

def main(): # 模块级函数

pass

* 可以使用多个空行分割多组相关的函数
* 函数中使用空行分割出逻辑相关的代码

### 2.5.编码

* 文件使用utf-8编码
* 文件头部键入# -\*-coding:utf-8-\*-标识

## 3.import语句

* import语句应该分行书写

# 正确的写法import osimport sys

# 不推荐的写法import os,sys

# 正确的写法from subprocess import Popne, PIPE

* import语句应该使用**absolute** import

# 正确的写法from foo.bar import Bar

# 不推荐的写法from ..bar import Bar

* import 语句应该放在文件头部，置于模块说明及docstring之后，与全局变量之前
* import语句应该按照顺序排序，每组之间用一个空行分割

import osimport sys

import lxmlimport bs4

import requests

* 导入其他模块的类定义时，可以使用相对导入

from myclass import Myclass

* 如果发生命名冲突，则可以使用命名空间

import barimport foo.bar

bar.get()

foo.bar.get()

## 4.空格

* 在二元操作符两边各空一格[=,-,+=,-=,==,is not,and,in]

# 正确的写法

i = i + 1

y += 1

z = x \* x + y \* y

c = (a + b) + (a - b)

# 不推荐的写法

i=i+1

y+=1

z=x\*x+y\*y

c=(a+b)+(a-b)

* 函数的参数列表中，,之后加空格

# 正确的写法def foo(var1, var2):

pass

# 不推荐的写法def bar(var1,var2):

pass

* 函数的参数列表中，默认值等号两边不要加空格

# 正确的写法def foo(var1, var2=1):

pass

# 不推荐的写法def bar(var1, var2 = 1):

pass

* 左括号之后，右括号之前不要加多余的空格

# 正确的写法def foo(var1, var2):

pass

# 不推荐的写法def bar( var1, var2 ):

pass

* 字典对象的左括号之前不要添加多余的空格

# 正确的写法

dict1['key'] = 1

# 不推荐的写法

dict1 ['key'] = 1

* 不要为对齐赋值语句而使用额外的空格

# 正确的写法

x = 1

y = 2

long\_variable = 3

# 不推荐的写法

x = 1

y = 2

long\_variable = 3

## 5.换行

python支持括号内的换行，这时有2种情况  
1.第二行缩进到括号的起始处

foo = long\_function\_name(var1, var2

var3, var4)

2.第二行缩进4个空格，适用于起始括号就换行的情况

def long\_function\_name(

var1, var2

var3, var4):

print(var1)

使用反斜杠\换行,二元运算符+ .等应该出现在行末；长字符串也应该用此法换行

session.query(MyTable).\

filter\_by(id=1).\

one()

print 'Hello, '\

'%s %s!' %\

('Harry', 'Potter')

禁止复合语句；即一行中包含多个语句

#正确的写法

do\_first()

do\_second()

do\_third()

#不推荐的写法

do\_first();do\_second();do\_third();

if/for/while一定要换行:

#正确的写法if foo == 'blash':

do\_something()

#不推荐的写法if foo == 'blash':do\_something()

### 6.docstring

docstring中最规范的2点：  
1.所有的公共模块，类，函数，方法。私有方法不一定需要，但应该在def后提供一个块注释来说明  
2.docstring的结束符'''应该独占一行，除非这个doctring只占一行

"""Return a foobar

Optional plotz says to frobnicate the bizbaz first.

"""

"""Oneline docstring"""

## 二，注释

### 1.注释

#### 1.1.块注释

“#”号后空一格，段落件用空行分开（同样需要“#”号）

# 块注释

# 块注释

#

# 块注释

# 块注释

#### 1.2.行注释

至少使用两个空格和语句分开，注意不要使用无意义的注释

# 正确的写法

x = x + 1 # 边框加粗一个像素

# 不推荐的写法(无意义的注释)

x = x + 1 # x加1

### 2.文档注释(Docstring)

作为文档的Docstring一般出现在模块头部、函数和类的头部，这样在python中可以通过对象的**doc**对象获取文档.  
编辑器和IDE也可以根据Docstring给出自动提示.

* 文档注释以 """ 开头和结尾, 首行不换行, 如有多行, 末行必需换行, 以下是Google的docstring风格示例

# -\*- coding: utf-8 -\*-"""Example docstrings.

This module demonstrates documentation as specified by the `Google Python

Style Guide`\_. Docstrings may extend over multiple lines. Sections are created

with a section header and a colon followed by a block of indented text.

Example:

Examples can be given using either the ``Example`` or ``Examples``

sections. Sections support any reStructuredText formatting, including

literal blocks::

$ python example\_google.py

Section breaks are created by resuming unindented text. Section breaks

are also implicitly created anytime a new section starts.

"""

* 不要在文档注释复制函数定义原型, 而是具体描述其具体内容, 解释具体参数和返回值等

# 不推荐的写法(不要写函数原型等废话)def function(a, b):

"""function(a, b) -> list"""

... ...

# 正确的写法def function(a, b):

"""计算并返回a到b范围内数据的平均值"""

... ...

* 对函数参数、返回值等的说明采用numpy标准, 如下所示

def func(arg1, arg2):

"""在这里写函数的一句话总结(如: 计算平均值).

这里是具体描述.

参数

----------

arg1 : int

arg1的具体描述

arg2 : int

arg2的具体描述

返回值

-------

int

返回值的具体描述

参看

--------

otherfunc : 其它关联函数等...

示例

--------

示例使用doctest格式, 在`>>>`后的代码可以被文档测试工具作为测试用例自动运行

>>> a=[1,2,3]

>>> print [x + 3 for x in a]

[4, 5, 6]

"""

* 文档注释不限于中英文, 但不要中英文混用
* 文档注释不是越长越好, 通常一两句话能把情况说清楚即可
* 模块、公有类、公有方法, 能写文档注释的, 应该尽量写文档注释

## 3.命名规范

### 3.1.模块

* 模块尽量使用小写命名，首字母保持小写，尽量不要用下划线(除非多个单词，且数量不多的情况)

# 正确的模块名import decoderimport html\_parser

# 不推荐的模块名import Decoder

### 3.2.类名

* 类名使用驼峰(CamelCase)命名风格，首字母大写，私有类可用一个下划线开头

class Farm():

pass

class AnimalFarm(Farm):

pass

class \_PrivateFarm(Farm):

pass

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

### 3.3.函数

* 函数名一律小写，如有多个单词，用下划线隔开

def run():

pass

def run\_with\_env():

pass

* 私有函数在函数前加一个下划线\_

class Person():

def \_private\_func():

pass

### 3.4.变量名

* 变量名尽量小写, 如有多个单词，用下划线隔开

if \_\_name\_\_ == '\_\_main\_\_':

count = 0

school\_name = ''

－ 常量采用全大写，如有多个单词，使用下划线隔开

MAX\_CLIENT = 100

MAX\_CONNECTION = 1000

CONNECTION\_TIMEOUT = 600

### 3.5.常量

* 常量使用以下划线分隔的大写命名

MAX\_OVERFLOW = 100

Class FooBar:

def foo\_bar(self, print\_):

print(print\_)