# 代码质量规范

## 简介

本系统为基于Django的web系统，涉及到Python、js、css、html代码的撰写，代码量较大，且团队合作撰写。因此需要统一代码编写规范，提高代码可读性、可维护性以及后续开发的便利性。

为了达到以上目标，首先需要从系统思想上进行规范，本系统的代码总体思想如下：

1. 前后端分离，前端主要负责用户交互，利用ajax请求数据；后端主要负责与数据库交互返回请求数据，逻辑计算。前后端分离是要前后端各司其职，方便系统维护和安全性。试想一下，将各种算法写进js内，js又放入html里面，那么系统的核心代码不是要公之于众，团队的工作成果会被别人轻而易举的窃取。同时，还会因为前端过重，影响加载时间和响应速度。
2. Js，css，html分离。有一句话很好的形容了js，css，html做的工作，“html是一个一动不动的裸体人，css让人穿上了衣服，js则让人动了起来”。这句话说明了它们之间功能的差异还是很大的。同样为了可以更好的维护和SEO，js，css，html应实现分离。
3. 多动态引用，少静态加载。Django提供了方便的动态加载的api，可以较快的动态加载。系统中写死的内容尽量通过调用数据库的形式动态加载，这样后续即便维护人员不懂得Python或者js，只需要会数据库的增删改查就可以进行系统的维护，维护成本大大降低。
4. 数据应属于创建他的人。数据是财产，数据是生产力，数据创造价值，同样数据应该遵循安全规范，数据归属应有界限。不经允许系统上的任何人不可以查看到别人的数据。在编码过程中，时刻注意数据在用户维度上的分割。
5. Web系统是可扩展的平台。在编码过程中，谨防做完了事，一了百了的思想。多留后路，多为后续的开发留便捷条件。Web系统不同于其他系统打包安装就完事，web系统是可持续开发的平台，开发无止境。

## 总体代码规范

### 2.1 命名规范

一段代码通过好的命名做到即便不了解语言语法就可以读懂代码逻辑，由此可见好的命名对代码可读性有多么重要。考虑到团队特点，系统特点等，本系统命名规范总体采用驼峰法（camel-case），即当变量名或函式名是由一个或多个单字连结在一起，而构成的唯一识别字时，第一个单词以小写字母开始；第二个单词的首字母大写或每一个单词的首字母都采用大写字母。

#### 2.1.1 变量命名

（1）局部变量：采用普通的驼峰法

（2）全局变量：全部大写

#### 2.1.2 类命名

类命名采用首字母大写字母开头大写字母分割方式。

#### 2.1.3 函数命名

函数命名可采用驼峰法命名。

### 2.2 注释规范

（1）文件头注释：在每个文件头中加入下面的注释块，并且在后续的维护中及时更新，添加修改历史：

（2）类注释：在每个类定义前加入注释块，并且在后续的维护中及时更新。

（3）函数注释：在每个函数定义前加入注释块，并且在后续的维护中及时更新。

（4）代码注释：对逻辑相对复杂，算法相对复杂，修改的代码等进行注释。

## 三、Python代码规范

本系统Python代码规范主要遵循PEP8代码规范，PEP8是是Python官方的代码规范，适用于主要的Python发布版本。

### 3.1 代码布局

（1）缩进

每次缩进使用 4 个空格。续行应该与被圆括号、方括号、花括号包裹起来的其他元素对齐，或者使用悬挂式缩进。当使用悬挂式缩进时，应该遵循这些注意事项：第一行不能有参数，应该使用进一步的缩进来将续行与其他行区分开。

符合本约定的代码：

# Aligned with opening delimiter

foo = long\_function\_name(var\_one, var\_two,

var\_three, var\_four)

# More indentation included to distinguish this from the rest.

def long\_function\_name(

var\_one, var\_two, var\_three,

var\_four):

print(var\_one)

不符合本约定的代码：

# Arguments on first line forbidden when not using vertical alignment

foo = long\_function\_name(var\_one, var\_two,

var\_three, var\_four)

# Further indentation required as indentation is not distinguishable

def long\_function\_name(

var\_one, var\_two, var\_three,

var\_four):

print(var\_one)

可选的符合约定的代码：

# Extra indentation is not necessary.

foo = long\_function\_name(

var\_one, var\_two,

var\_three, var\_four)

结尾的括号另起一行，如下所示：

my\_list = [

1, 2, 3,

4, 5, 6,

]

result = some\_function\_that\_takes\_arguments(

'a', 'b', 'c',

'd', 'e', 'f',

)

or

my\_list = [

1, 2, 3,

4, 5, 6,

]

result = some\_function\_that\_takes\_arguments(

'a', 'b', 'c',

'd', 'e', 'f',

)

## 2.2 制表符

不使用Tap，更不能混合使用Tap和空格

Python 3 不支持空格缩进与制表符缩进混用。   
Python 2 中的混用缩进代码也应该被转换为统一使用空格。   
当使用 -t 选项来调用 Python 2 命令行工具时，运行混用缩进的代码会报出警告，当使用 -tt 选项时，运行混用缩进的代码会报出错误。强力建议使用这两个选项。

## 2.3 单行最大长度

每行最大长度79，换行可以使用反斜杠，最好使用圆括号。   
文档字符串、注释等最大行宽72   
换行点要在操作符的后边敲回车。   
确保使用适当的行续缩进。在二元操作符两端，换行的推荐位置是在操作符之后，而不是操作符之前。

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("I don't think so -- values are %s, %s" %

(width, height))

Blob.\_\_init\_\_(self, width, height,

color, emphasis, highlight)

## 2.4 空白行

使用两个空白行来分隔顶级函数定义、类定义。   
使用单个空白行来分隔类内的方法定义。   
额外的空白行可以被（尽量少的）用来分隔几组相关的函数。在一堆相关的单行代码之间，空白行应该被省略。   
在函数中（尽量少的）使用空白行来区分逻辑代码块。

## 2.5 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。**

#符合约定的代码：

import os

import sys

from subprocess import Popen, PIPE

#不符合本约定的代码：

import os,sys

**（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。**   
**（4）多条 Import 语句总应该遵循这样的顺序书写：**

* 标准库的导入
* 相关第三方库导入
* 本地应用/库的相关导入

**（5）当在某个包含类的模块中导入类时，这样的书写方式是合理的：**

from myclass import MyClass

from foo.bar.yourclass import YourClass

但如果这样的书写方式引起类名冲突，则请这样书写：

import myclass

import foo.bar.yourclass

并使用 “myclass.MyClass” 和 “foo.bar.yourclass.YourClass” 来对其进行引用。

# 3. 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。

#符合约定的代码

spam(ham[1], {eggs: 2})

#不符合约定的代码

spam( ham[ 1 ], { eggs: 2 } )

（2）逗号、冒号、分号前不要加空格。

#符合约定的代码

if x == 4: print x, y; x, y = y, x

#不符合约定的代码

if x == 4 : print x , y ; x , y = y , x

（3）函数的左括号前不要加空格。如Func(1)。

#符合约定代码

spam(1)

#不符合约定的代码

spam (1)

（4）序列的左括号前不要加空格。如list[2]。

#符合约定的代码

dict['key'] = list[index]

#不符合约定的代码

dict ['key'] = list [index]

（5） 操作符左右各加一个空格，不要为了对齐增加空格。

#符合约定的代码

x = 1

y = 2

long\_variable = 3

#不符合约定的代码

x = 1

y = 2

long\_variable = 3

（6） 函数默认参数使用的赋值符左右省略空格。

#符合约定的代码

def complex(real, imag=0.0):

return magic(r=real, i=imag)

#不符合约定的代码

def complex(real, imag = 0.0):

return magic(r = real, i = imag)

（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。

# 4. 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。**

# 5. 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](http://www.python.org/dev/peps/pep-0257/)

"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""

# 6. 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。

**（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

# 7. 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：

if foo.startswith('bar'):

优于

if foo[:3] == 'bar':

7 使用isinstance()比较对象的类型。比如

if isinstance(obj, int):

优于

if type(obj) is type(1):

8 判断序列空或不空，有如下规则

if not seq:

if seq:

优于

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。

## 四、html代码规范

## 五、js代码规范

## 六、css代码规范