不管脚本还是程序代码，一律到生产环境的代码按以下规范执行。

## 1.开头

开头必须有如下内容

# -\*- coding: utf-8 -\*-  
# @Author: zy  
# @Date: 2020-10-24 14:15  
# @Last Modified by: zy  
# @Last Modified time: 2020-10-25 14:39

## 2.导入格式

每个导入独占一行，禁止使用from xxx import \*，用什么模块导入什么模块，不得使用\*。

import foo  
from foo import bar

## 3.命名规则

module\_name, package\_name, ClassName, method\_name, ExceptionName, function\_name, GLOBAL\_VAR\_NAME, instance\_var\_name, function\_parameter\_name, local\_var\_name.

### 3.1应该避免的名称

1. 单字符名称, 除了计数器和迭代器.
2. 包/模块名中的连字符(-)
3. 双下划线开头并结尾的名称(Python保留, 例如\_\_init\_\_)

### 3.2命名约定

1. 所谓"内部(Internal)"表示仅模块内可用, 或者, 在类内是保护或私有的.
2. 用双下划线(\_\_)开头的实例变量或方法表示类内私有.
3. 对类名使用大写字母开头的单词(如CapWords, 即Pascal风格), 但是模块名应该用小写加下划线的方式(如lower\_with\_under.py). 尽管已经有很多现存的模块使用类似于CapWords.py这样的命名, 但现在已经不鼓励这样做, 因为如果模块名碰巧和类名一致, 这会让人困扰.

## 4．入口文件

除脚本外的程序必须有一个入口文件main.py及入口函数main（）。

def main():  
 pass  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

## 5．函数

### 5.1每个函数都应该有功能注释，来描述函数做什么，而不是描述函数怎么做。除非一些很复杂的代码，在代码上一行添加注释。

### 5.2还应该有输入和输出的详细描述。

### 5.3 return或者yield，应该描述返回值的类型及语义。

示例：

def add\_num(num, value):  
 *"""  
 数字增加  
 num : int 被加数  
 value : int 加数  
 return : int 和  
  
 # Author : zy  
 # Date : 2020-10-25 11:25  
 # Last Modified by : zy  
 # Last Modified time : 2020-10-25 12:25  
 """* total = num + value  
 return total

## 6.类

### 6.1.类名应该为大驼峰命名法ClassName

### 6.2.所有没有继承的类，都应该填写继承基类object

### 6.3. 类应该在其定义下有一个用于描述该类的文档字符串.

示例：

class Num(object):  
 *"""  
 数字类，会有一些对数字操作的方法  
 """* pass

## 7.注释

除了类和函数说明使用三引号注释，其他地方使用#注释。无论是代码块还是单行注释

## 8.其他

### 8.1 任何打开链接的使用方法，最后均要关闭。

### 8.2 不要在循环中对字符串进行累加，因为字符串是不可变类型，这样做会创建不必要的临时对象。替代方案为，加所有字串加入一个列表，最后进行join。

### 8.3 使用todo时，请写上名字及解释要做什么。如果是“将来要做什么”，请指定一个日期。

## 8.4 每次写完代码时，使用一下pycharm自动格式化代码功能。通常快捷键为ctrl+atl+l

## 8.5 不管脚本还是程序比需要要有readme