# 项目编码规范说明书

团队名称：WGYG

指导教师：代祖华

完成时间：2022 年 6 月 20 日

团队成员：魏瑾川 郭清华 姚恪 古丽妮尕尔

# 编码规范

## 缩进

统一使用 4 个空格进行缩进

## 行宽

每行代码尽量不超过 80 个字符(在特殊情况下可以略微超过 80，但最长不得超过 120)

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
- 太长可能是设计有缺陷

## 引号

简单说，自然语言使用双引号，机器标示使用单引号，因此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；很多情况还是 unicode，使用 `u"你好世界"`

- 机器标识 使用单引号 `'...'` 例如 dict 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 `"""....."""`

## 空行

- 模块级函数和类定义之间空两行；
- 类成员函数之间空一行；

```
1.  class A:
2.
3.      def __init__(self):
```

```
        pass

    def hello(self):
        pass


    def main():
        pass
```

- 可以使用多个空行分隔多组相关的函数
- 函数中可以使用空行分隔出逻辑相关的代码

## 编码

- 文件使用 UTF-8 编码
- 文件头部加入#--*conding:utf-8*--标识 (Python 3 以上版本不需要)

## import 语句

- import 语句应该分行书写

```
#正确的写法
import os
import sys

#不推荐的写法
import sys,os

#正确的写法
from subprocess import Popen, 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，于全局变量之前；
- import 语句应该按照顺序排列，每组之间用一个空行分隔

```
import os
import sys

import msgpack
import zmq

import foo
```

- 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，则可使用命名空间

```
import bar
import foo.bar

bar.Bar()
foo.bar.Bar()
```

## 空格

- 在二元运算符两边各空一格 `[=,-,+=,==,>,in,is not, and]`:

```
# 正确的写法
i = i + 1
submitted += 1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)

# 不推荐的写法
i=i+1
submitted +=1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

- 函数的参数列表中，`,`之后要有空格

- 函数的参数列表中，默认值等号两边不要添加空格

```
# 正确的写法
def complex(real, imag):
    pass

# 不推荐的写法
def complex(real,imag):
    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 complex(real, imag=0.0):
    pass

# 不推荐的写法
def complex(real, imag = 0.0):
    pass
```

- 字典对象的左括号之前不要多余的空格

```
# 正确的写法
spam(ham[1], {eggs: 2})

# 不推荐的写法
spam( ham[1], { eggs : 2 } )
```

- 不要为对齐赋值语句而使用的额外空格

```
# 正确的写法
dict['key'] = list[index]

# 不推荐的写法
dict ['key'] = list [index]
```

# 换行

Python 支持括号内的换行。这时有两种情况。

1. 第二行缩进到括号的起始处

```
foo = long_function_name(var_one, var_two, var_three, var_four)
```

2. 第二行缩进 4 个空格，适用于起始括号就换行的情形

```
def long_function_name(
        var_one, var_two, var_three,
```

```
3.          var_four):
4.      print(var_one)
```

- 使用反斜杠\换行，二元运算符 `+` `.` 等应出现在行末；长字符串也可以用此法换行

```
1.  session.query(MyTable).\
2.          filter_by(id=1).\
3.          one()
4.
5.  print 'Hello, '\
6.      '%s %s!' %\
7.      ('Harry', 'Potter')
```

- 禁止复合语句，即一行中包含多个语句：

```
1.  # 正确的写法
2.  do_first()
3.  do_second()
4.  do_third()
5.
6.  # 不推荐的写法
7.  do_first();do_second();do_third();
```

- `if/for/while` 一定要换行：

```
1.  # 正确的写法
2.  if foo == 'blah':
3.      do_blah_thing()
4.
5.  # 不推荐的写法
6.  if foo == 'blah': do_blash_thing()
```

# 注释

## 块注释

- “#”号后空一格，段落件用空行分开（同样需要“#”号）

```
1.  # 块注释
2.  # 块注释
3.  #
4.  # 块注释
5.  # 块注释
```

## 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
```

## docstring

- docstring 的规范在 PEP 257 中有详细描述，其中最其本的两点：

1. 所有的公共模块、函数、类、方法，都应该写 docstring。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`"""`应该独占一行，除非此 docstring 只有一行。

```
"""Return a foobar
Optional plotz says to frobnicate the bizbaz first.
"""

"""Oneline docstring"""
```

## 命名规范

- 应避免使用小写字母 `l(L)`，大写字母 `O(o)`或 `I(i)`单独作为一个变量的名称，以区分数字 1 和 0;
- 包和模块使用全小写命名，尽量不要使用下划线
- 类名使用 `CamelCase` 命名风格，内部类可用一个下划线开头
- 函数使用下划线分隔的小写命名
- 当参数名称和 Python 保留字冲突，可在最后添加一个下划线，而不是使用缩写或自造的词
- 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_)
```