Flask开发团队Pocoo的内部编码风格指南

Gtlions Lai

Published
with GitBook

Table of Contents

Flask开发团队Pocoo的内部编码风格指南

> 如何编写规范的Python代码？Python开发文档又该怎样写？Python官方提供的编码风格指南——PEP 8中，明确回答了上述两个问题。PEP 8内容虽然广泛，但是大部分开发团队并没有止步于PEP 8中的要求。像Flask框架的开发团队Pocoo，就是在此基础上，编写了自己内部的编码规范，对PEP 8中的规定作了细微的调整，并进行了扩展延伸。我们一起来看看Pocoo团队会给出怎样的答案呢。

团队项目

Pocoo团队开发了许多广受欢迎的项目，其中包括：

- Flask微网络开发框架
- Jinja2模板引擎
- Pygments语法高亮包
- Sphinx文档处理器
- Werzeug WSGI工具集

Pocoo团队编码风格指南适用于所有Pocoo团队的项目。总体来说，Pocoo团队编码风格指南严格遵循了PEP8的要求，但略有一些不同之处，并进行了一定的扩展延伸。

Pocoo Team简介

Pocoo团队的成员来自Python社区，统一以Pocoo的名义参与多个Python库和应用的开发工作。团队由Georg Brandl和Armin Ronacher领导。

代码布局（General Layout）

缩进（**Indentation**）

4个空格。不能使用Tab制表符，没有例外。

每行最大长度

79个字符，如果必要可以接受最多84个字符。尽量避免代码嵌套层级过多，可以通过巧妙地使用`break`、`continue`和`return`语句实现。

长语句换行

编写长语句时，可以使用换行符（\）换行。在这种情况下，下一行应该与上一行的最后一个“.”句点或“=”对齐，或者是缩进4个空格符：

```
this_is_a_very_long(function_call, 'with many parameters') \
    .that_returns_an_object_with_an_attribute

MyModel.query.filter(MyModel.scalar > 120) \
             .order_by(MyModel.name.desc()) \
             .limit(10)
```

如果你使用括号“()”或花括号“{}”为长语句换行，那么下一行应与括号或花括号对齐：

```
this_is_a_very_long(function_call, 'with many parameters',
                    23, 42, 'and even more')
```

对于元素众多的列表或元组，在第一个“[”或“(”之后马上换行：

```
items = [
    'this is the first', 'set of items', 'with more items',
    'to come in this line', 'like this'
]
```

空行

顶层函数与类之间空两行，此外都只空一行。不要在代码中使用太多的空行来区分不同的逻辑模块。

示例：

```
def hello(name):
    print 'Hello %s!' % name

def goodbye(name):
    print 'See you %s.' % name

class MyClass(object):
    """This is a simple docstring."""

    def __init__(self, name):
        self.name = name

    def get_annoying_name(self):
        return self.name.upper() + '!!!!111'
```

表达式和语句

一般空格规则

- 单目运算符与运算对象之间不空格（例如，-，~等），即使单目运算符位于括号内部也一样。
- 双目运算符与运算对象之间要空格。

好风格：

```
exp = -1.05
value = (item_value / item_count) * offset / exp
value = my_list[index]
value = my_dict['key']
```

坏风格：

```
exp = - 1.05
value = ( item_value / item_count ) * offset / exp
value = (item_value/item_count)*offset/exp
value=( item_value/item_count ) * offset/exp
value = my_list[ index ]
value = my_dict ['key']
```

不能编写尤达语句（**Yoda Statements**，指的是类似星战中尤达大师那样颠倒句子的正常顺序）：

不要拿常量与变量进行对比，应该拿变量与常量进行对比。

好风格：

```
if method == 'md5':
    pass
```

坏风格：

```
if 'md5' == method:
    pass
```

比较：

- 任意类型之间的比较，使用“**==**”和“**!=**”。
- 与单例（singletons）进行比较时，使用 `is` 和 `is not` 。
- 永远不要与 `True` 或 `False` 进行比较（例如，不要这样写： `foo == False` ，而应该这样写： `not foo` ）。

否定成员关系检查：

使用 `foo not in bar` ，而不是 `not foo in bar` 。

命名规范

- 类名称：采用骆驼拼写法（CamelCase），首字母缩略词保持大写不变（`HTTPWriter`，而不是`HttpWriter`）。
- 变量名：小写以及下划线（lowercase_with_underscores）。
- 方法与函数名：小写以及下划线（lowercase_with_underscores）。
- 常量：大写以及下划线（UPPERCASE_WITH_UNDERSCORES）。
- 预编译的正则表达式：name_re。

受保护的元素以一个下划线为前缀。双下划线前缀只有定义混入类（mixin classes）时才使用。

如果使用关键词（keywords）作为类名称，应在名称后添加后置下划线（trailing underscore）。允许与内建变量重名，不要在变量名后添加下划线进行区分。如果函数需要访问重名的内建变量，请将内建变量重新绑定为其他名称。

函数和方法的参数：

- 类方法：`cls`为第一个参数。
- 实例方法：`self`为第一个参数。
- property函数中使用匿名函数（lambdas）时，匿名函数的第一个参数可以用`x`替代，例如：`display_name = property(lambda x: x.real_name or x.username)`。

实例检查：

使用 `isinstance(a, C)`，而不是 type(a) is C`。但是一般要避免做实例检查。建议检查实例的特性。

文档字符串

文档字符串规范：

所有文档字符串均以reStructuredText格式编写，方便Sphinx处理。文档字符串的行数不同，布局也不一样。如果只有一行，代表字符串结束的三个引号与代表字符串开始的三个引号在同一行。如果为多行，文档字符串中的文本紧接着代表字符串开始的三个引号编写，代表字符串结束的三个引号则自己独立成一行。

```
def foo():
    """This is a simple docstring."""

def bar():
    """This is a longer docstring with so much information in there
 that it spans three lines.  In this case, the closing triple quote
 is on its own line.
 """
```

一般来说，文档字符串应分成简短摘要（尽量一行）和详细介绍。如果必要的话，摘要与详细介绍之间空一行。

模块头部：

模块文件的头部包含有utf-8编码声明（如果模块中使用了非ASCII编码的字符，建议进行声明），以及标准的文档字符串。

```
# -*- coding: utf-8 -*-
"""
 package.module
 ~~~~~~~~~~~~~~

 A brief description goes here.

 :copyright: (c) YEAR by AUTHOR.
 :license: LICENSE_NAME, see LICENSE_FILE for more details.
"""
```

请注意，要让你开发的Flask扩展（extension）通过官方团队审核，则必须提供相应的版权与协议文件。

注释

注释的规范与文档字符串编写规范类似。二者均以reStructuredText格式编写。如果使用注释来编写类属性的文档，请在#符号后添加一个冒号，：。

```
class User(object):
    #: the name of the user as unicode string
    name = Column(String)
    #: the sha1 hash of the password + inline salt
    pw_hash = Column(String)
```

- 原文链接：http://www.pocoo.org/internal/styleguide/