作者 | Alex Omeyer

译者 | 翟珂

Python
的风格优雅干净，但语法干净并不等同于编写的代码也是干净的。开发人员仍然需要学习Python最佳实践和设计模式。

# 什么是干净的代码？

C++的发明者Bjarne
Stroustrup
说过一句话清楚地解释了干净代码的含义：“我喜欢我的代码是优雅和高效的。逻辑应该是直截了当的，这样就很难隐藏错误；依赖关系应该是最小的，这样便于维护；错误处理应该是完整的，符合明确的策略；性能应该是接近最佳的，这样就不会诱使人们用无原则的优化使代码变得混乱。干净的代码能做好这件事。”

从这句话中，我们可以挑选出干净代码的一些品质：

- 干净的代码是有重点的。每个函数、类或模块都应该做一件事，而且要做得好。
- 干净的代码容易阅读和推理。根据《面向对象的分析和设计与应用》一书的作者Grady Booch的说法：干净的代码读起来就像写好的散文。
- 干净的代码很容易调试。
- 干净的代码易于维护。也就是说，其他开发人员可以轻松阅读和优化它。
- 干净的代码具有高性能。

开发人员可以随心所欲地编写他们的代码，因为没有固定的或约束性的规则来要求他/她编写干净的代码。而糟糕的代码会产生技术债务，从而对公司造成严重后果。

在本文中，我们将看看一些帮助我们在Python中编写干净代码的设计模式。让我们在下文中了解它们。

# 编写干净Python代码的手段

命名规则：

命名规则是编写干净代码的最有用和最重要的方面之一。在给变量、函数、类等命名时，要使用有意义的、能揭示意图的名字。而这意味着我们会倾向于使用长的描述性名称，而不是短的模糊不清的名称。

下面是一些例子：

1、使用易于阅读的长描述性名称。这将省去写不必要的注释，如下所示：

```
# ???
# au????????
au = 105

# ??
total_active_users = 1051.2.3.4.5.6.
```

2、使用描述内容名称。
其他开发人员应该能够从名称中找出你的变量存储的内容。简而言之，你的代码应该易于阅读和推理。

```
# ???
c = [“UK”, “USA”, “UAE”]

for x in c:
print(x)

# ??
cities = [“UK”, “USA”, “UAE”]
    for city in cities:
        print(city)1.2.3.4.5.6.7.8.9.10.
```

3、避免使用模棱两可的简称。
变量应该有一个长的描述性名称，而不是一个容易混淆的简称。

```
# ???
fn = 'John'
Ln = ‘Doe’
cre_tmstp = 1621535852
```

```
# ??
first_name = ‘John’
Las_name = ‘Doe’
creation_timestamp = 16215358521.2.3.4.5.6.7.8.9.
```

4、始终使用相同的词汇。
与你的命名规则保持一致。当其他开发人员处理你的代码时，保持一致的命名规则对于消除混淆非常重要。这适用于命名变量、文件、方法甚至目录结构。

```
# ???
client_first_name = ‘John’
customer_last_name = ‘Doe;

# ??
client_first_name = ‘John’
client_last_name = ‘Doe’

Also, consider this example:
#???
def fetch_clients(response, variable):
    # ??
    pass

def fetch_posts(res, var):
    # ??
    pass

# ??
def fetch_clients(response, variable):
    # ??
    pass

def fetch_posts(response, variable):
    # ??
    pass1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20.2
1.22.23.24.25.26.
```

5、在你的编辑器中开始跟踪代码库的问题。

让工程师可以轻松地跟踪和查看代码本身的问题是保持Python代码库清洁的一个主要手段。允许工程师在编辑器中跟踪代码库问题可以让工程师们：

- 全面了解技术债务
- 查看每个代码库问题的上下文
- 减少上下文切换
- 不断解决技术债务问题

你可以使用各种工具来跟踪你的技术债务，但最快速和最简单的方法是使用VSCode或JetBrains的免费Stepsize扩展，它可以与Jira
、Linear、Asana和其他项目管理工具集成。

6、不要使用魔法值。魔法值
是具有特殊的、硬编码语义的数字，它出现在代码中但没有任何解释。所以我们将这些数字以文字形式出现在我们代码中的多个位置。

```
import random

# ???
def roll_dice():
    return random.randint(0, 4)  # 4???????

# ??
DICE_SIDES = 4

def roll_dice():
    return random.randint(0, DICE_SIDES)1.2.3.4.5.6.7.8.9.10
.11.
```

函数：

7、保持一致的函数命名规则。

正如上面的变量所见，在命名函数时要坚持一个命名习惯。使用不同的命名习惯会使其他开发者感到困惑。

```
# ???
def get_users():
    # ??
```

```
    Pass

def fetch_user(id):
    # ??
    Pass

def get_posts():
    # ??
    Pass

def fetch_post(id):
    # ??
    pass

# ??
def fetch_users():
    # ??
    Pass

def fetch_user(id):
    # ??
    Pass

def fetch_posts():
    # ??
    Pass

def fetch_post(id):
    # ??
    pass1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20.2
1.22.23.24.25.26.27.28.29.30.31.32.33.
```

8、函数应该只做一件事，而且要做得好
。
写短而简单的函数，执行单一的任务。需要注意的是，如果你的函数名称包含“and”，你可能需要把它拆分成两个函数。

```
# ???
def fetch_and_display_users():
```

```
users = [] # ?? api ?????

    for user in users:
        print(user)


# ??
def fetch_usersl():
    users = [] # ?? api ?????
        return users

def display_users(users):
for user in users:
        print(user)1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.
```

9、不要使用布尔值。
布尔值（真或假）。每种结果应该作为一个单独的函数去调用，而不是当作函数的入参。

类：

10. 不要添加多余的描述。在使用类时，变量名不要添加不必要的前缀。

```
# ???
class Person:
    def __init__(self, person_username, person_email, person
_phone, person_address):
        self.person_username = person_username
        self.person_email = person_email
        self.person_phone = person_phone
        self.person_address = person_address

# ??
class Person:
    def __init__(self, username, email, phone, address):

        self.username = username
        self.email = email
        self.phone = phone
```

```
        self.address = address1.2.3.4.5.6.7.8.9.10.11.12.13.
14.15.16.
```

在上面的例子中，由于我们已经在Person类里面了，所以没有必要在每个类的变量上添加person_前缀。

奖励：模块化你的代码

为了保持你的代码的条理性和可维护性，把你的逻辑分成不同的文件或类，我们称之为模块。Python中的模块是一个以.py为扩展名的文件。每个模块都应该专注于做一件事，并且把它做好。

你可以遵循面向对象的OOP原则，例如遵循基本的OOP原则，如封装、抽象、继承和多态。

# 结论

编写干净的代码有很多优点，如：提高软件质量
、增强代码可维护性和消除技术债务等。而这些提高编写干净代码的手段同时也适用于其他语言，我希望通过阅读这篇文章，你已经对干净代码和编写干净代码的一些手段有了足够的了解。

原文链接：https://dzone.com/articles/10-must-know-patterns-for-writing-clean-code-with-1

# 译者介绍

翟珂，51CTO社区编辑，目前在杭州从事软件研发工作，做过电商、征信等方面的系统，享受分享知识的过程，充实自己的生活。

来源： 51CTO技术栈