让 Python 代码更易维护的七种武器

LCTT

（点击上方公众号，可快速关注）

编译：linux中国-Hank Chow，英文：Jeff Triplett

https://linux.cn/article-10059-1.html

检查你的代码的质量，通过这些外部库使其更易维护。

可读性很重要。

— Python 之禅The Zen of Python，Tim Peters

随着软件项目进入“维护模式”，对可读性和编码标准的要求很容易落空（甚至从一开始就没有建立过那些标准）。然而，在代码库中保持一致的代码风格和测试标准能够显著减轻维护的压力，也能确保新的开发者能够快速了解项目的情况，同时能更好地全程保持应用程序的质量。

使用外部库来检查代码的质量不失为保护项目未来可维护性的一个好方法。以下会推荐一些我们最喜爱的检查代码（包括检查 PEP 8 和其它代码风格错误）的库，用它们来强制保持代码风格一致，并确保在项目成熟时有一个可接受的测试覆盖率。

检查你的代码风格

PEP 8 是 Python 代码风格规范，它规定了类似行长度、缩进、多行表达式、变量命名约定等内容。尽管你的团队自身可能也会有稍微不同于 PEP 8 的代码风格规范，但任何代码风格规范的目标都是在代码库中强制实施一致的标准，使代码的可读性更强、更易于维护。下面三个库就可以用来帮助你美化代码。

1、 Pylint

Pylint 是一个检查违反 PEP 8 规范和常见错误的库。它在一些流行的编辑器和 IDE 中都有集成，也可以单独从命令行运行。

执行 pip install pylint 安装 Pylint 。然后运行 pylint [options] path/to/dir 或者 pylint [options] path/to/module.py 就可以在命令行中使用 Pylint，它会向控制台输出代码中违反规范和出现错误的地方。

你还可以使用 pylintrc 配置文件来自定义 Pylint 对哪些代码错误进行检查。

2、 Flake8

Flake8 是“将 PEP 8、Pyflakes（类似 Pylint）、McCabe（代码复杂性检查器）和第三方插件整合到一起，以检查 Python 代码风格和质量的一个 Python 工具”。

执行 pip install flake8 安装 flake8 ，然后执行 flake8 [options] path/to/dir 或者 flake8 [options] path/to/module.py 可以查看报出的错误和警告。

和 Pylint 类似，Flake8 允许通过配置文件来自定义检查的内容。它有非常清晰的文档，包括一些有用的提交钩子，可以将自动检查代码纳入到开发工作流程之中。

Flake8 也可以集成到一些流行的编辑器和 IDE 当中，但在文档中并没有详细说明。要将 Flake8 集成到喜欢的编辑器或 IDE 中，可以搜索插件（例如 Sublime Text 的 Flake8 插件）。

3、 Isort

Isort 这个库能将你在项目中导入的库按字母顺序排序，并将其正确划分为不同部分（例如标准库、第三方库、自建的库等）。这样提高了代码的可读性，并且可以在导入的库较多的时候轻松找到各个库。

执行 pip install isort 安装 isort，然后执行 isort path/to/module.py 就可以运行了。文档中还提供了更多的配置项，例如通过配置 .isort.cfg 文件来决定 isort 如何处理一个库的多行导入。

和 Flake8、Pylint 一样，isort 也提供了将其与流行的编辑器和 IDE 集成的插件。

分享你的代码风格

每次文件发生变动之后都用命令行手动检查代码是一件痛苦的事，你可能也不太喜欢通过运行 IDE 中某个插件来实现这个功能。同样地，你的同事可能会用不同的代码检查方式，也许他们的编辑器中也没有那种插件，甚至你自己可能也不会严格检查代码和按照警告来更正代码。总之，你分享出来的代码库将会逐渐地变得混乱且难以阅读。

一个很好的解决方案是使用一个库，自动将代码按照 PEP 8 规范进行格式化。我们推荐的三个库都有不同的自定义级别来控制如何格式化代码。其中有一些设置较为特殊，例如 Pylint 和 Flake8 ，你需要先行测试，看看是否有你无法忍受但又不能修改的默认配置。

4、 Autopep8

Autopep8 可以自动格式化指定的模块中的代码，包括重新缩进行、修复缩进、删除多余的空格，并重构常见的比较错误（例如布尔值和 None 值）。你可以查看文档中完整的更正列表。

运行 pip install --upgrade autopep8 安装 Autopep8。然后执行 autopep8 --in-place --aggressive --aggressive 就可以重新格式化你的代码。aggressive 选项的数量表示 Auotopep8 在代码风格控制上有多少控制权。在这里可以详细了解 aggressive 选项。

5、 Yapf

Yapf 是另一种有自己的配置项列表的重新格式化代码的工具。它与 Autopep8 的不同之处在于它不仅会指出代码中违反 PEP 8 规范的地方，还会对没有违反 PEP 8 但代码风格不一致的地方重新格式化，旨在令代码的可读性更强。

执行 pip install yapf 安装 Yapf，然后执行 yapf [options] path/to/dir 或 yapf [options] path/to/module.py 可以对代码重新格式化。定制选项的完整列表在这里。

6、 Black

Black 在代码检查工具当中算是比较新的一个。它与 Autopep8 和 Yapf 类似，但限制较多，没有太多的自定义选项。这样的好处是你不需要去决定使用怎么样的代码风格，让 Black 来给你做决定就好。你可以在这里查阅 Black 有限的自定义选项以及如何在配置文件中对其进行设置。

Black 依赖于 Python 3.6+，但它可以格式化用 Python 2 编写的代码。执行 pip install black 安装 Black，然后执行 black path/to/dir 或 black path/to/module.py 就可以使用 Black 优化你的代码。

检查你的测试覆盖率

如果你正在进行编写测试，你需要确保提交到代码库的新代码都已经测试通过，并且不会降低测试覆盖率。虽然测试覆盖率不是衡量测试有效性和充分性的唯一指标，但它是确保项目遵循基本测试标准的一种方法。对于计算测试覆盖率，我们推荐使用 Coverage 这个库。

7、 Coverage

Coverage 有数种显示测试覆盖率的方式，包括将结果输出到控制台或 HTML 页面，并指出哪些具体哪些地方没有被覆盖到。你可以通过配置文件自定义 Coverage 检查的内容，让你更方便使用。

执行 pip install coverage 安装 Converage 。然后执行 coverage [path/to/module.py] [args] 可以运行程序并查看输出结果。如果要查看哪些代码行没有被覆盖，执行 coverage report -m 即可。

持续集成工具

持续集成Continuous integration（CI）是在合并和部署代码之前自动检查代码风格错误和测试覆盖率最小值的过程。很多免费或付费的工具都可以用于执行这项工作，具体的过程不在本文中赘述，但 CI 过程是令代码更易读和更易维护的重要步骤，关于这一部分可以参考 Travis CI 和 Jenkins。

以上这些只是用于检查 Python 代码的各种工具中的其中几个。如果你有其它喜爱的工具，欢迎分享。

【关于投稿】

如果大家有原创好文投稿，请直接给公号发送留言。

① 留言格式：

【投稿】+《 文章标题》+ 文章链接

② 示例：

【投稿】《不要自称是程序员，我十多年的 IT 职场总结》：http://blog.jobbole.com/94148/

③ 最后请附上您的个人简介哈~