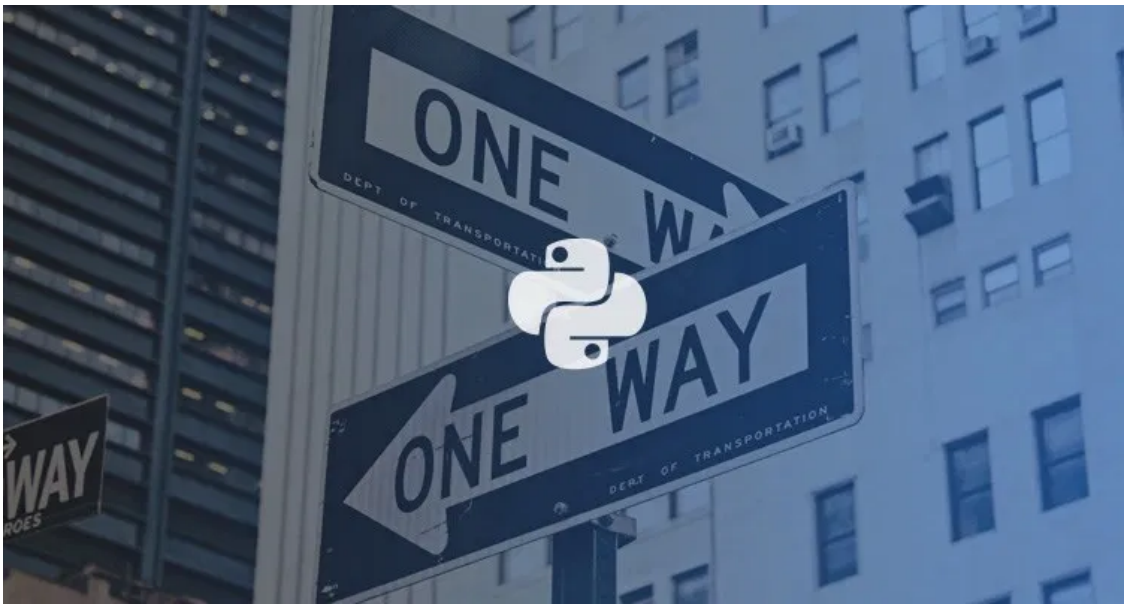
Which Python static analysis tools should I use?



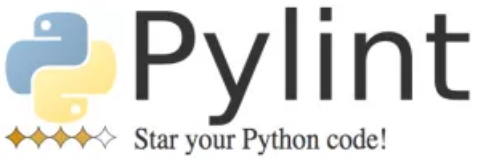
AUTHOR
Jaime

DATE
August 8, 2018

CATEGORY
Developer Education

As you might have read in our introductory article about static code analysis, there are a lot of reasons why you should implement it in your development workflow. When programming in Python, there are many Python static analysis tools to choose from. However, not all of them are worth your time. In this review, we'll be taking a look at our favorite options and explain which ones to use (if you don't program in Python we perform similar reviews for Java, Ruby, Scala and PHP).

1: Pylint



One tool stands out from the rest — Pylint doesn't just have the best range of features, but it's also constantly maintained, making it a must-have tool for Python developers.

It's been around for 13 years, and over that time it has included features like coding standards, error detection and refactoring by detecting duplicated code.

Out of the box Pylint is easy to set up, requiring a minimal amount of configuration, but it's fully customizable if you want it to be — by editing a file you can select which errors and conventions are most relevant to you.

Running Pylint on some code will result in something like this:

```
$ pylint cards.py
************* Module python_cards.cards
W:  4, 0: Found indentation with tabs instead of spaces (mixed-indentation)
C:  1, 0: Missing module docstring (missing-docstring)
C:  3, 0: Missing class docstring (missing-docstring)
C:  3, 0: Old-style class defined. (old-style-class)
R:  3, 0: Too few public methods (0/2) (too-few-public-methods)
W:  9,20: Redefining built-in 'list' (redefined-builtin)
```

Most of the messages here will be self-explanatory, but the first letter in each line will map to Convention, Refactor, Warning, or Error — and with regards to coding style, Pylint follows the PEP8 style guide. Pylint ships with Pyreverse, which is used to create UML diagrams for your code.

It's possible to automate Pylint with Apycot, Hudson or Jenkins, and it also integrates with several editors. If your feature isn't available, you can also write small plugins to add personal features.

With support for Pylint being very broad, we're hard pressed to find a reason not to use this stellar tool for Python static analysis.

2: pyflakes

Another tool that can help with Python static analysis is Pyflakes. Its approach is centered around trying not to emit false positives, which brings its own set of advantages and disadvantages.

For example, it only examines the syntax tree of each file individually. Combined with a limited set of reportable errors, this makes it faster than Pylint, but it also means it's more limited in terms of the things it can check.

This makes sense, however, when looking at the context of the tool's creation. Back when it was first being developed, the author was working in an environment where all developers had "years of experience without a single intern in sight." He's also on the

record for saying that the code quality was “very high”, meaning his tool only needed to catch minor issues.

While Pyflakes doesn’t do any stylistic checks, there’s another tool that combines pyflakes with style checks against PEP8 called Flake8. On top of this, it also adds powerful per-project configuration abilities.

3: Mypy

Slightly different than the earlier two options, Mypy is a static type checker for Python. Its main requirement is that your code is annotated using the Python 3 function annotation syntax (PEP 484 notation) in order for it to type check your code and find common bugs. It’s worth checking out the examples here.

```
def fib(n: int) -> Iterator[int]:
    a, b = 0, 1
    while a < n:
        yield a
        a, b = b, a+b
```

Its goal is to combine the benefits of dynamic typing and static typing, and while Mypy is still in development, it already supports a significant subset of Python features. Mypy’s type checking is done in compile-time — type declarations act as machine-checked documentation and static typing makes your code easier to understand and easier to modify without introducing bugs.

It’s important to point out Mypy currently only supports Python 3 syntax — while they’re working on Python 2 support, it’s still in an early stage of development.

You can find the documentation here. There’s also a list of planned features.

It should be noted that mypy currently supports Python 3 syntax and Python 2 support is still in early development. In general, it’s good to know that the software is considered to be experimental, meaning future changes may break backward compatibility.

Which one should I use?

There’s no conclusive answer here, as all tools have stronger and weaker points.

Currently Pylint is our go-to tool, but both Pyflakes and Mypy have interesting features that could prove to be useful for some users.

Some other tools are also worth mentioning, like PySonar2 (a type inferences and indexer), AutoPep8 (which automatically fixes PEP8). Also, don’t forget to check out the

Code Quality mailing list, which currently covers PEP8, Pyflakes, mccabe, Flake8 and pylint.

If you're looking for a simple and time-saving way to integrate Pylint you should definitely check out Codacy — it's the automated code review platform that lets you easily point to your git repository so you can get additional reports and deltas per commit. With our free trial waiting for you, there's no reason not to give it a spin.

Tools For Other Coding Languages

Check out this page for reviews on static analysis tools in different coding languages.

PREVIOUS ARTICLE
How to code review in a Pull Request

NEXT ARTICLE
Put BDD (Behavior Driven Development) in practice with Scala

You may also like

Free Codacy Pro account to fight COVID-19

How Agile & Container technology led to the rise of enterprise DevSecOps

Top 6 items for your code review checklist

Recent posts

Enhanced security for C++, Java, and Scala with Clang-Tidy and SpotBugs

April 27, 2020

Improve the efficiency of your remote engineering team

April 13, 2020

Further Enterprise security analysis for Scala

April 8, 2020

Free Codacy Pro account to fight COVID-19

March 25, 2020

Introducing GitHub Apps for improved user access control

March 20, 2020