Pydocstyle checks docstrings written within Python code, for compliance with the PEP 257 standard- an official Python document which dictates some rules for writing docstrings (Rachum & Kothari, 2022). Pylint is a tool that finds "code smells" through static analysis- "code smells" are something I would define as certain writing practices which lower code's quality and maintainability. Pyflakes fulfils a similar purpose, but is marketed as performing its analysis quite efficiently (PyCQA, 2022). Pycodestyle also has a similar role, however, its analysis is based on the standards outlined in PEP 8- an official document which dictates a coding style to follow for Python (PyCQAb, 2016).

Pydocstyle serves a unique purpose, however, there is clear overlap between pycodestyle, pyflakes, and pylint, and it may be appropriate to use only one. I would recommend the use of pycodestyle above all others, because it enforces a coding standard officially adopted by the Python language- pyflakes and pylint do not guarantee this and therefore using either of them in conjunction with pycodestyle may lead to the enforcement of an inconsistent coding style.

These programs can be used in various ways in real-world development environments. They could be made part of a CI pipeline- a popular approach is to have linting tools generate reports on code submitted to a repository as part of a pull request, which is done by applications such as SonarCloud (SonarSource, 2022). However, I think that tools of this nature shouldn't generate reports- rather, any code committed should immediately be updated to enforce the standards set by the linting program. Having worked with this type of workflow myself, I have seen that developers may not have the time to (manually) update their code to fix these issues, and they might also forget to do so. Code elements checked by linting programs do not directly change a program's behaviour and therefore carry very little risk of altering behaviour, making it worthwhile to automate this aspect of development.

**References**

PyCQA. (2016) pycodestyle’s documentation. Available from: <https://pycodestyle.pycqa.org/en/latest/> [Accessed 22 May 2022].

PyCQA. (2022) Pyflakes. Available from: <https://github.com/PyCQA/pyflakes> [Accessed 22 May 2022].

Rachum, A. & Kothari, S. (2022) pydocstyle’s documentation. Available from: <http://www.pydocstyle.org/en/stable/> [Accessed 22 May 2022].

SonarSource. (2022) Scan your code with SonarCloud. Available from: <https://github.com/SonarSource/sonarcloud-github-action> [Accessed 22 May 2022].