Python Styleguide

Dimitar Dimitrov

25. April 2017

Zweck und Sinn eines Styleguides

- Zweck
  - Konsistenz
  - Programme erzeugen, die leicht
    - zu **lesen**,
    - zu **verstehen**,
    - und zu **ändern** sind
- Zielgruppe
  - Entwickler
- Ansatz
  - Richtlinien

Styleguides und Python

- Python Enhancement Proposals (PEPs)
  - `https://www.python.org/dev/peps/`
- PEP 8 – Style Guide for Python Code
  - `https://www.python.org/dev/peps/pep-0008/`
  - Code wird viel öfter gelesen als geschrieben
  - Es gibt Gründe, die Richtlinien zu ignorieren
- PEP 20 – The Zen of Python
  - `https://www.python.org/dev/peps/pep-0020/`
  - `import this`

pep8 vs. PEP 8

- pep8 ist ein Werkzeugt, das Code nach den Richtlinien in PEP 8 überprüft.
- pep8 heißt inzwischen `pycodestyle`
- `pycodestyle --show-source --show-pep8 FILE`
- `pycodestyle --statistics -qq FOLDER`
- In Python-Projekte über `setup.cfg` konfiguierbar
  - ```
    [pycodestyle]
    ignore = E226,E302,E41
    max-line-length = 160
    ```
- Mittels `# noqa` direkt in Code
    ```

Beyond PEP 8

- Python Code Quality Authority
  - `https://github.com/PyCQA`
  - Werkzeuge zum Überprüfen der Qualität von Python Quellcode

Pyflakes, Flake8, Pylint

- Pyflakes
  - `https://github.com/PyCQA/pyflakes`
  - Prüft Python Quellcode nach Fehler
  - Prüft den Stil des Codes **nicht**.
- **Flake8**
  - `https://gitlab.com/pycqa/flake8`
  - `http://flake8.pycqa.org/en/latest/`
  - Kombiniert Pyflakes und pycodestyle
- Pylint
  - `https://github.com/PyCQA/pylint`
  - Analysiert Python Quellcode
  - `pyreverse` generiert UML-Diagramme

Empfehlungen

- Styleguide als formales Dokument
  - Titel, Author, Datum, Version
  - Inhaltsverzeichnis
  - Liste der Änderungen mit Datum
  - Seitennummer
  - Beispiele
- `pycodestyle`/`Flake8` und Git
  - `flake8 --install-hook git`
  - `git config --bool flake8.strict true`