Spickzettel: Statische Codeanalyse in Python

Ziel

Fehler, Stilprobleme, Sicherheitslücken und schlechte Praktiken im Python-Code frühzeitig erkennen – **ohne Ausführung** des Codes.

Tools zur statischen Analyse

`flake8`

- Kombination aus `pyflakes`, `pycodestyle` und Plugins
- Prüft Syntax, Stil, unnötige Importe

```
pip install flake8
flake8 src/
```

`pylint`

- Umfassender Checker (Stil, Refactoring, Bugs, Konventionen)
- Gibt Score von 0–10 aus

```
pip install pylint
pylint src/
```

`mypy`

- Statische Typprüfung für `type hints`
- Zeigt Inkompatibilitäten bei Typannotationen

```
pip install mypy
mypy src/
```

`bandit`

- Sicherheitsscanner für Python (z. B. eval, subprocess, Secrets)

```
pip install bandit
bandit -r src/
```

`vulture`

- Findet unbenutzten Code

```
pip install vulture
vulture src/
```

Integration in CI/CD

- Kombinierbar mit `pre-commit`, `tox`, GitHub Actions
- Beispiel `pre-commit-config.yaml`:

```
- repo: https://github.com/pycqa/flake8
  rev: v6.1.0
  hooks:
    - id: flake8
```

Best Practices

- Kombiniere mehrere Tools für besten Effekt
- Verwende `pre-commit` zur lokalen Absicherung
- Tool-Konfiguration in `.flake8`, `pyproject.toml` etc.
- Sicherheitsanalyse regelmäßig im CI laufen lassen

Statische Analyse spart Zeit beim Debuggen, verbessert Codequalität und schafft Vertrauen in den Code – schon bevor er ausgeführt wird.