Spickzettel: Git Hook – pre-commit mit pylint

Ziel

- Vor jedem Commit soll automatisch **`pylint`** ausgeführt werden.
- Bei Fehlern wird der Commit abgebrochen.

Schritt 1: Hook-Datei anlegen

Pfad: `.git/hooks/pre-commit`

```
#!/bin/bash

# Nur .py-Dateien prüfen, die geändert wurden
files=$(git diff --cached --name-only --diff-filter=ACM | grep '\.py$')

if [ -z "$files" ]; then
  echo "Keine Python-Dateien zum Prüfen."
  exit 0
fi

# pylint auf jede Datei einzeln anwenden
pass=true
for file in $files; do
  echo "Prüfe $file mit pylint..."
  pylint "$file"
  if [ $? -ne 0 ]; then
    pass=false
  fi
done

if ! $pass; then
  echo "\n   Commit abgebrochen wegen pylint-Fehlern."
  exit 1
fi

echo "   Alle pylint-Checks bestanden. Commit erlaubt."
exit 0
```

Schritt 2: Hook ausführbar machen

```
chmod +x .git/hooks/pre-commit
```

Plattformhinweise

Linux & macOS

- Obiges Bash-Skript funktioniert direkt.
- Terminal verwenden, `chmod +x` nicht vergessen.
- Python & `pylint` müssen im `PATH` verfügbar sein.

Windows

- Standardmäßig funktionieren Bash-Hooks nur mit Git Bash (z. B. bei Git for Windows).
- Stelle sicher, dass `pylint` über `PATH` verfügbar ist (z. B. via `pip install pylint`).
- Falls Bash nicht vorhanden ist, kann ein `.bat`-Hook verwendet werden:

```
@echo off
for /f %%f in ('git diff --cached --name-only --diff-filter=ACM ^|
findstr /R ".*\.py$"') do (
    echo Prüfe %%f mit pylint...
    pylint %%f
    if errorlevel 1 (
        echo Fehler in %%f
        exit /b 1
    )
)
echo Alle Dateien bestanden.
```

Datei als `.git/hooks/pre-commit.bat` speichern.

Test

1. Ändere oder erstelle `.py`-Dateien.
2. `git add .`
3. `git commit -m "Test"` → Sollte nur funktionieren, wenn pylint keine Fehler findet.

Tipps

- Konfiguration über `.pylintrc` einbinden
- Für Projektweite Nutzung: Tools wie `pre-commit` Framework einsetzen
- Nur bestimmte Pfade prüfen → `grep '^src/'` etc.
- Andere Tools kombinieren: `black`, `flake8`, `pytest`, etc.