Spickzettel: Testabdeckung in Python messen

Ziel

Den Umfang getesteten Codes ermitteln, um Lücken sichtbar zu machen und Qualität zu steigern.

Tool: `coverage.py`

- Standardwerkzeug für Testabdeckung in Python
- Integration mit `pytest`, `unittest` und CI/CD

Installation

```
pip install coverage
```

Nutzung mit `pytest`

```
coverage run -m pytest
coverage report
coverage html  # erzeugt HTML-Bericht
```

Nutzung mit `unittest`

```
coverage run -m unittest discover
coverage report
```

Beispielausgabe

```
Name             Stmts   Miss  Cover
------------------------------
src/utils.py        20      0   100%
src/main.py         50     10    80%
```

HTML-Report

```
coverage html
# öffne htmlcov/index.html im Browser
```

- Zeigt Zeile-für-Zeile welche Codeteile abgedeckt sind

Erweiterungen

- `pytest-cov`: Plugin für nahtlose Integration

```
pip install pytest-cov
pytest --cov=src --cov-report=term --cov-report=html
```

Best Practices

- Nicht 100% Coverage erzwingen – Fokus auf kritische Pfade
- Coverage bei jedem CI-Run prüfen
- Reports regelmäßig analysieren & Lücken schließen

Hinweis

- `coverage run` trackt nur Python-Code – keine Shell-, C-, SQL-Anteile etc.
- Nutze `.coveragerc` zur Feinkonfiguration (z. B. Ausschlüsse, Pfade)

Testabdeckung ist ein Werkzeug, kein Ziel – Qualität entsteht durch sinnvolle Tests, nicht durch Prozentzahlen.