Spickzettel: Testkriterien für gute Python-Tests

Ziel

Worauf sollte man achten, um sinnvolle, zuverlässige und wartbare Tests zu schreiben? Diese Kriterien helfen bei der Beurteilung.

Gute Tests sind ...

1. Isoliert

- Jeder Test testet **nur eine Sache**
- Keine Abhängigkeiten zu anderen Tests oder externen Daten

2. Reproduzierbar

- Tests liefern **immer** das gleiche Ergebnis
- Kein Zufall, keine Uhrzeit, keine Netzwerk-Abhängigkeit

3. Schnell

- Tests sollen CI/CD nicht ausbremsen
- Faustregel: Jeder Test < 1 Sekunde

4. Verständig

- Aussagekräftiger Testname (`test_login_with_invalid_password_fails`)
- Klarer Zweck aus Code und ggf. Kommentar ersichtlich

5. Unabhängig vom Kontext

- Läuft in beliebiger Umgebung (lokal, CI, Container)
- Keine versteckten Abhängigkeiten zu Pfaden, Variablen

6. Vollständig

- Relevante Sonderfälle und Fehlerpfade werden mitgetestet
- Positive & negative Fälle, Edge Cases

Zusätzliche Kriterien

- **Keine Logik im Test**: Test sollte prüfen, nicht neu rechnen

- **Testdaten klar benennen**: Inputs, Outputs, Mocks
- **Trennschärfe**: kein "Catch-all"-Test, sondern gezielte Einheiten
- **Setup minimieren**: Nur was für den Test notwendig ist

Review-Checkliste für Tests

- [ ] Testname beschreibt das erwartete Verhalten?
- [ ] Ist klar, was getestet wird?
- [ ] Ist das Ergebnis eindeutig überprüfbar?
- [ ] Wird ein echter Fehlerfall behandelt?
- [ ] Läuft der Test schnell & isoliert?

Gute Tests machen refactoring sicher, Fehler sichtbar und Teamarbeit zuverlässiger – sie sind das Rückgrat für langfristige Codequalität.