Spickzettel: Python – `pytest` Framework

Ziel

Moderne, schlanke und erweiterbare Tests mit `pytest` schreiben – minimaler Code, maximale Flexibilität.

Vorteile von `pytest`

- Kein Boilerplate-Code notwendig
- Erkennt Funktionen, die mit `test_` beginnen
- Unterstützt einfache Funktionen und komplexe Fixtures
- Erweiterbar durch Plugins (`pytest-cov`, `pytest-mock`, `pytest-django` …)

Beispiel-Test mit `pytest`

```
def add(a, b):
    return a + b

def test_add():
    assert add(2, 3) == 5
```

Häufige Features

Funktion	Bedeutung
`assert`	Direkte Python-Assertions
`pytest.raises()`	Erwarteter Fehler
Fixtures	Setup/Teardown über Parameter
Markierungen	z. B. `@pytest.mark.slow`, `@pytest.mark.parametrize`

Testausführung

```
pytest                  # alle Tests im Projekt
pytest test_modul.py    # gezielter Testlauf
pytest -v               # ausführlicher (verbose)
```

Best Practices

- Testdateien beginnen mit `test_`, Funktionen auch
- Modularisierung durch Fixtures (`conftest.py` für globale Fixtures)
- Coverage mit `pytest --cov=modulname`

- Für größere Projekte: Ordner `tests/` mit klarer Struktur

`pytest` ist ideal für moderne Python-Projekte: klar, leistungsfähig und einfach zu integrieren.