Python Testing mit pytest

Python-Code systematisch und wartbar testen

01 INTRO

Automatisierte Tests gehören zu professioneller Softwareentwicklung. pytest bietet dafür einen schlanken Einstieg, unterstützt aber ebenso umfangreiche Testsuiten mit Fixtures, Parametrisierung, Mocking und einer klaren Strukturierung der Tests.

Der Kurs vermittelt den praktischen Einsatz von pytest von den ersten Unit Tests bis zum Aufbau wartbarer Testsuiten. Im Mittelpunkt stehen aussagekräftige Tests, sinnvolle Teststrukturen und der gezielte Umgang mit Abhängigkeiten.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Sichere Python-Grundkenntnisse

02 TRAININGSPROGRAMM

Inhalte

Einstieg in pytest

- Aufgaben automatisierter Tests
- Installation und Projektaufbau
- Test Discovery
- Aufbau eines pytest-Tests
- Assertions
- Tests ausführen und auswählen
- Testergebnisse verstehen

Gute Tests schreiben

- Arrange – Act – Assert
- kleine und fokussierte Tests
- aussagekräftige Testnamen
- positive und negative Testfälle
- Exceptions testen
- Floating-Point-Vergleiche
- typische Fehler in Unit Tests

Fixtures

- wiederverwendbare Testvorbereitung
- Fixture Injection
- Setup und Teardown
- Fixture Scopes
- Abhängigkeiten zwischen Fixtures
- yield-Fixtures
- conftest.py
- eingebaute pytest-Fixtures

Parametrisierung

- @pytest.mark.parametrize
- mehrere Testdaten
- Kombination von Parametern
- lesbare Test-IDs
- parametrisierte Fixtures
- datengetriebene Tests

Tests strukturieren

- Tests nach Modulen und Funktionen organisieren
- Testklassen sinnvoll einsetzen
- gemeinsame Fixtures
- Marker
- Tests auswählen und gruppieren
- Unit-, Integrations- und andere Testebenen trennen
- Struktur größerer Testsuiten

Abhängigkeiten isolieren

- warum Abhängigkeiten Tests erschweren
- Test Doubles
- Stubs, Fakes und Mocks
- unittest.mock
- Mock und MagicMock
- Rückgabewerte und Seiteneffekte
- Aufrufe überprüfen
- monkeypatch
- wann Mocking sinnvoll ist – und wann nicht

Dateien, Daten und externe Ressourcen testen

- temporäre Dateien und Verzeichnisse
- tmp_path
- Konfigurationen testen
- Umgebungsvariablen
- Testdaten bereitstellen
- Dateisystemzugriffe isolieren
- externe Ressourcen kontrolliert einbinden

Fehler und Exceptions testen

- erwartete Exceptions
- pytest.raises
- Fehlermeldungen überprüfen
- Warnings testen
- Fehlerpfade gezielt abdecken

Testqualität und Coverage

- was Code Coverage aussagt
- was Coverage nicht aussagt
- Coverage mit pytest
- fehlende Testfälle erkennen
- Branch Coverage
- Coverage als Hilfsmittel statt Zielgröße

Tests diagnostizieren

- ausführliche pytest-Ausgaben
- fehlgeschlagene Assertions analysieren
- Output und Logging untersuchen
- Debugging fehlgeschlagener Tests
- einzelne Tests gezielt ausführen
- typische Ursachen instabiler Tests

Wartbare Testsuiten

- Duplikation in Tests vermeiden
- Testhilfsfunktionen und Fixtures
- Testdaten sinnvoll organisieren
- Tests voneinander unabhängig halten
- schnelle und langsame Tests unterscheiden
- fragile Tests erkennen
- Tests als ausführbare Dokumentation

pytest im Entwicklungsprozess

- Tests lokal ausführen
- automatisierte Testläufe
- Exit Codes
- Reports und Testergebnisse
- grundlegende Einbindung in CI/CD
- sinnvolle Testauswahl für unterschiedliche Teststufen

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	3 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung

Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner mit aktueller Python-Version und Entwicklungsumgebung

> **Inhouse-Schulungen**
> Die Schwerpunkte können an vorhandene Kenntnisse, eingesetzte Technologien und konkrete Anforderungen des Teams angepasst werden.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Der Kurs richtet sich an Python-Entwicklerinnen und -Entwickler, die ihren Code automatisiert testen und mit pytest zuverlässige und wartbare Testsuiten aufbauen möchten.

Er eignet sich sowohl für Teilnehmer, die bisher nur einzelne Tests geschrieben haben, als auch für Entwickler, die pytest bereits einsetzen und ihre Tests systematischer strukturieren möchten.

Die Teilnehmer sollten mit folgenden Themen sicher umgehen können:

- grundlegende Python-Syntax
- Datenstrukturen
- Funktionen
- Module und Packages
- Exceptions
- grundlegende objektorientierte Programmierung

> **Testerfahrung**
> Erfahrung mit automatisierten Tests oder pytest ist nicht erforderlich.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- automatisierte Tests mit pytest erstellen und ausführen
- aussagekräftige und fokussierte Testfälle entwickeln
- Fixtures zur Vorbereitung und Wiederverwendung von Testkontexten einsetzen
- Tests und Fixtures parametrisieren
- Testsuiten sinnvoll strukturieren
- externe Abhängigkeiten mit geeigneten Test Doubles isolieren
- Mocking und monkeypatch gezielt einsetzen
- Fehler- und Ausnahmefälle systematisch testen
- Coverage-Ergebnisse sinnvoll interpretieren

- fehlgeschlagene Tests systematisch analysieren
- wartbare und voneinander unabhängige Tests entwickeln
- pytest in einen automatisierten Entwicklungsprozess integrieren

06 FAQ

Häufige Fragen

Ist der Kurs nur für Unit Tests gedacht?

Nein. Unit Tests bilden einen Schwerpunkt, pytest eignet sich aber ebenso für Integrations- und andere automatisierte Tests. Der Kurs behandelt deshalb auch die Strukturierung unterschiedlicher Testebenen.

Muss ich pytest bereits kennen?

Nein. Der Kurs beginnt mit den grundlegenden Mechanismen und führt anschließend zu Fixtures, Parametrisierung, Mocking und größeren Testsuiten.

Wird auch Mocking behandelt?

Ja. Dabei geht es nicht nur um die Bedienung von unittest.mock, sondern auch darum, wann ein Mock sinnvoll ist und wann andere Test Doubles oder echte Komponenten die bessere Lösung sind.

Wird Test Driven Development behandelt?

TDD kann anhand einzelner Übungen demonstriert werden, ist aber nicht der Schwerpunkt des Kurses. Im Mittelpunkt steht der systematische Einsatz von pytest.

Wird CI/CD behandelt?

Die grundlegende Integration automatisierter pytest-Läufe wird behandelt. Aufbau und Architektur vollständiger CI/CD-Pipelines sind nicht Schwerpunkt dieses Kurses.

Kann der Kurs an unseren vorhandenen Python-Code angepasst werden?

Ja. Bei Inhouse-Schulungen können Beispiele, Übungen und Schwerpunkte an die vorhandene Codebasis, Teststrategie und Entwicklungsumgebung angepasst werden.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/python-pytest/