Objektorientierte Programmierung mit Python

Objektorientierte Konzepte verstehen und mit Python sinnvoll einsetzen

01 INTRO

Python unterstützt objektorientierte Programmierung, ohne sie zu erzwingen. Dadurch lassen sich objektorientierte Konzepte gezielt dort einsetzen, wo sie Struktur schaffen und die Weiterentwicklung einer Anwendung erleichtern.

Der Kurs vermittelt die objektorientierten Mechanismen von Python von Klassen und Vererbung bis zu Komposition, Protokollen und dem Python-Datenmodell. Im Mittelpunkt steht dabei nicht nur die Syntax, sondern die Frage, wie Klassen und Objektmodelle sinnvoll entworfen werden.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Sichere Python-Grundkenntnisse

02 TRAININGSPROGRAMM

Inhalte

Klassen und Objekte

- Klassen definieren und Objekte erzeugen
- Attribute und Methoden
- Instanz- und Klassenattribute
- self und Objektzustand
- Konstruktion und Initialisierung
- Klassen als eigene Datentypen

Attribute kontrollieren

- öffentliche und interne Attribute
- Properties
- Getter und Setter in Python
- berechnete Attribute
- Validierung von Attributen
- __slots__ im Überblick

Klassenmethoden und statische Methoden

- Instanzmethoden
- @classmethod
- @staticmethod
- alternative Konstruktoren
- geeignete Einsatzgebiete

Vererbung

- Basisklassen und abgeleitete Klassen
- Methoden überschreiben
- super()
- abstrakte Basisklassen
- Mehrfachvererbung
- Method Resolution Order
- Grenzen und Risiken von Vererbung

Komposition statt Vererbung

- Objekte aus anderen Objekten zusammensetzen
- Delegation
- lose Kopplung
- Verantwortlichkeiten sinnvoll verteilen
- Vererbung und Komposition gegeneinander abwägen

Das Python-Datenmodell

- Special Methods und Protokolle
- __repr__ und __str__
- Vergleich und Gleichheit
- Hashbarkeit
- Container-Verhalten
- Callable Objects
- eigene Klassen in Python integrieren

Dataclasses

- @dataclass
- automatisch erzeugte Methoden
- Default Values und Factories
- unveränderliche Dataclasses
- Vergleich und Ordnung
- Dataclasses als einfache Domänenobjekte

Interfaces und Polymorphie in Python

- Duck Typing
- Polymorphie ohne gemeinsame Basisklasse
- abstrakte Basisklassen
- Protocol
- strukturelle Typisierung
- Abhängigkeit von Verhalten statt konkreten Klassen

Beziehungen zwischen Objekten

- Assoziation
- Aggregation und Komposition
- Objektgraphen
- Lebenszyklen und Verantwortlichkeiten
- bidirektionale Beziehungen
- typische Modellierungsprobleme

Entwurf objektorientierter Modelle

- Verantwortlichkeiten von Klassen
- Kohäsion und Kopplung
- kleine und fokussierte Schnittstellen
- Tell, Don't Ask
- Law of Demeter
- typische Code Smells
- wann eine Klasse nicht die richtige Lösung ist

Objektorientierung und funktionale Elemente

- Funktionen als First-Class Objects
- Funktionen statt zustandsloser Klassen
- immutable Daten
- Comprehensions und Generatoren
- objektorientierte und funktionale Ansätze kombinieren

Praktische Modellierung

- Anforderungen in Klassen und Verantwortlichkeiten übersetzen
- ein kleines Domänenmodell entwickeln
- bestehendes Modell analysieren und verbessern
- Erweiterbarkeit berücksichtigen
- Refactoring eines gewachsenen Objektmodells

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 objektorientierte Programme systematisch entwerfen und die objektorientierten Möglichkeiten von Python gezielt einsetzen möchten.

Er eignet sich sowohl für Teilnehmer, die Objektorientierung bisher hauptsächlich über die grundlegende Klassensyntax kennen, als auch für Entwickler mit OOP-Erfahrung in Java, C# oder anderen Sprachen, die die Besonderheiten von Python kennenlernen möchten.

Die Teilnehmer sollten mit folgenden Themen sicher umgehen können:

- grundlegende Python-Syntax
- Datentypen und Datenstrukturen
- Bedingungen und Schleifen
- Funktionen
- Module
- einfache Klassen und Objekte

Objektorientierte Vorkenntnisse
Grundkenntnisse objektorientierter Programmierung sind hilfreich, aber nicht erforderlich.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- Klassen und Objekte in Python sicher modellieren
- Attribute und Objektzustände kontrolliert verwalten
- Vererbung und Komposition bewusst einsetzen
- Polymorphie und Duck Typing verstehen und nutzen
- abstrakte Basisklassen und Protocols einsetzen
- wichtige Mechanismen des Python-Datenmodells verwenden
- Dataclasses sinnvoll einsetzen
- Verantwortlichkeiten zwischen Klassen sinnvoll verteilen
- Kopplung und Kohäsion objektorientierter Modelle beurteilen
- typische Schwächen bestehender Objektmodelle erkennen und verbessern
- entscheiden, wann objektorientierte und wann einfachere Python-Konstrukte geeigneter sind

06 FAQ

Häufige Fragen

Ist der Kurs auch geeignet, wenn ich Objektorientierung bereits aus Java oder C# kenne?

Ja. Die grundlegenden Konzepte sind übertragbar, Python setzt sie jedoch teilweise anders um. Der Kurs legt deshalb besonderen Wert auf Python-spezifische Mechanismen wie Duck Typing, Protocols, Dataclasses und das Python-Datenmodell.

Muss ich bereits objektorientiert mit Python programmiert haben?

Nein. Sichere Python-Grundkenntnisse reichen aus. Die objektorientierten Konzepte werden systematisch aufgebaut.

Werden Design Patterns behandelt?

Einzelne Entwurfsmuster können im Zusammenhang mit konkreten Modellierungsproblemen auftreten. Der Schwerpunkt liegt jedoch auf objektorientierter Modellierung und den Python-spezifischen Möglichkeiten, nicht auf einem Katalog klassischer Design Patterns.

Geht es auch um SOLID?

Die zugrunde liegenden Fragen wie Verantwortlichkeiten, Kopplung, Erweiterbarkeit und geeignete Schnittstellen werden behandelt. SOLID kann dabei als Orientierung dienen, wird aber nicht schematisch auf Python übertragen.

Kann der Kurs mit anderen Python-Themen kombiniert werden?

Ja. Je nach Zielsetzung lassen sich beispielsweise Inhalte aus „Python für Fortgeschrittene“, „Clean Python“ oder „Python Testing mit pytest“ ergänzen oder vertiefen.

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