

Python für Experten

Komplexe Code-Beispiele

Das Ziel dieses Python Trainings ist es, Programmierern, die mit den Grundlagen von Python vertraut sind und bereits mit Python gearbeitet haben, fortgeschrittene Python-Themen zu vermitteln. Das betrifft die Arbeit mit IDEs, erweiterte Objektorientierung, fortgeschrittene Programmier Techniken und praktische Anwendungen. Alle Themen werden durch umfangreiche Übungen unterstützt.

Kursinhalt

- Fortgeschrittene Objektorientierung
- Klassen optimieren
- Pythons Datenmodell
- Interfaces, Protocols und ABCs
- Design Prinzipien der OOP
- Kontextmanagers
- Multiprocessing, Threading und Asyncio
- Metaprogrammierung
- Deskriptoren und Metaklassen

Alle Themen werden durch umfangreiche Übungen unterstützt. Sie arbeiten während des Kurses mit dem Trainer zusammen und verwenden Pythons interaktive JupyterLabs. Gerne können Sie Beispiele aus ihrer täglichen Arbeit mitbringen. Wenn es die Zeit erlaubt, wird der Trainer diese in den Kurs einbeziehen. Wir bitten um die Bereitstellung solcher Beispiele schon vor dem Kurs.

Zu diesem Python Training erhalten Sie die Unterlagen in elektronischer Form in englischer Sprache.

Zielgruppe

Der Kurs richtet sich an alle mit Interesse an Advanced-Themen im Bereich Python. Er nutzt umfangreiche interaktive Übungen zur direkten Anwendung und anschließenden Festigung des Verständnisses.

Voraussetzungen

Gute Python Kenntnisse werden vorausgesetzt. Diese können z. B. im Kurs Python für Programmierer – Grundlagen für den schnellen Umstieg erworben werden. Alternativ dient auch der Kurs Python für Fortgeschrittene – Objektorientierung, Tools und Best Practice als Vorbereitung.

Dieser Kurs im Web



Alle tagesaktuellen Informationen und Möglichkeiten zur Bestellung finden Sie unter dem folgenden Link: www.experteach.ch/go/PYFF

Vormerkung

Sie können auf unserer Website einen Platz kostenlos und unverbindlich für 7 Tage reservieren. Dies geht auch telefonisch unter 06074 4868-0.

Garantierte Kurstermine

Für Ihre Planungssicherheit bieten wir stets eine große Auswahl garantierter Kurstermine an.

Ihr Kurs maßgeschneidert

Diesen Kurs können wir für Ihr Projekt exakt an Ihre Anforderungen anpassen.

Training	Preise zzgl. MwSt.
Termine in Deutschland	4 Tage CHF 2.855,-
Online Training	4 Tage CHF 2.855,-
Termin/Kursort	Kurssprache Deutsch
16.09.-19.09.24 Düsseldorf	16.09.-19.09.24 Online

Stand 14.04.2024



Inhaltsverzeichnis

Python für Experten – Komplexe Code-Beispiele

- 1 Fortgeschrittene Objektorientierung**
 - 1.1 Das Python Data Modell**
 - 1.1.1 Beispiel: Französisches Kartendeck
 - 1.1.2 Das Collection API
 - 1.1.3 Überblick: Special Methods
 - 1.2 Einfache Klassen mit NamedTuple und Dataclass**
 - 1.2.1 NamedTuples
 - 1.2.2 Dataclasses
 - 1.3 Klassen optimieren mit __slots__**
 - 1.3.1 Schnellere Attribut-Operationen
 - 1.3.2 Slots brauchen weniger Speicher
 - 1.3.3 Keine Multiple Vererbung mehr
 - 1.4 Interfaces, Protocols und ABCs**
 - 1.4.1 Protokolle als informelle Interfaces
 - 1.4.2 Explizite Interfaces mit ABCs
- 2 Design Prinzipien**
 - 2.1 Das Zen von Python**
 - 2.2 Einfache Design-Prinzipien**
 - 2.3 SOLID Designprinzipien**
 - 2.3.1 Single Responsibility Prinzip
 - 2.3.2 Open-Closed Prinzip
 - 2.3.3 Liskov Substitutions Prinzip
 - 2.3.4 Interface Segregations Prinzip
 - 2.3.5 Dependency Inversion Prinzip
 - 2.4 Designmuster**
 - 2.4.1 Klassifizierung von Designmustern
 - 2.5 Kreative Designmuster**
 - 2.5.1 Das Singleton Muster
 - 2.5.2 Das Factory Muster
 - 2.6 Strukturelle Designmuster**
 - 2.6.1 Das Adapter Muster
 - 2.6.2 Das Proxy Muster
 - 2.7 Verhaltensorientierte Designmuster**
 - 2.7.1 Das Observer Muster
 - 2.7.2 Das Mediator Muster
- 3 Kontext Managers**
 - 3.1 Das with-Statement**
 - 3.1.1 Übliche Anwendungen von Kontextmanagern
 - 3.1.2 Das Kontextmanager Interface
 - 3.1.3 Mehrere Kontextmanager öffnen
 - 3.2 Eigene Kontext Manager Klassen**
 - 3.3 Das Modul contextlib**
 - 3.3.1 Klassen und Dekoratoren in contextlib
- 3.4 Kontext Managers als Dekoratoren**
- 4 Parallele Programmierung in Python**
 - 4.1 Einleitung**
 - 4.2 Prozesse, Threads und Asynchronität**
 - 4.3 Python Globales Interpreter Lock (GIL)**
 - 4.4 Auswahl der Parallelisierungs-API**
 - 4.4.1 CPU vs. IO Lastigkeit
 - 4.4.2 Threading vs. asyncio
 - 4.4.3 Pools vs. Klassen
 - 4.4.4 Pools vs. Executors
 - 4.5 Multiprocessing**
 - 4.5.1 Die Process Klasse
 - 4.6 Threading**
 - 4.6.1 Datenaustausch zwischen Threads
 - 4.6.2 Fuzzing gegen Race-Conditions
 - 4.6.3 Kritische Bereiche absichern mit Locks
 - 4.7 Asynchronität**
 - 4.7.1 Python-Coroutinen
 - 4.7.2 Synchronisation mit Events
 - 4.7.3 Ergebnisse aus Coroutinen - Futures
 - 4.7.4 Bibliotheken für die Asynchrone Programmierung
- 5 Metaprogrammierung**
 - 5.1 Attribut Deskriptoren**
 - 5.2 Attributhandhabung - Eine Übersicht**
 - 5.2.1 Special Attributes
 - 5.2.2 Builtin-Funktionen zur Attributhandhabung
 - 5.2.3 Special Methods zur Attributhandhabung
 - 5.3 Das Deskriptor-Protokoll**
 - 5.4 Eigene Deskriptoren**
 - 5.4.1 Überschreibende und Nichtüberschreibende Deskriptoren
 - 5.4.2 Methoden sind Deskriptoren
 - 5.4.3 Deskriptoren - Best Practices
 - 5.5 Klassen als Objekte**
 - 5.5.1 type als Klassen-Factory
 - 5.5.2 Wie Datenklassen wirklich funktionieren
 - 5.5.3 Klassendekoratoren als Metaprogrammierung
 - 5.6 Metaklassen**
 - 5.7 Was passiert wann?**

