MICRO CONSULT

Stand 20.05.2026

Design Patterns mit Python - Live-Online-Training

**Ziele -
Ihr Nutzen**

Sie erhalten einen tiefen Einblick in die klassischen Design Patterns.

Dieser Einblick umfasst die Begrifflichkeit für die eindeutige Kommunikation mit Ihren Kollegen und das bessere Verständnis Ihrer Softwarearchitektur.

Ausgehend von der Begrifflichkeit erlernen Sie in dem Training die Anwendung der wichtigsten Design Patterns.

Diese schließt vor allem die besondere Charakteristik der objektorientierten Programmiersprache Python ein.

Abgerundet wird das Training durch eine Vorstellung der bekanntesten Architektur-Patterns, die typischerweise helfen, die Architektur des Gesamtsystems mithilfe von Design Patterns zu implementieren.

Teilnehmer

Softwareentwickler und Softwarearchitekten

Voraussetzungen

Kenntnisse, wie sie im Training "Python: Objektorientierte Skriptsprache" vermittelt werden.

Live Online Training

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-PYTH-DP

Präsenz-Training - Deutsch

Dauer
3 Tage

Design Patterns mit Python - Live-Online-Training

Inhalt

Design Patterns (DP)

- Definition: Was sind Design Patterns?
- Ursprünge der Design Patterns
- Vorteile und Nutzen der Anwendung von Design Patterns (Was zeichnet sie aus?)
- Was sind Anti-Patterns?
- Unterschiede zwischen Patterns und Design Patterns
- Komponenten und Architektur von Patterns
- Qualitätsmerkmale von Patterns
- GoF - Gang of Four Design Patterns
- Entwickeln von Patterns

Die wichtigsten Design Patterns: Zweck, Funktion, Anwendbarkeit, Beispiele

- Factory Method - Fabrikmethode: Delegation der Erzeugung von Objekten an Unterklassen
- Singleton: Garantiert, dass eine Klasse nur ein Exemplar besitzt

- Adapter: Übersetzung einer Schnittstelle in eine andere Schnittstelle
- Bridge - Brücke: Entkopplung des Interfaces von der Implementierung
- Decorator: Dynamische Erweiterung von Zuständigkeit eines Objektes (Wrapper)
- Composite - Kompositum: Zusammenfügen von Objekten zu Baumstrukturen
- Proxy - Stellvertreter: Kapselung von dem Zugriff auf ein Objekt
- Observer - Beobachter: Definition von 1-zu-n-Abhängigkeiten zwischen Objekten
- Visitor - Besucher: Definition neuer Operationen ohne Änderung der Objektstruktur
- Template Method - Schablonenmethode: Kapselung einer Operation, die auf einer Objektstruktur ausgeführt wird, als Objekt
- Strategy - Strategie: Definition einer Familie von Algorithmen und Kapselung in Objekten

Anwendung der Design Patterns

- Python-spezifische Idiome
- Idiome: Besonderheiten, abstrakte Methode, abstrakte Klassen, Fabrikmethode, Singleton, Adapter, Stellvertreter, Schablonenmethode, Strategie
- Dekoratoren: Geschichte, Vorteile, Implementierung, Besonderheiten, Beispiele
- Generatoren: Details, Iterator-Protokoll, Konsument, Beispiel

Architektur-Patterns

- Layers - Layered Achitecture Patterns
- Pipes and Filters
- Broker: Einführung eines Vermittlers (Brokers) zwischen Client und Server
- Model View Controller (MVC)
- Publish Subscriber
- Reactor - Lösung für eventgetriebene Anwendungen: Gleichzeitige Annahme von Clientanfragen und Verteilung an Serviceanbieter (Server)

Übung im Python Design Patterns Workshop

- Ein hoher Übungsanteil vermittelt praxisnahe Kenntnisse für die Python Design Patterns