

## ***PL106 Python-Programmierung Advanced (Unix/Linux)***

### **Kurzbeschreibung:**

Mit Python ist es möglich, auch komplexere Programme übersichtlich und effizient zu verwirklichen. Der Kurs **PL106 Python-Programmierung Advanced (Unix/Linux)** vermittelt das nötige Hintergrundwissen und die dazugehörige Programmierpraxis.

### **Zielgruppe:**

Das Training **PL106 Python-Programmierung Advanced (Unix/Linux)** richtet sich an:

- System-Administratoren
- Datenbank-Administratoren
- Applikations-Administratoren
- Netzwerk-Administratoren

### **Voraussetzungen:**

Um dem Lerntempo und den Lerninhalten des Kurses **PL106 Python-Programmierung Advanced (Unix/Linux)** gut folgen zu können, benötigen Sie Kenntnisse auf Administrationsebene von Unix/Linux-Systemen.

Wir empfehlen vorab den Besuch des Trainings [PL105 Python-Programmierung Basics \(Unix/Linux\)](#).

### **Sonstiges:**

**Dauer:** 5 Tage

**Preis:** 2790 Euro plus Mwst.

### **Ziele:**

Im Workshop **PL106 Python-Programmierung Advanced (Unix/Linux)** lernen Sie fortgeschrittene Programmier Techniken von Python zu verstehen, einzusetzen und zu optimieren. Folgende Kenntnisse und Fähigkeiten werden vermittelt:

- Funktionen,
- Hierarchische Klassen,
- Performance-Tuning,
- Debugging

Zahlreiche Beispiele überführen das Verständnis in praxisnahe Codes. Der Kurs basiert auf Python 3.x.

## Inhalte/Agenda:

- ♦ **Python-Internia**
  - ♦ · Mutable vs. Immutable und damit verbundene Irrtümer
  - ♦ · Magic Methods und ihre Zuordnung zu Operatoren und Funktionen
- ♦ **Performance-Optimierung**
  - ♦ · Generatoren und Iteratoren statt vollständiger Objekt-Listen
  - ♦ · Messung von Ausführungszeit, Memory- und CPU-Last
  - ♦ · Paralleles Processing und Multithreading
  - ♦ · Effiziente im Unterschied zu belastenden Code-Beispielen
- ♦ **Knapper Code**
  - ♦ · Comprehensions für Listen, Dictionaries, Generatoren und Sets
  - ♦ · Lambda-Funktionen, z.B. bei fortgeschrittenem Sortieren
  - ♦ · Collections statt der Basis-Datentypen
  - ♦ · Modularisierung über eigene Module und Pakete
- ♦ **Fortgeschrittene Erstellung eigener Funktionen**
  - ♦ · Funktionen mit Default-Parametern und flexibler Parameter-Liste
  - ♦ · Annotations in Funktions-Parametern
  - ♦ · Geltungsbereich von Objekten innerhalb und außerhalb einer Funktion
  - ♦ · Funktions-Dekoratoren
  - ♦ · "yield" statt "return"
- ♦ **Objektorientierung**
  - ♦ · Klassen, Instanzen, Metaklassen und Vererbung
  - ♦ · Attribute und Methoden zu Klassen
  - ♦ · Setter, Getter und Property-Attribute
  - ♦ · Erstellung eigener Klassen
- ♦ **Diverses**
  - ♦ · Advanced Regular Expressions: Look-Arounds, greedy vs. non-greedy, compile
  - ♦ · Eigene Exceptions definieren und auslösen
  - ♦ · Neue Features ab Python 3.6
- ♦ **Debugging**