Python – Grundlagen

Python systematisch lernen und praktisch einsetzen

01 INTRO

Python ermöglicht einen schnellen Einstieg in die Programmierung, wird aber ebenso für professionelle Anwendungen, Automatisierung, Datenverarbeitung, Webentwicklung und zahlreiche weitere Aufgaben eingesetzt.

Der Kurs vermittelt Python von Grund auf und legt dabei Wert auf mehr als die reine Syntax. Die Teilnehmer lernen, Programme selbstständig zu entwickeln, sinnvoll zu strukturieren, Fehler systematisch zu finden und die wichtigsten Werkzeuge des Python-Ökosystems einzusetzen.

DAUER

5 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Keine Python-Kenntnisse erforderlich

02 TRAININGSPROGRAMM

Inhalte

Einstieg in Python

- Eigenschaften und Einsatzgebiete von Python
- Python-Versionen und Implementierungen
- Interpreter und interaktive Shell
- Python-Programme ausführen
- Entwicklungsumgebung einrichten
- Quelltexte und Dateien
- erste Python-Programme
- Fehlermeldungen lesen und verstehen

Entwicklungsumgebung und Python-Ökosystem

- Python installieren und verwalten
- virtuelle Umgebungen
- venv
- Pakete mit pip installieren
- PyPI
- Abhängigkeiten eines Projekts
- Interpreter und virtuelle Umgebung in der IDE
- grundlegende Projektstruktur

Variablen und Datentypen

- Namen und Variablen
- dynamische Typisierung
- Ganzzahlen und Fließkommazahlen
- boolesche Werte
- Strings
- None
- Typen untersuchen und konvertieren
- Operatoren
- Ausdrücke und Zuweisungen

Arbeiten mit Strings

- Strings erzeugen und verarbeiten
- Indizierung und Slicing
- wichtige String-Methoden
- Suchen und Ersetzen
- Aufteilen und Zusammenfügen
- Formatierung mit f-Strings
- Escape-Sequenzen
- Unicode und Zeichencodierung im Überblick

Listen und Tupel

- Sequenzen
- Listen erzeugen und verändern
- Elemente hinzufügen und entfernen
- Indizierung und Slicing
- Listen durchsuchen
- Sortieren
- Tupel
- Packing und Unpacking
- Listen und Tupel sinnvoll unterscheiden

Dictionaries und Sets

- Schlüssel-Wert-Strukturen
- Dictionaries erzeugen und verändern
- auf Werte zugreifen
- über Dictionaries iterieren
- Dictionary-Methoden
- Sets
- Mengenoperationen
- geeignete Datenstruktur auswählen
- verschachtelte Datenstrukturen

Programmabläufe steuern

- if, elif und else
- Vergleichsoperatoren
- logische Operatoren
- Truth Values
- verschachtelte Bedingungen
- for-Schleifen
- while-Schleifen
- range
- break und continue
- Schleifen sinnvoll strukturieren

Iteration und Comprehensions

- über Sequenzen iterieren
- enumerate
- zip
- mehrere Werte gleichzeitig verarbeiten
- List Comprehensions
- Dictionary- und Set-Comprehensions im Überblick
- Comprehension oder klassische Schleife?
- lesbaren Python-Code schreiben

Funktionen

- Funktionen definieren und aufrufen
- Parameter und Argumente
- Rückgabewerte
- Default-Parameter
- Positions- und Keyword-Argumente
- mehrere Werte zurückgeben
- lokale und globale Namen
- Gültigkeitsbereiche
- Funktionen sinnvoll schneiden
- Docstrings

Module und Packages

- Programme auf mehrere Dateien verteilen
- Module importieren
- unterschiedliche Importformen
- eigene Module entwickeln
- Standardbibliothek verwenden
- __name__ und __main__
- Packages im Überblick
- wiederverwendbaren Code organisieren
- Importfehler verstehen

Dateien und Verzeichnisse

- Textdateien öffnen und schließen
- Dateien lesen und schreiben
- with und Context Manager
- Zeichencodierungen
- pathlib
- Pfade und Verzeichnisse
- Dateien suchen
- CSV-Dateien
- JSON lesen und schreiben
- typische Fehler beim Dateizugriff

Fehler und Exceptions

- Syntaxfehler und Laufzeitfehler unterscheiden
- Exceptions verstehen
- try und except
- mehrere Exception-Typen
- else und finally
- Exceptions auslösen
- Fehlermeldungen sinnvoll behandeln
- Fehler nicht einfach verschlucken
- Tracebacks lesen

Debugging

- Fehler systematisch eingrenzen
- Fehlermeldungen und Tracebacks auswerten
- Debugger verwenden
- Breakpoints
- Variablen und Programmzustand untersuchen
- schrittweise Programmausführung
- typische Anfängerfehler erkennen
- Debugging statt zufälliger Änderungen

Objektorientierung – Grundlagen

- Klassen und Objekte
- Attribute und Methoden
- self
- Konstruktoren mit __init__
- Instanzen erzeugen
- Objektzustand verändern
- __str__ und __repr__ im Überblick
- einfache Beziehungen zwischen Objekten
- wann eine Klasse sinnvoll ist

Python-Standardbibliothek

- Nutzen der Standardbibliothek
- datetime
- math
- random
- statistics
- pathlib
- collections im Überblick
- Dokumentation gezielt nutzen
- vorhandene Lösungen statt Neuerfindungen

Daten verarbeiten

- strukturierte Daten einlesen
- Daten filtern
- Daten transformieren
- aggregieren und auswerten
- sortieren
- verschachtelte Daten verarbeiten
- Ergebnisse formatieren und ausgeben
- kleine Verarbeitungspipelines aufbauen

Sauberen Python-Code schreiben

- PEP 8 im Überblick
- sprechende Namen
- Funktionen statt langer Programmblöcke
- Duplikation vermeiden
- Konstanten
- Kommentare und Docstrings
- einfache Lösungen bevorzugen
- Lesbarkeit und Wartbarkeit
- Code schrittweise verbessern

Einführung in automatisierte Tests

- warum automatisierte Tests sinnvoll sind
- Testfälle formulieren
- Arrange – Act – Assert
- einfache Tests mit pytest
- erwartete Ergebnisse prüfen
- Fehlerfälle testen
- Funktionen testbar gestalten
- Tests wiederholt ausführen

Logging und Diagnose

- print und Logging unterscheiden
- Python-Modul logging
- Log-Level
- Informationen, Warnungen und Fehler
- aussagekräftige Logmeldungen
- Logging in kleinen Anwendungen
- Fehler nachvollziehbar machen

Kommandozeilenprogramme

- Argumente an Programme übergeben
- sys.argv
- argparse im Überblick
- Optionen und Parameter
- Hilfeausgabe
- Rückgabecodes
- kleine Python-Werkzeuge erstellen

Abschlussprojekt

- eine überschaubare Anwendung selbstständig entwickeln
- Anforderungen in Teilaufgaben zerlegen
- geeignete Datenstrukturen auswählen
- Funktionen und Module strukturieren
- Dateien oder strukturierte Daten verarbeiten
- Fehlerfälle berücksichtigen
- Tests für zentrale Funktionen erstellen
- Anwendung testen und debuggen
- Ergebnisse gemeinsam besprechen und verbessern

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	5 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 Teilnehmerinnen und Teilnehmer, die Python systematisch erlernen und anschließend selbstständig für praktische Programmieraufgaben einsetzen möchten.

Er eignet sich für Einsteiger in Python ebenso wie für Teilnehmer mit Erfahrungen in anderen Programmiersprachen, die einen strukturierten Einstieg in Python suchen.

> **Voraussetzungen**
> Python-Kenntnisse werden nicht vorausgesetzt. Grundlegende Erfahrung im Umgang mit Computern und Dateien sollte vorhanden sein. Allgemeine Programmiererfahrung ist hilfreich, aber nicht erforderlich.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- Python-Programme selbstständig erstellen und ausführen
- eine geeignete Entwicklungsumgebung und virtuelle Umgebungen verwenden
- grundlegende Python-Datentypen und Datenstrukturen gezielt einsetzen
- Programmabläufe mit Bedingungen und Schleifen steuern
- Funktionen entwickeln und Programme sinnvoll strukturieren
- Module und grundlegende Packages verwenden
- Dateien sowie CSV- und JSON-Daten verarbeiten
- Fehler und Exceptions kontrolliert behandeln
- Tracebacks verstehen und Fehler mit einem Debugger systematisch untersuchen
- einfache Klassen und Objekte entwickeln
- geeignete Funktionen der Python-Standardbibliothek finden und einsetzen
- lesbaren und wartbaren Python-Code schreiben
- einfache automatisierte Tests mit pytest erstellen
- Logging zur Diagnose von Programmen einsetzen
- einfache Kommandozeilenprogramme entwickeln
- kleinere Python-Anwendungen von der Aufgabenstellung bis zum lauffähigen Programm selbstständig umsetzen

06 FAQ

Häufige Fragen

Ist der Kurs für Programmieranfänger geeignet?

Ja. Python-Kenntnisse oder Erfahrungen mit einer anderen Programmiersprache werden nicht vorausgesetzt. Die Programmierkonzepte werden zusammen mit Python systematisch eingeführt.

Ist der Kurs auch für Entwickler geeignet, die bereits eine andere Sprache kennen?

Ja. Teilnehmer mit Programmiererfahrung können viele grundlegende Konzepte schneller einordnen und sich stärker auf die Besonderheiten von Python konzentrieren.

Wird objektorientierte Programmierung behandelt?

Ja, die grundlegenden Konzepte von Klassen und Objekten werden eingeführt. Die systematische Vertiefung ist Gegenstand von PY-003 „Objektorientierte Programmierung mit Python“.

Werden automatisierte Tests behandelt?

Ja. Die Teilnehmer lernen grundlegende automatisierte Tests mit pytest kennen. Der systematische Aufbau größerer Testsuiten, Fixtures, Parametrisierung und Mocking wird in PY-004 „Python Testing mit pytest“ vertieft.

Werden auch Dateien und Daten verarbeitet?

Ja. Textdateien, CSV und JSON gehören zum Kurs. Damit können bereits während der Schulung praxisnahe kleine Anwendungen entwickelt werden.

Wird eine Entwicklungsumgebung verwendet?

Ja. Die Teilnehmer arbeiten mit einer Entwicklungsumgebung und lernen auch virtuelle Python-Umgebungen, pip und den grundlegenden Umgang mit Python-Paketen kennen.

Kann der Kurs inhaltlich angepasst werden?

Ja. Bei Inhouse-Schulungen können Schwerpunkte, Beispiele und Übungen an die Vorkenntnisse und typischen Aufgaben des Teams angepasst werden.

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