Python für Fortgeschrittene

Objektorientierung, Tools und Best Practice

Programmierkenntnisse erweisen sich, unabhängig vom Anwendungsfall, als zunehmend wertvolle Fähigkeit. Python ist hierbei eine der aktuell am weitest verbreiteten und gefragten Programmiersprachen. Wenn Sie bereits mit den ersten Grundlagen von Python vertraut sind, bringt Sie dieses Python Training auf den nächsten Level.

Sie werden zu Beginn Grundelemente der objektorientierten Programmierung, wichtige Bausteine der Standardbibliothek und tiefergehende Softwarekonzepte kennen lernen. Software unterliegt immer strengeren Qualitätsstandards. Daher werden wir uns mit Themen wie Python Code Design und Code Analysetools beschäftigen.

Python kann auf die unterschiedlichsten Datenquellen zugreifen und Prozesse automatisieren. Am Beispiel von Rest API Abfragen und Datenbankanbindungen werden wir Abfragen und Berechnung durchführen, ohne dabei das Rad neu zu erfinden. Wir bedienen uns vorgefertigter Module und passen diese an unsere Bedürfnisse an.

Kursinhalt

- Grundkonzepte der Objektorientierung
- Dekoratoren, Generatoren
- Wichtige Elemente der Standardbibliotheken
- Fehlerbehebung, Debugging, Logfiles
- Paket- und Abhängigkeitsverwaltung
- Virtuelle Python-Umgebungen
- Statische Codeanalyse
- Testautomatisierung, Testbibliotheken
- GIT Integration in Visual Studio Code

E-Book Sie erhalten das ausführliche deutschsprachige Unterlagenpaket aus der Reihe ExperTeach Networking – Print, E-Book und personalisiertes PDF! Bei Online-Teilnahme erhalten Sie das E-Book sowie das personalisierte PDF.

Zielgruppe

Der Kurs richtet sich an alle, die bereits erste Programmierkenntnisse besitzen und sich nun weiter mit Python und der objektorientierten Programmierung beschäftigen wollen und die vor ersten Projektaufgaben stehen, die sie mit Python realisieren möchten.

Voraussetzungen

Sie benötigen für diesen Kurs bereits solide Python-Grundlagen in Bezug auf Datentypen, Funktionen und Schleifenkonstruktionen. Diese können z. B. in unserem Kurs Python für Einsteiger – Einführung in die Programmierung erworben werden.

Dieser Kurs im Web

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

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	5 TageCHF 3.295,-
Termine in Österreich	5 TageCHF 3.295,-
Online Training	5 TageCHF 3.295,-

Termin/Kursort — Kurssprache Deutsch

10.06.-14.06.24	HY Frankfurt	09.09.-13.09.24	HY Online
10.06.-14.06.24	HY Online	21.10.-25.10.24	HY Online
22.07.-26.07.24	HY München	21.10.-25.10.24	HY Wien
22.07.-26.07.24	HY Online	09.12.-13.12.24	HY Frankfurt
09.09.-13.09.24	HY Düsseldorf	09.12.-13.12.24	HY Online

Stand 21.04.2024

ExperTeach

Python für Fortgeschrittene

Inhaltsverzeichnis

Python für Fortgeschrittene – Objektorientierung, Tools und Best Practice

1 Visual Studio Code und Jupyter
1.1 Was ist Visual Studio Code?
1.1.1 Visual Studio Code vs. Visual Studio
1.1.2 Visual Studio Code unter Windows installieren
1.1.3 Optionen während der Installation
1.1.4 Das Python-Extension Pack für VS Code
1.1.5 Sprachunterstützung von VS Code
1.1.6 Visual Studio Code updaten
1.1.7 Die Oberfläche von VS Code
1.1.8 Die Oberfläche von VS Code - Activity Bar und Sidebar
1.1.9 Das Search Tool
1.1.10 Die Git-Leiste
1.1.11 Git in Visual Studio Code nutzen
1.1.12 Ein Remoterepository in VS Code klonen
1.1.13 Ein lokales Git-Repository anlegen
1.1.14 Lokales Repository mit Remoterepository synchronisieren
1.1.15 GitLens
1.1.16 Neue Dateien in VS Code anlegen
1.1.17 Mit einzelnen Codedateien in Visual Studio Code arbeiten
1.1.18 Visual Studio Code einen Ordner hinzufügen
1.1.19 Mit Workspaces in VS Code arbeiten
1.1.20 Struktur von Workspaces
1.1.21 Auf Remote Terminals entwickeln
1.1.22 Debugging in Visual Studio Code
1.1.23 Einen Linter für Python in VS Code nutzen
1.1.24 Linter in Aktion
1.1.25 Python Code in Visual Studio Code testen
1.1.26 Ein Testframework in VSCode aktivieren
1.1.27 Tests erzeugen und durchführen
1.2 Jupyter Notebooks
1.2.1 Interaktive Code-Zellen
1.2.2 Grafische Oberflächen funktionieren
1.2.3 Exporte in andere Formate und Hilfen
1.2.4 Editieren und Ausführen von Zellen
1.2.5 Shortcuts sind im Markdown Modus gefährlich
1.2.6 Dynamische HTML Seiten einbetten
1.2.7 Variable Inspector als Debugger
1.3 Anaconda - Scientific Power Package für ML
1.3.1 Die Anaconda Foren/Learning - Dashboards
1.4 Jupyter Notebooks im VSC

2 Skriptaufrufe
2.1 Die Kommandozeilen-Optionen
2.1.1 Module aufrufen python -m
2.1.2 pdb: Der Python Debugger
2.1.3 Das timeit-Modul zur Zeitmessung
2.1.4 Mit JSON-Dateien arbeiten: json.tool
2.1.5 Weitere Beispiele: compileall und tkinter
2.2 Argumente
2.2.1 Die Liste sys.argv
2.2.2 Argumente parsen mit dem argparse-Modul
2.3 Die Shebang Line
2.4 Module und Pakete
2.4.1 Top-level code environment
2.4.2 Verwendung der Variable __name__
2.4.3 Wichtige Dateien in Paketen

3 Objektorientierte Programmierung
3.1 Grundprinzipien der OOP
3.1.1 Module und Systeme
3.1.2 Strukturelle Elemente objektorientierter Software
3.1.3 Beziehungen zwischen Objekten
3.1.4 Design Pattern (Entwurfsmuster)
3.2 Objektorientierung und Python
3.2.1 Basisbeispiel
3.2.2 Sichtbarkeiten
3.2.3 Destruktor
3.2.4 Statische Elemente
3.2.5 Mehrfachvererbung

4 Dekoratoren
4.1 Einführung
4.2 Benötigte Konzepte
4.2.1 Funktions-Referenzen
4.2.2 Funktionen in Funktionen
4.2.3 Funktionen als Übergabeparameter
4.2.4 Funktionen als Rückgabewert
4.3 Einfache Dekoratoren
4.3.1 Die Wrapper-Funktion
4.3.2 Eine Funktion dekorieren
4.4 Anwendungsfälle von Dekoratoren
4.4.1 Argumentüberprüfung
4.4.2 Funktionsaufrufe zählen
4.4.3 Aufruf einer Funktion zeigen (Logging)
4.5 Dekoratoren mit Übergabeparametern
4.6 Klassen als Dekoratoren
4.6.1 Eine Klasse als Dekorator benutzen
4.6.2 Dekorator-Klassen mit Übergabeparametern
4.7 Typische Dekoratoren und das Modul functools
4.7.1 Functools Dekoratoren

5 Generatoren und Asynchronität in Python
5.1 Generatoren in Python 3
5.1.1 Eigene range() programmieren
5.1.2 Generator zip()
5.1.3 range() mit richtiger Argumentreihenfolge
5.2 Generator-Funktionen
5.3 Cooperative multitasking and Coroutines
5.4 Asynchrone Code-Ausführung

6 Unittest, Doctest
6.1 Unittest
6.1.1 unittest - Die Klasse TestCase
6.1.2 assert Methoden
6.1.3 setUp und tearDown Methoden
6.1.4 Testen von Fehlermeldungen
6.1.5 Überspringen und erwartete Fehlschläge
6.2 doctest - Docstrings für Tests nutzen
6.2.1 Einfache Verwendung von doctest
6.2.2 Die Ausgabe von doctest
6.2.3 Testen der Methoden einer Klasse
6.2.4 Tests in einer Textdatei
6.3 Test Driven Development (TDD) – Tests zuerst
6.4 Static Code Analysis
6.4.1 Linting mit pylint
6.4.2 Type Checks mit mypy

7 Eigene Exceptions und Logging
7.1 Exceptions in Python
7.1.1 Hierarchie der Builtin Exceptions (Ausschnitt)
7.1.2 Eigene Exceptions definieren
7.1.3 Exception Chaining
7.2 Logging
7.2.1 Modul Logging
7.2.2 Modul Loguru

8 Virtuelle Umgebungen
8.1 Abhängigkeiten von Modulen
8.2 Das Konzept einer virtuellen Umgebung
8.3 Das Tool venv

9 Eigene Packages und der PyPI
9.1 Angeleitete Übung zur Einführung von Packages
9.1.1 Relative Importe - Schritt 2
9.1.2 Eigenes Paket mit __init__.py - Schritt 3
9.2 Eigene Packages auf PyPI veröffentlichen
9.2.1 setup.py
9.2.2 Erstellung der Distribution
9.2.3 Auf PyPI veröffentlichen
9.2.4 Wichtige Zusatzinformationen

10 Code Distribution
10.1 Distribution mit Python Setuptools
10.2 Das Distributions Problem
10.3 Distribution mit PyInstaller
10.3.1 Vorbereitung und Build-Erzeugnisse
10.3.2 Funktionsweise und One-File
10.3.3 Kommandozeilenargumente für PyInstaller
10.3.4 PyInstaller mit Python-Code ausführen

11 GUI mit TKinter und PyQt5
11.1 GUI mit Tkinter
11.2 Die Alles-In-Einem-File-Variante
11.3 PyGubu – Ein Wysiwyg - Editor für Tkinter
11.4 GUI mit PyQt5
11.4.1 PyQt5, PySide2 und Lizenzierung
11.4.2 Erster Start von PyQt5-Designer
11.4.3 Die Oberfläche von Qt-Designer
11.4.4 Layouts in Qt Designer
11.4.5 PyQt5-Designs in Python-Code laden
11.4.6 Buttons und Methoden verknüpfen

ExperTeach AG
Kronenstrasse 11 · 8735 St. Gallenkappel · Telefon: +41 55 420 2591 · Fax: +41 55 420 2592 · info@experteach.ch · www.experteach.ch