

Advanced Python Programmeren

Doelgroep Cursus Advanced Python Programmeren

De cursus Advanced Python Programmeren is bedoeld voor Python developers die meer willen weten over de Python taal en die zich willen bekwamen in geavanceerde aspecten van Python.

Voorkennis Advanced Python

Om aan deze cursus te kunnen deelnemen is kennis van en ervaring met [programmeren in Python](#) vereist.

Uitvoering Training Advanced Python

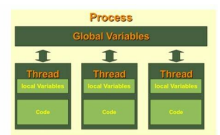
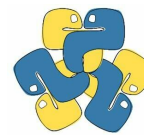
De theorie wordt behandeld aan de hand van presentatie slides. Illustratieve demo's verduidelijken de concepten. De theorie wordt afgewisseld met oefeningen. De cursustijden zijn van 9.30 tot 16.30.

Officieel Certificaat Advanced Python Programmeren

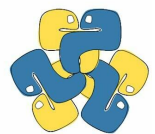
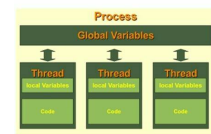
De deelnemers krijgen na het goed doorlopen van de cursus een officieel certificaat Advanced Python Programmeren.

Duur: 4 dagen

Prijs: € 2650

[Open Rooster](#)


Advanced Python Programming



Inhoud Advanced Python Programmeren

In de cursus Advanced Python Programmeren komen geavanceerde aspecten van de programmeertaal Python aan de orde die de development van Python software vereenvoudigen en versnellen.

Advanced Classes

In de eerste plaats komen een aantal geavanceerde aspecten van classes aan de orde zoals multiple inheritance, polymorfisme en operator overloading.

Modules en Packages

Vervolgens wordt aandacht besteed aan het gebruik van modules and packages en leren deelnemers zelf packages te maken, te uploaden en te installeren in een virtuele omgeving. Het benaderen van XML en JSON data staat eveneens op het programma en er wordt besproken hoe logging kan worden geïmplementeerd in Python programma's.

Generators en Decorators

Verder komen iterators aan de orde waarmee lazy evaluation mogelijk wordt, evenals generators en coroutines waarmee concurrent geprogrammeerd kan worden. Dan wordt ingegaan op decorators waarmee functionaliteit zoals caching en proxying aan bestaande functies en classes kan worden toegevoegd.

Design Patterns

In de module patterns wordt de Python implementatie van verschillende standaard Design Patterns behandeld en wordt uitgelegd hoe deze in de Python Library zijn geïmplementeerd. Daarna wordt aandacht besteed aan een advanced feature als meta programming.

Processes en Threads

Eveneens wordt ingegaan op de creatie van processes en threads, synchronisatie tussen threads en het optimaliseren van de performance van Python code. Aansluitend hierop komt de nieuwe asyncio module aan bod, waarmee asynchrone IO met futures kan worden gerealiseerd. Ook inter-procescommunicatie door middel van sockets en pipes staat op het programma.

Unit Testing

En tenslotte komt unit en mock testing aan bod in het kader van test automation.

Modules Advanced Python Programmeren

Module 1 : Advanced Classes	Module 2 : Modules and Packages	Module 3 : XML en JSON Access
Classes Recapitulation Data Hiding Property Syntax Inheritance super Keyword Multiple Inheritance Constructor Chaining Checking Relationships isinstance and issubclass Overriding Methods __str__ and __repr__ Class Methods Operator Overloading Polymorphism	import Statement from ... import Statement Locating Modules Packages in Python Explicit and Implicit Import Namespaces and Scoping Test Harnas Virtual Environments and Activation Distribution of Packages Installing packages pip install Using Python Package index PyPI commands Uploading Package with Setup	XML Parsing Pull versus Push Parsing Python XML Libraries DOM and SAX DOM Navigation and Manipulation XPath Minidom ElementTree Reading and Writing XML Searching and Validating XML XML Manipulation JSON library Dictionary to JSON conversion Loading and Dumping JSON
Module 4 : Logging	Module 5 : Generators	Module 6 : Decorators
logging Module When Use Logging Log Levels Logging Configuration Log in Multiple Modules Formatting Logging Logging Components Logger per Module Handlers and Filters Logging Flow Formatting Logger Adapter	Iteration Iterables Iteration Protocols Supporting Iteration Generators Generator Functions Convenient Iterator Generator Expression Expression Syntax Building Blocks Chaining generators Coroutines	Functions as Objects Passing and Returning Functions What is a Decorator? Decorator Syntax Types of Decorators Passing Arguments Multiple Decorators Class Decorators Singleton Class Why Decorators Need for AOP Crosscutting Security Concern
Module 7 : Patterns in Python	Module 8 : Meta Programming	Module 9 : Threads
What are Patterns? Singleton Pattern Adapter Pattern Chain of Responsibility Pattern Observer Patters Patterns or Principles Everything is Object EAFP Duck Typing Monkey Patching Dependency Injection None Context Managers	Classes as Objects Metaclasses Object from Metaclass Class of Class Descriptor Protocol Lookup Property Functions and Methods Classes and Types Object Creation Metaclass Singleton As Type Object Construction	Thread Characteristics Threads in Python Current Thread Daemon Threads Joining Threads Derived Thread Class Signaling Threads Lock Object Locks as Context Managers Condition Synchronization Barriers Semaphores Thread Local Data
Module 10 : Async IO	Module 11 : Networking	Module 12 : Unit Testing
Concurrent Execution Multiprocessing Subprocess Scheduler Queue AsyncIO Task Future Concurrent Futures Eventloop	Network Layering TCP/IP Layering UDP versus TCP TCPv4 versus TCPv6 sockets Connectionless Services Connection Oriented Services Socket Utility Functions Asynchronous Servers Using Pipes Anonymous and Named Pipes	What is Unit Testing? Automated Testing Test Driven Development Traditional versus TDD Unit Testing in Python Python Test Frameworks Test Cases Assertions Fixture Test Suite