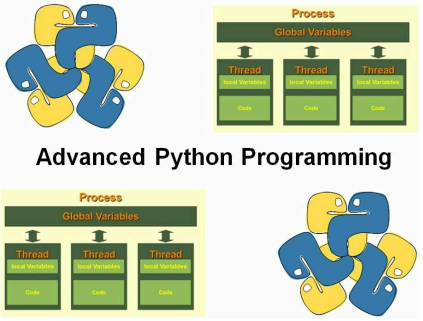
Advanced Python Programmeren

Duur: 4 dagen **Prijs: € 2650**

Open Rooster



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.

Inhoud Cursus 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 Cursus Advanced Python Programmeren

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