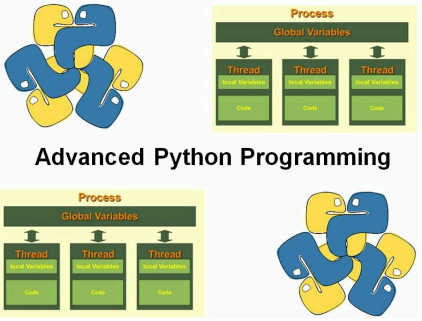
Advanced Python Programming

Duration: 4 days **Price: € 2650**

Open Schedule



Audience Course Advanced Python Programming

The course Advanced Python Programming is intended for Python developers who want to know more about the Python language and who wish to become proficient in advanced aspects of Python.

Prerequisites Course Advanced Python Programming

To participate in this course knowledge of and experience with programming in Python is required.

Realization Training Advanced Python Programming

The theory is discussed on the basis of presentation slides. Illustrative demos illustrate the concepts. The theory is interspersed with exercises. The course material is in English.

Official Certificate Advanced Python Programming

Participants receive an official certificate Advanced Python Programming after successful completion of the course.

Content Course Advanced Python Programming

The course Advanced Python Programming discusses advanced aspects of the Python programming language that simplify and accelerate the development of Python software.

Advanced Classes

In the first place a number of advanced aspects of classes are discussed such as multiple inheritance, polymorphism and operator overloading.

Modules and Packages

Subsequently attention is paid to the use of modules and packages and participants learn how to create packages themselves, how to upload and install them in a virtual environment. Accessing XML and JSON data is also on the program and it is discussed how logging can be implemented in Python programs.

Generators and Decorators

Additionally iterators that deal with lazy evaluation are discussed as well as generators and coroutines with which concurrent programs can be made. Then decorators that are used to add functionality such as caching and proxying to existing functions and classes, are discussed.

Design Patterns

In the module patterns the Python implementation of different standard Design Patterns is explained.
Next attention is paid to an advanced feature like meta programming.

Processes and Threads

Also the creation of processes and threads is discussed as well as the synchronization between threads and optimizing the performance of Python code. Subsequently the new asyncio module is on the course program with which asynchronous IO with futures can be realized. Inter-process communication via sockets and pipes is also treated.

Unit Testing

And finally unit and mock testing is discussed in the context of test automation.

Modules Course Advanced Python Programming

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