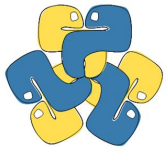
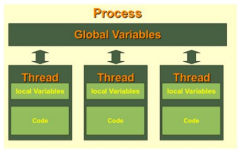
PRG404 : Advanced Python Programmeren

Code : PRG404 **Duur :** 3 dagen **Categorie :** Scripting





Doelgroep :

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

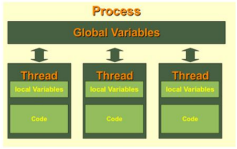
Voorkennis :

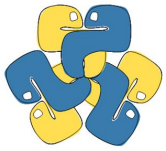
Om aan deze cursus te kunnen deelnemen is kennis van en ervaring met programmeren in Python vereist.

Uitvoering :

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

Advanced Python Programming





Inhoud :

In de cursus Advanced Python komen geavanceerde aspecten van de programmeertaal Python aan de orde die de development van Python software vereenvoudigen en versnellen. De nieuwste versies van Python 2.x en 3.x voegen interessante kenmerken toe aan de taal en in deze cursus leren de deelnemers hoe ze deze kunnen gebruiken. Zo komen iterators aan de orde die lazy evaluation mogelijk maken waarbij een object pas wordt gegenereerd als het nodig is en ook generators en coroutines waarmee concurrent geprogrammeerd kan worden. Vervolgens wordt ingegaan op decorators waarmee functionaliteit zoals caching en proxying, aan bestaande functies en classes kan worden toegevoegd. En ook context managers worden besproken waarmee met het with statement code robuster wordt en exception handling eenvoudiger. In de module patterns wordt de Python implementatie van verschillende Design Patterns behandeld en wordt vervolgens aandacht besteed aan het pythonic principe dat stelt "It's easier to ask for forgiveness than permission (EFAP)". Een principe dat ondersteunt wordt door de robuste exception afhandeling in Python. Voor veel problemen bestaan in Python standaard oplossingen waar in andere omgevingen Design Patterns voor nodig zijn. Deze pythonic oplossingen worden in de module conventions besproken. Verder wordt aandacht besteed aan geavanceerde features zoals meta programming, het gebruik van comprehensions en descriptors. Tenslotte wordt ingegaan op de coding style volgens de Python style guide (PEP8) en het optimaliseren van de performance van Python code.

Module 1 : Iterators and generators

What are Iterators?
Lazy evaluation
yielding versus returning
itertools module
What are Generators?
Generator expressions
Bidirectional communication
Chaining generators
Coroutines

Module 2 : Decorators

What are Decorators?
Tweaking original object
Replacing original object
Decorators on classes
Decorators on functions
Copying the docstring
Examples in library
Deprecation of functions
while-loop removing decorator
Plugin registration system

Module 3 : Context Managers

What are Context managers?
with statement
Catching exceptions
Defining context managers
Using Context managers
Examples standard library
contextlib

Module 4 : Patterns in Python

EFAP principle
Singletons
Singleton variants
null Objects
null versus None
Proxies
Proxy examples
Observer
Publish and subscribe
Constructor

Module 5 : Conventions in Python

Pythonic principles
Out of the box solutions
Wrapping instead of inheritance
Dependency injections
Factories
Duck typing
Monkey patching
Callbacks

Module 6 : Meta Programming

What are meta classes?
Default meta class
Dynamic classes
Creating classes
Creating object
Adding base classes
Adding fields
Adding methods
Meta class hook

Module 7 : Comprehensions

What are comprehensions?
Lambda Operator
Filter
Reduce and Map
Functional Programming
Generator comprehensions
List comprehensions
Dictionary comprehensions
Set comprehensions

Module 8 : Descriptors and Style

Python descriptors
Descriptors protocol
set, get and delete
Property type descriptors
Decorator type descriptors
Run time descriptors
Python style
Style guide PEP8
pylint and pep8.py

Module 9 : Python Performance

Optimization Guidelines
Influencing speed factors
Optimization strategies
Improving algorithms
Caching
Data Structures
Testing speed
Psyco JIT Compiler