Course Outline | Python Programming Advanced

4 day(s)

Take your Python skills further with our Advanced Python course. Delve into object-oriented programming with classes and inheritance, explore Python's object model and metaprogramming. Master design patterns including iterator, decorator, and factory patterns. Sharpen your testing with Python's unit test library and learn test-driven development. Enhance your debugging skills, manage exceptions, and write maintainable code with best practices in coding standards, documentation, and reproducible environments. Learn code re-use through packaging, and optimise for performance with threading and asynchronous programming. Finally, understand functional programming within Python's capabilities.

Course Outline

Object Oriented Programming

- Classes, instances, constructors, attributes and methods
- Python's object model, special methods and protocols
- Properties, slots, abstract classes
- Inheritance and composition, polymorphism
- Meta programming

Design Patterns

- Iterator design pattern, iter() and next(), generators, generator expressions, generator functions, classes as iterators
- Decorator design pattern, functions as first class objects, monkey patching, enclosures, variadic function parameters, decorator factories, decorating a class
- Proxy pattern, lazy loading
- Factory pattern
- Template pattern
- Observer pattern

Automated Testing

- Python's unit test library, pytest
- Test coverage, mocking
- Test driven development (TDD)
- Testing with date and time

Logging, Debugging and Exceptions

- • Simple to advanced logging, logging handlers, configuration
- Debugging tools and tips
- Raising and handling exceptions

Maintainable Code

- Coding standards, type hints, refactoring and linting
- Documentation
- Reproducible coding environments

Code Re-use

- Importing, standard library, Python Package Index (PyPI)
- Packaging your code, submitting a package to PyPI

Optimising

- Optimising steps
- Timing and profiling
- Optimising for CPU usage and for memory usage
- Threads versus processes

- Multi threading and multi processing
- Producer-consumer pattern
- Thread and process pools
- Asynchronous programming

Functional Programming and Python

- Pros, cons and alternatives
- Map, filter, reduce, lambda
- Other FP-related Python functions