Python: Object-Oriented Programming

Hands-on course of 5 days - 35h
Ref.: PYT - Price 2024: CHF2 960 (excl. taxes)
The price for the 2025 session dates may be revised

EDUCATIONAL OBJECTIVES

At the end of the training, the trainee will be able to:

Master the syntax of the Python language

Acquire the essential notions of object-oriented programming

Know and implement different Python modules

Designing graphic interfaces

Implementing tools for testing and evaluating the quality of a Python program

TRAINER QUALIFICATIONS

The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS

The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

TEACHING AIDS AND TECHNICAL RESOURCES

• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.
• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.
• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

TERMS AND DEADLINES

Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES

Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@ORSYS.fr to review your request and its feasibility.

THE PROGRAMME

last updated: 01/2018

1) Syntax of Python language

- Identifiers and references. Coding conventions and naming rules.
- Blocks and comments.
- Available data types.
- Variables, formated display, local and global scope.
- Working with numeric types, working with character strings.
- Working with dynamic tables (list), static tables (tuple) and dictionaries.
- Using files.
- The if/elif/else conditional structure.
- Logical operators and comparison operators.
- while and for iterator loops. Break/continue iteration interrupts.
- The range function.
- Writing and documenting functions.
- Lambda expressions.
- Generators.
- Structuring code into modules.
- Hands-on work¤Installing and getting started with the Python interpreter.

Installing and getting started with the Python interpreter.

2) Object-Oriented Approach

- The principles of the Object paradigm.
- Defining an object (state, behavior, identity).
- The notion of a class, attributes, and methods.
- Encapsulating data.
- Communication between objects.
- Inheritance, transmitting a class's characteristics.
- Notion of polymorphism.
- Association between classes.
- Interfaces.
- Overview of UML.
- Diagrams of classes, sequences, activities, etc.
- Notion of design patterns.

- Hands-on work ¤UML modeling of a simple case study.
UML modeling of a simple case study.

3) Object-Oriented Programming in Python

- The particular features of the Python Object model.
- Writing classes and instantiating them.
- Constructors and destructors.
- Attribute and method access protection.
- The need for the Self parameter.
- Simple inheritance, multiple inheritance, polymorphism.
- Notions of visibility.
- Special methods.
- Introspection.
- Implementing interfaces.
- Best practices and common design models.
- The use of the exception mechanism for error management.
- Hands-on work ¤Exercises in different object-oriented concepts by implementing the case study.
Exercises in different Object-oriented concepts by implementing the case study.

4) Use of StdLib

- Passing arguments on the command line.
- The use of the Python regular expression engine with the "re" module, special characters, cardinality.
- Working with the file system.
- Overview of some important modules of the standard library: “sys”, “os”, “os.path” modules.
- Packaging and installing a Python library.
- Access to the relational database, the operation of the API DB.
- Hands-on work ¤Implementing Python modules: Regular expressions, accessing a database
Implementing Python modules: Regular expressions, access to a database

5) QA tools

- Static code analysis tools (pylint, pychecker).
- Analyzing analysis reports (types of messages, warnings, errors).
- Automatic documentation extraction.
- The Python debugger (step-by-step execution and post-mortem analysis).
- Test-driven development.
- Python unit test modules (Unittest., etc.).
- Automating tests, aggregating tests.
- Code coverage tests, profiling.
- Hands-on work ¤Using the tools pylint and pychecker to check Python code. Implementing unit tests.
Using the tools pylint and pychecker to check Python code. Implementing unit tests.

6) Creating the TkInter HMI

- The principles of programming graphical user interfaces
- Overview of the TkInter library.
- The main containers.
- Overview of the widgets available (Button, Radiobutton, Entry, Label, Listbox, Canvas, Menu, Scrollbar, Text, etc.).
- The window manager.
- Placement of components, different layouts.
- Event management, the "event" object.
- Multi-window applications.
- Hands-on work ¤Designing a graphical user interface with the Tkinter library.
Designing a graphical user interface with the Tkinter library.

7) Python/C interface

- Overview of the Ctypes module.
- Loading a C library.
- Calling a function.
- Rewriting a Python function in C with the Python/C API.
- Creating C modules for Python.
- The Python interpreter in C.
- Using the code profiler.
- Hands-on work¤Calling functions written in C from Python. Creating C modules for Python with Pyrex.

Calling functions written in C from Python. Creating C modules for Python with Pyrex.

8) Conclusion

- Critical analysis of Python.
- Evolution of the language.
- Webography and bibliography elements.

DATES

REMOTE CLASS
2024 : 16 Dec

2025 : 03 Mar, 12 May, 18 Aug,
03 Nov