Course : Campus Atlas - Python, object-oriented programming

Practical course - 5d - 35h00 - Ref. PYE
Price : 2610 € E.T.

NEW

On completion of the course, participants will be able to use the main features of the Python language to develop multiplatform applications. This training program is intended for employees of professional branches covered by the OPCO Atlas.

Teaching objectives

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

- Understand the basics of the Python language and its ecosystem
- Learn the principles of object-oriented programming
- Understanding and using functions and modules
- Design graphic interfaces
- Use tools to test and evaluate the quality of a Python program

Intended audience

For OPCO Atlas members: developers, engineers, project managers with close links to development.

Prerequisites

Basic programming skills (preferably in object-oriented languages).

Practical details

Hands-on work

Practical exercises and/or case studies.

Teaching methods

60% pratique – 40% théorie. Pour optimiser le parcours d'apprentissage, des modules e-learning peuvent être fournis avant et après la session présentielle ou la classe virtuelle, sur simple demande du participant.

Course schedule

PARTICIPANTS

For OPCO Atlas members: developers, engineers, project managers with close links to development.

PREREQUISITES

Basic programming skills (preferably in object-oriented languages).

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.

1 Algorithms, reasoning before design - Pre-training digital learning content

- Introduction to algorithms.
- Basic instructions in pseudo-code.

Digital activities

This online training course teaches you to think before you design a program, and covers the basics of algorithmics. Fundamental instructions in pseudo-code will also be covered.

2 Introduction to Python

- Installation and configuration of the environment.
- Fundamental data types.
- Basic data structures.
- Variables and scope.
- Operators and expressions.
- Python style and conventions (PEP 8).

Hands-on work

Setup de l'environnement. Types et structures. Style et bonnes pratiques.

3 Control structures and functions

- Advanced control structures.
- Defining and using functions.
- Multiple parameters and returns.
- Error and exception handling.
- Python modules and packages.
- Good structuring practices.

Hands-on work

Structures de contrôle. Fonctions et modularité. Gestion des erreurs.

4 Object-oriented approach

- Fundamental principles of OOP.
- Classes and instances.
- Inheritance and composition.
- Polymorphism and abstraction.
- Interfaces and abstract classes.
- Design patterns in Python.

Hands-on work

Introduction à la programmation orientée objet (POO). Création de classes. Design patterns.

5 Advanced object handling

- Properties and descriptors.
- Magic methods.
- Metaclasses and decorators.
- Introspection and reflection.
- Memory management.
- OOP best practices.

Hands-on work

Properties et descripteurs. Méthodes spéciales. Métaprogrammation.

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.

6 Advanced OOP

- Special Python methods.
- Multiple inheritance and method order resolution (MRO).
- Context managers and Python protocols.
- Advanced design patterns.
- Mixins and composition.
- Anti-patterns and common pitfalls.

Hands-on work
Méthodes spéciales avancées. Héritage et composition. Design patterns avancés.

7 Advanced error handling

- Complete exception hierarchy.
- Create custom exceptions.
- Advanced logging and configuration.
- Debugging and troubleshooting.
- Asynchronous error handling.
- Good robustness practices.

Hands-on work
Exceptions personnalisées. Logging avancé. Debugging et tests.

8 Using StdLib

- Command-line arguments and options (argparse).
- Regular expressions (re).
- File system manipulation (os, pathlib).
- Important modules (datetime, collections, itertools).
- Databases with sqlite3.

Hands-on work
Manipulation de fichiers. Expressions régulières. Base de données SQLite.

9 Standard Python library

- sys and os modules for system interaction.
- Advanced file manipulation with pathlib.
- Specialized collections (collections, heapq).
- Regular expressions with re.
- Serialization with pickle and json.
- Competition with threading and multiprocessing.

Hands-on work
Manipulation système. Collections avancées. RegEx et sérialisation.

10 CLI tools and utilities

- Arguments and options with argparse.
- Configuration with configparser.
- Logging in CLI.
- Rich console interfaces.
- System automation.
- CLI tool distribution.

Hands-on work
Parsing d'arguments. Interface utilisateur. Packaging et distribution.

11 Testing and Documentation

- Unit testing with unittest and pytest.
- Test Driven Development (TDD).
- Documentation with docstrings and type hints.
- Generating documentation with Sphinx.
- Code coverage measurement.
- Continuous integration.

Hands-on work
Tests unitaires. Documentation. Couverture et CI.

12 Code quality

- Static analysis tools (pylint, flake8).
- Style conventions (PEP 8).
- Refactoring techniques.
- Code review and peer programming.
- Measures of complexity.
- Automated controls.

Hands-on work
Analyse statique. Refactoring. Revue de code.

13 Graphical interfaces with Tkinter

- Introduction to Tkinter and its alternatives.
- Basic widgets and layouts.
- Event management.
- MVC pattern in GUI.
- Styles and themes.
- Good interface practices.

Hands-on work
Fondamentaux Tkinter. Widgets avancés. Architecture MVC.

14 Advanced interface and GUI testing

- Custom widgets.
- Advanced styles and themes.
- Animation and visual effects.
- Graphical interface testing.
- Accessibility.
- Internationalization.

Hands-on work
Widgets custom. Tests GUI. Internationalisation.

15 Interfacing with C

- Introduction to ctypes and its alternatives.
- C data types and their mapping.
- Call C functions from Python.
- Memory management and pointers.
- Wrapping libraries C.
- Performance and debugging.

Hands-on work
Introduction à ctypes. Appels de fonctions. Performance et debug.

16 Optimization with Cython

- Introduction to Cython and its ecosystem.
- Cython syntax and static types.
- Compilation and build process.
- Optimization of existing Python code.
- Integration with C/C++.
- Profiling and benchmarking.

Hands-on work
Fondamentaux Cython. Optimisation. Integration C/C++

17 Profiling and performance

- Profiling tools (cProfile, line_profiler).
- CPU and memory performance analysis.
- Algorithmic optimization.
- Benchmarking and metrics.
- Memory leaks and garbage collection.
- Production monitoring.

Hands-on work
Profilage CPU. Mémoire et GC. Benchmarking.

18 Optimization best practices

- Python optimization patterns.
- High-performance architectures.
- Monitoring and observability.
- Optimized deployment.
- Horizontal and vertical scalability.
- Load and stress tests.

Hands-on work
Architecture performante. Tests de charge. Monitoring production.

19 Launching the synthesis project

- Presentation of project topics.
- Development methodology.
- Team organization.
- Planning and milestones.
- Architecture and design.
- Quality standards.

Hands-on work
Initialisation projet. Architecture. Setup projet.

20 Development - Phase 1

- Setting up the basic structure.
- Implementation of core features.
- Unit testing and documentation.
- Git workflow and collaboration.
- Code review and quality.
- Progress monitoring.

Hands-on work
Core features. Git workflow. Point d'avancement.

21 Development - Phase 2

- Finalization of functionalities.
- Integration testing.
- User documentation.
- Preparing for deployment.
- Optimization and refactoring.
- Demo preparation.

Hands-on work
Features avancées. Documentation. Préparation démo.

22 Defense and assessment

- Project presentations.
- Technical demonstrations.
- Feedback.
- Assessment of prior learning.
- Discussion perspectives.
- Training feedback.

Hands-on work
Présentations. Revue technique. Perspectives. Clôture de la formation.

23 Python 3, the language basics - Post-training digital learning content

- Introduction.
- Data types.
- Algorithms.
- Data manipulation.

Digital activities

This online training course introduces the essential basics of the Python language to learn how to program efficiently. Participants will study program structure, data types, functions and algorithmic concepts, before moving on to data manipulation and object-oriented programming. Using a database with SQLAlchemy and applying best practices to develop clean, maintainable Python code will also be covered.

24 Python 3, advanced concepts - Post-training digital learning content

- Object model.
- Typical objects.
- Testing.
- XML.
- Document generation.

Digital activities

This online training course introduces the Python object model and typed objects, one of Python's modern development axes. Participants will then be able to build high-performance, modern applications and secure data processing. They will discover best practices for testing their code and ensuring its quality. They will also learn how to manipulate XML with Python, and generate PDF, openDocument and image documents.

Dates and locations

REMOTE CLASS
2026 : 1 June, 14 Sep., 23 Nov.

METZ
2026 : 1 June, 23 Nov.

PARIS LA DÉFENSE
2026 : 18 May, 7 Sep., 16 Nov.

NANCY
2026 : 14 Sep.