Date de publication : 04/02/2026

Formation : Python, programmation objet

Formation pratique - 5j - 35h00 - Réf. PYE
Prix : 2610 € H.T.

NEW

À l'issue de la formation, le participant sera capable d'utiliser les principales fonctionnalités du langage Python pour développer des applications multiplateformes.

Objectifs pédagogiques

À l'issue de la formation, le participant sera en mesure de :

- Comprendre les bases du langage Python et son écosystème
- Acquérir les principes de la programmation objet
- Comprendre et utiliser les fonctions et modules
- Concevoir des interfaces graphiques
- Utiliser les outils de test et d'évaluation de la qualité d'un programme Python

Public concerné

Développeurs, ingénieurs, chefs de projet proches du développement.

Prérequis

Avoir des connaissances de base en programmation (souhaitable en langage objet).

Méthodes et moyens pédagogiques

Travaux pratiques

Exercices pratiques et/ou études de cas.

Méthodes pédagogiques

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.

PARTICIPANTS

Développeurs, ingénieurs, chefs de projet proches du développement.

PRÉREQUIS

Avoir des connaissances de base en programmation (souhaitable en langage objet).

COMPÉTENCES DU FORMATEUR

Les experts qui animent la formation sont des spécialistes des matières abordées. Ils ont été validés par nos équipes pédagogiques tant sur le plan des connaissances métiers que sur celui de la pédagogie, et ce pour chaque cours qu'ils enseignent. Ils ont au minimum cinq à dix années d'expérience dans leur domaine et occupent ou ont occupé des postes à responsabilité en entreprise.

MODALITÉS D'ÉVALUATION

Le formateur évalue la progression pédagogique du participant tout au long de la formation au moyen de QCM, mises en situation, travaux pratiques…
Le participant complète également un test de positionnement en amont et en aval pour valider les compétences acquises.

Modalités d'évaluation

Le formateur évalue la progression pédagogique du participant tout au long de la formation au moyen de QCM, mises en situation, travaux pratiques…

Le participant complète également un test de positionnement en amont et en aval pour valider les compétences acquises.

Programme de la formation

1 Algorithmique, raisonner avant de concevoir - OPTION digital learning préformation

- Introduction à l'algorithmique.
- Les instructions de base en pseudo-code.

Activités digitales

Cette formation en ligne apprend à raisonner avant de concevoir un programme en découvrant les bases de l'algorithmique. Les instructions fondamentales en pseudo-code seront également abordées.

2 Introduction à Python

- Installation et configuration de l'environnement.
- Types de données fondamentaux.
- Structures de données de base.
- Variables et portée.
- Opérateurs et expressions.
- Style et conventions Python (PEP 8).

Travaux pratiques

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

3 Structures de contrôle et fonctions

- Structures de contrôle avancées.
- Définition et utilisation des fonctions.
- Paramètres et retours multiples.
- Gestion des erreurs et exceptions.
- Modules et packages Python.
- Bonnes pratiques de structuration.

Travaux pratiques

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

4 Approche orientée objet

- Principes fondamentaux de la POO.
- Classes et instances.
- Héritage et composition.
- Polymorphisme et abstraction.
- Interfaces et classes abstraites.
- Design patterns en Python.

Travaux pratiques

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

MOYENS PÉDAGOGIQUES ET TECHNIQUES

• Les moyens pédagogiques et les méthodes d'enseignement utilisés sont principalement : aides audiovisuelles, documentation et support de cours, exercices pratiques d'application et corrigés des exercices pour les formations pratiques, études de cas ou présentation de cas réels pour les séminaires de formation.
• À l'issue de chaque formation ou séminaire, ORSYS fournit aux participants un questionnaire d'évaluation du cours qui est ensuite analysé par nos équipes pédagogiques.
• Une feuille d'émargement par demi-journée de présence est fournie en fin de formation ainsi qu'une attestation de fin de formation si le participant a bien assisté à la totalité de la session.

MODALITÉS ET DÉLAIS D'ACCÈS

L'inscription doit être finalisée 24 heures avant le début de la formation.

ACCESSIBILITÉ AUX PERSONNES HANDICAPÉES

Pour toute question ou besoin relatif à l'accessibilité, vous pouvez joindre notre équipe PSH par e-mail à l'adresse psh-accueil@orsys.fr.

5 Manipulation avancée des objets

- Properties et descripteurs.
- Méthodes magiques.
- Métaclasses et décorateurs.
- Introspection et réflexion.
- Gestion de la mémoire.
- Bonnes pratiques POO.

Travaux pratiques
Properties et descripteurs. Méthodes spéciales. Métaprogrammation.

6 POO Avancée

- Méthodes spéciales Python approfondies.
- Héritage multiple et résolution d'ordre de méthodes (MRO).
- Context managers et protocoles Python.
- Design patterns avancés.
- Mixins et composition.
- Anti-patterns et pièges courants.

Travaux pratiques
Méthodes spéciales avancées. Héritage et composition. Design patterns avancés.

7 Gestion d'erreurs avancée

- Hiérarchie complète des exceptions.
- Création d'exceptions personnalisées.
- Logging avancé et configuration.
- Debugging et troubleshooting.
- Gestion des erreurs asynchrones.
- Bonnes pratiques de robustesse.

Travaux pratiques
Exceptions personnalisées. Logging avancé. Debugging et tests.

8 Utilisation de la StdLib

- Arguments et options en ligne de commande (argparse).
- Expressions régulières (re).
- Manipulation du système de fichiers (os, pathlib).
- Modules importants (datetime, collections, itertools).
- Bases de données avec sqlite3.

Travaux pratiques
Manipulation de fichiers. Expressions régulières. Base de données SQLite.

9 Bibliothèque standard Python

- Modules sys et os pour l'interaction système.
- Manipulation avancée de fichiers avec pathlib.
- Collections spécialisées (collections, heapq).
- Expressions régulières avec re.
- Sérialisation avec pickle et json.
- Concurrence avec threading et multiprocessing.

Travaux pratiques
Manipulation système. Collections avancées. RegEx et sérialisation.

10 Outils et utilitaires CLI

- Arguments et options avec argparse.
- Configuration avec configparser.
- Logging en CLI.
- Interfaces console riches.
- Automation système.
- Distribution d'outils CLI.

Travaux pratiques
Parsing d'arguments. Interface utilisateur. Packaging et distribution.

11 Tests et documentation

- Tests unitaires avec unittest et pytest.
- Test Driven Development (TDD).
- Documentation avec docstrings et type hints.
- Génération de documentation avec Sphinx.
- Mesure de couverture de code.
- Intégration continue.

Travaux pratiques
Tests unitaires. Documentation. Couverture et CI.

12 Qualité du code

- Outils d'analyse statique (pylint, flake8).
- Conventions de style (PEP 8).
- Techniques de refactoring.
- Revue de code et pair programming.
- Mesures de complexité.
- Automatisation des contrôles.

Travaux pratiques
Analyse statique. Refactoring. Revue de code.

13 Interfaces graphiques avec Tkinter

- Introduction à Tkinter et ses alternatives.
- Widgets fondamentaux et layouts.
- Gestion des événements.
- Pattern MVC en GUI.
- Styles et thèmes.
- Bonnes pratiques d'interface.

Travaux pratiques
Fondamentaux Tkinter. Widgets avancés. Architecture MVC.

14 Interface avancée et tests GUI

- Widgets personnalisés.
- Styles et thèmes avancés.
- Animation et effets visuels.
- Tests d'interfaces graphiques.
- Accessibilité.
- Internationalisation.

Travaux pratiques
Widgets custom. Tests GUI. Internationalisation.

15 Interfaçage avec C

- Introduction à ctypes et ses alternatives.
- Types de données C et leur mapping.
- Appel de fonctions C depuis Python.
- Gestion de la mémoire et pointeurs.
- Wrapping de bibliothèques C.
- Performance et debugging.

Travaux pratiques
Introduction à ctypes. Appels de fonctions. Performance et debug.

16 Optimisation avec Cython

- Introduction à Cython et son écosystème.
- Syntaxe et types statiques Cython.
- Compilation et build process.
- Optimisation de code Python existant.
- Intégration avec C/C++.
- Profiling et benchmarking.

Travaux pratiques
Fondamentaux Cython. Optimisation. Integration C/C++

17 Profilage et performance

- Outils de profilage (cProfile, line_profiler).
- Analyse de performance CPU et mémoire.
- Optimisation algorithmique.
- Benchmarking et métriques.
- Memory leaks et garbage collection.
- Monitoring en production.

Travaux pratiques
Profilage CPU. Mémoire et GC. Benchmarking.

18 Meilleures pratiques d'optimisation

- Patterns d'optimisation Python.
- Architectures performantes.
- Monitoring et observability.
- Déploiement optimisé.
- Scalabilité horizontale et verticale.
- Tests de charge et stress.

Travaux pratiques
Architecture performante. Tests de charge. Monitoring production.

19 Lancement du projet de synthèse

- Présentation des sujets de projet.
- Méthodologie de développement.
- Organisation des équipes.
- Planning et milestones.
- Architecture et design.
- Standards de qualité.

Travaux pratiques
Initialisation projet. Architecture. Setup projet.

20 Développement – phase 1

- Mise en place de la structure de base.
- Implémentation des features core.
- Tests unitaires et documentation.
- Git workflow et collaboration.
- Code review et qualité.
- Suivi d'avancement.

Travaux pratiques
Core features. Git workflow. Point d'avancement.

21 Développement – phase 2

- Finalisation des fonctionnalités.
- Tests d'intégration.
- Documentation utilisateur.
- Préparation du déploiement.
- Optimisation et refactoring.
- Préparation de la démo.

Travaux pratiques

Features avancées. Documentation. Préparation démo.

22 Soutenance et bilan

- Présentations des projets.
- Démonstrations techniques.
- Retours d'expérience.
- Évaluation des acquis.
- Discussion perspectives.
- Feedback formation.

Travaux pratiques

Présentations. Revue technique. Perspectives. Clôture de la formation.

23 Python 3, les fondamentaux du langage - OPTION digital learning post-formation

- Introduction.
- Types de données.
- Algorithmique.
- Manipulation de données.

Activités digitales

Cette formation en ligne présente les bases essentielles du langage Python pour apprendre à programmer efficacement. Les participants étudieront la structure d'un programme, les types de données, les fonctions et les notions d'algorithmique, avant d'aborder la manipulation de données et la programmation orientée objet. L'utilisation d'une base de données avec SQLAlchemy et l'application des bonnes pratiques pour développer du code Python propre et maintenable seront également abordées.

24 Python 3, concepts avancés — OPTION digital learning post-formation

- Modèle objet.
- Objets typés.
- Tests.
- XML.
- Génération de document.

Activités digitales

Cette formation en ligne présente le modèle objet de Python et les objets typés, un des axes de développement moderne de Python. Les participants seront alors capables de construire des applications performantes et modernes et de sécuriser le traitement des données. Ils seront amenés à découvrir les meilleures pratiques pour tester leur code et ainsi assurer leur qualité. Ils verront également quelques recettes pour manipuler du XML avec Python, puis pour générer des documents PDF, openDocument ou encore des images.

Dates et lieux

CLASSE À DISTANCE
2026 : 1 juin, 14 sep., 23 nov.

LILLE
2026 : 1 juin, 14 sep., 23 nov.

PARIS LA DÉFENSE
2026 : 18 mai, 7 sep., 16 nov.