

# Formation Python avancé

**Objectifs :** Découvrir les concepts avancés du langage Python  
Interfacer Python avec d'autres langages  
Python et le génie logiciel

**Compétences visées :** - Maîtriser la programmation orientée objet en Python, y compris les types, les classes et la gestion des exceptions pour aborder des concepts avancés comme la méta-programmation.  
- Développer et structurer des packages Python en respectant les meilleures pratiques avec la gestion de l'installation et de la distribution.  
- Garantir la qualité du code Python en respectant les normes de codage, en écrivant des tests unitaires et en utilisant des outils d'analyse pour améliorer la couverture de test et l'optimisation des performances.  
- Optimiser les performances des scripts Python, en utilisant des techniques de réduction de complexité, de multithreading et d'analyse de la mémoire.  
- Créer des extensions Python en les intégrant avec des langages comme C ou C++.

**Durée :** 3 jour(s) (21 heures)

**Public :** Tout développeur ou scientifique ayant une expérience du langage Python

**Pré-requis :** Pour suivre ce stage dans de bonnes conditions, il est recommandé d'avoir suivi en amont la formation [Python - Bases et introduction aux bibliothèques scientifiques](#)

**Méthode pédagogique :** Chaque chapitre s'achève par des travaux pratiques qui mettent en oeuvre les éléments présentés. Les TP utilisent les outils Pycharm ou Spyder selon les souhaits  
Pédagogie active mêlant exposés, exercices et applications pratiques dans le logiciel Python.

**Modalités d'évaluation :** Un formulaire d'auto-évaluation proposé en amont de la formation nous permettra d'évaluer votre niveau et de recueillir vos attentes. Ce même formulaire soumis en aval de la formation fournira une appréciation de votre progression.

Des exercices pratiques seront proposés à la fin de chaque séquence pédagogique pour l'évaluation des acquis.

En fin de formation, vous serez amené(e) à renseigner un questionnaire d'évaluation à chaud.

Une attestation de formation vous sera adressée à l'issue de la session.

Trois mois après votre formation, vous recevrez par email un formulaire d'évaluation à froid sur l'utilisation des acquis de la formation.

**Accessibilité :** Vous souhaitez suivre notre formation Formation par ville et êtes en situation de handicap ? Merci de nous contacter afin que nous puissions envisager les adaptations nécessaires et vous garantir de bonnes conditions d'apprentissage

**Tarifs :**

- Présentiel : 1650 € HT

- Distanciel : 1500 € HT

(-10% pour 2 inscrits, -20% dès 3 inscrits)

**Option(s) :**

- Forfait déjeuners : 60 € HT

## Nos prochaines sessions

### Distance

du 3 au 5 février 2025

du 17 au 19 novembre 2025

### Lyon

du 9 au 11 décembre 2024

du 12 au 14 mai 2025

du 8 au 10 décembre 2025

### Paris

du 2 au 4 décembre 2024

du 29 au 31 janvier 2025

du 16 au 18 juin 2025

du 1 au 3 décembre 2025

### Toulouse

du 24 au 26 mars 2025

du 29 sept. au 1 oct. 2025

## Programme :

### - Rappels de Programmation Orientée Objet

- Types de base
- Création de classes
  - Héritage, Polymorphisme...
- Traitement des Exceptions
  - raise, try, except, finally
- Le "Data-Model" et les fonctions "magiques"
- Importations "avancées" - utilisation de . et ..

### - Syntaxe avancée

- Listes en "compréhension"
- Itérateurs et générateurs
- Modules itertools, collections
- Lambda fonctions
- Décorateurs
- Instructions with et Contextlib

- Instruction yield
- Programmation asynchrone
- Coroutines

## - Classes avancées

- Sous-classer les types de base
- Résolution des héritages multiples
- Cas de la méthode "super"
- Descripteurs `__get__` et `__set__`
- Propriétés (properties)
- `dict__` et `__slots__`
- Classes abstraites
- Méta-programmation

## - Introduction à l'écriture de packages

- "Meilleures pratiques"
- `setup.py` et scripts de contrôle
- L'utilitaire pip
- Installer un package
- Désinstaller un package
- Enregistrer et uploader un package

## - Qualité logicielle

- Annotations
- Respect de la PEP 8 - normes de codage
- Tests unitaires (doctest et unittest)
- Taux de couverture - coverage

## - Solutions d'optimisation

- Réduction de la complexité
- Bytecode et le module "dis"
- Multithreading
- Multiprocessing
- Gestion des caches

- Profiling
- Analyse de l'occupation mémoire

## **- Interfaçage avec C / C++**

- Objectif et principe
- SWIG
- Cython
- Le module ctypes

*Date de dernière modification : 5 novembre 2024*