# Titre

Python avancé

# Contenu validé

Oui

# Catch phrase (2 lignes max)

Pendant cette formation centrée sur le pragmatisme, vous découvrirez comment fonctionnent vraiment les fonctions, le modèle objet et l’écosystème de bibliothèque.

# Description (10 lignes max)

Vous avez une bonne maîtrise de Python, mais certaines parties restent un peu mystérieuses ? C’est tout à fait normal, ce langage, bien que très facile d’accès n’en est pour autant pas simpliste et utilise des mécanismes parfois complexes.

Cette formation vise d’une part à vous donner une compréhension des mécanismes subtils qui font la force de Python (les opérations paresseuses et les générateurs, les fonctions avancées et les décorateurs, l’orienté objet et l’accès aux attributs). D’autre part, seront abordées les bonnes pratiques de développement (formatage du code, tests et mock, isoler et rendre reproductibles les projets...).

# Identifiant technique

pyavance

# Domaine

développement

# Sous-domaine

Python

# Url

/formations/developpement/python-avance

# Ordre dans la page

3

# Prochaines sessions

## Lyon

21-22 juin 2021

## Lille

14-15 juin 2021

## Paris

28-29 juin 2021

# Public visé / participants

* Développeurs
* Ingénieurs
* chercheurs ayant une bonne connaissance de Python.

# Prérequis

* Connaitre et maitriser la syntaxe de base de python, la création de fonctions ainsi que les notions de programmation orientée objet (attribut, classes…).
* Être à l’aise avec la ligne de commande.
* Une connaissance des bonnes pratiques en développement (gestion de version, configuration d’éditeurs de texte) est un plus pour certaines parties

# Durée

4j soit 28h

# Équilibre théorie / pratique

35% théorie / 65% pratique

# Objectifs pédagogiques

* Utiliser les techniques avancées du langage Python : “Context Manager”, “Closures”, Decorateurs, fonctions avancées
* Identifier les patrons de conceptions (Design Patterns) à utiliser dans des cas spécifiques
* Analyser les performances d’un programme python pour en identifier les goulots d’étranglement à optimiser
* Packager et déployer ses bibliothèques
* Identifier et utiliser les bibliothèques incontournables du langage pour réaliser des développements complexes rapidement

# Tarifs

1650 € HT (pauses et repas inclus)

# Plan de formation / programme

## **Le langage Python**

### Présentation / historique et contexte / philosophie

### Cas d’utilisation et de non utilisation

### Présentation des outils de développement

TP : installation d’un interpréteur Python, d’un éditeur de texte et familiarisation avec les outils

## **Rappels sur Python**

### Formatage et affichage des variables

### La gestion d’erreur et la compréhension des exceptions

### Fonctions.

* Portée des variables et la règle LEGB
* Les lambda expression
* Les générateurs

### La structuration du code en modules.

### L'utilisation des fichiers et encodage de caractères.

TP : exercices permettant d’illustrer la spécificité d’un code Pythonique

## **Python avancé**

### Usages avancés des fonctions

* Fonctions d’ordre supérieur
* Présentation des décorateurs et de leurs usages classiques
* Décorateurs de la bibliothèque standard

### Les classes de la bibliothèque collections

### Les méthodes magiques des classes

### Générateurs, coroutines et le mot clé yield

### Gestionnaires de contextes (context manager), définition et cas d’usages

* + Sous forme de classe
  + Sous forme de fonctions

## **Qualité de code (QA)**

### Les environnements virtuels pour gérer facilement les projets

* Le concept et les cas d’usages des environnements virtuels
* Installation de bibliothèques tierces avec pip
* Création et évolution d’environnements virtuels

### Les outils d’analyse statique et l’annotation de type

* Présentation du concept
* Présentation des outils (pylint, mypy)

### Le formatage

* Présentation de la PEP 8
* Utilisation d’autoformatter (black, autopep8)

### La gestion de la documentation

* Écriture de la documentation (docstring, formattage)
* Les outils d’extraction de documentation

### Les tests unitaires

* Le concept de tests unitaires
* Les outils de tests unitaires (unittest, pytest) en python, notion de couverture de tests et génération de rapports
* Présentation du concept de développement piloté par les tests (TDD)
* L'automatisation des tests, l'agrégation de tests.
* Intégration dans un pipeline d’intégration continue (CI)

### Débogage

* Le concept de déboguer du code
* Le débogueur textuel de python
* Les débogueurs graphiques

**TP : Configurer son éditeur de code pour intégrer des outils de QA, ajout de tests unitaires sur des codes précédents**

## **Programmation orientée objet avancée**

### Rappel des spécificités des objets en Python

### Définition et utilisation des différents types de méthodes en Python

* Les méthodes classiques
* Les méthodes de classe
* Les méthodes statiques

### Le “duck typing” et l’interprétation

### Spécificité de l’accès aux attributs en Python

* Les méthodes magiques \_\_getattribute\_\_ et \_\_getattr\_\_
* Les “property” pour l’encapsulation

### L’héritage multiple et ses cas d’utilisation

### Les patrons de conception (design patterns)

* Le concept de design patterns et leur utilité
* Présentation de quelques design patterns connus et leurs cas d’utilisation classique

TP : exercices sur la modélisation pythonique des objets en utilisant les différents mécanismes vus en classe. Utilisation de designs patterns pour organiser un code

## **Améliorer les performances de codes Python**

### Mesurer les performances

* Analyser les temps de réponses(« timeit » & « cprofile », techniques de visualisations)

### Examiner la consommation mémoire

### Initiation aux traitements répartis

* Utilisation du multiprocessing
* Présentation du concept de “Map Reduce”
* Présentation du module Dask

TP : identification des goulots d’étranglement et optimisation des performances d’un programme simple

## **Utilisation de bibliothèques classiques de l’écosystème**

### Découvrir les points importants pour choisir une bibliothèque

### Installer correctement une bibliothèque Python

### Étudier et parcourir la documentation d’une bibliothèque pour implémenter nos besoins

TP : Mise en pratique sur un cas d’analyse de données (écosystème scientifique) ou un cas intéressant les stagiaires