Sparks.

Programme de formation
Python Perfectionnement

Objectifs pédagogiques

Cette formation Python Perfectionnement vous permettra de consolider et d'étoffer vos connaissances sur le langage de programmation. Vous apprendrez à utiliser les design patterns, mettrez en œuvre les meilleures pratiques de développement, et saurez exploiter les fonctionnalités avancées du langage pour développer des applications stables et robustes. Sur simple demande, cette formation avancée sur Python pourra être complétée ou adaptée via des modules spécifiques, conformément aux besoins des apprenants : programmation graphique, développement Web avec Django, introduction à Python pour la Data Science, focus sur une librairie…

- Implémenter des design patterns en Python
- Connaître et mettre en œuvre les meilleures pratiques de développement
- Utiliser les fonctionnalités avancées du langage
- Packager et déployer des librairies
- Manipuler des ensembles de données avec Python
- S'initier à la programmation réseau
- Mesurer et améliorer les performances de ses applications

Prérequis

Avoir suivi la formation Python ou posséder des connaissances équivalentes.

Durée

3 jours

Public

Développeurs, architectes techniques, administrateurs, ingénieurs DevOps

Programme de formation

Introduction à la formation Python Perfectionnement : rappels et notions avancées
Présentation générale de la formation Python avancé
Installation de Python, configuration d'un environnement de développement
Fonctions avancées, passage d'arguments
Les décorateurs
La fermeture (closure)
Les design patterns (types, application, recherche)

Exemples de cas pratiques : création d'un décorateur, suivre un design pattern (Factory, Singleton, …).

Programmation Orientée Objet avancée avec Python
L'introspection
Les propriétés
Les itérateurs et générateurs
Les classes abstraites
Les métaclasses
Les gestionnaires de contexte pour gérer les ressources

Exemples de cas pratiques : utilisation d'une classe et d'une méthode abstraite, implémentation d'une métaclasse.

Gestion des librairies
Le Python Package Index (PyPI)
Installer des librairies avec pip
Gestion des versions
Créer un environnement avec virtualenv
Construire un package de distribution avec setuptools

Exemple de cas pratique : portage de code entre différentes versions de Python, création d'environnements isolés avec virtualenv, packaging de librairies.

Traitement de données en Python
Les concepts de sérialisation et de désérialisation
Utiliser ElementTree pour extraire des données d'un fichier XML
Principes de base pour l'exploitation de bases de données
Accès à SQLite ou MySQL

Exemples de cas pratiques : recherche d'informations dans un fichier XML, création d'un programme Python pour accéder à une base MySQL, mise au point de requêtes.

Sockets et programmation réseau
Concepts réseaux
Introduction au module Socket, aux objets sockets et leurs méthodes
Les sockets en mode connecté (TCP), non connecté (UDP)

Exemple de cas pratiques : création d'un serveur/client socket

Amélioration des performances
Mesurer les performances des applications : timeit, cProfile et pstats
Bien choisir les structures de données
Les compréhensions de liste
Parallélisation : multithreading vs multiprocessing

Exemples de cas pratiques : mesure du temps d'exécution d'une fonction, profiling d'un programme, optimisation d'un code grâce à la compréhension de liste.