Sparks.

Programme de formation
Python initiation et avancé

• Objectifs

- Connaître les concepts du langage Python ainsi que les principales librairies scientifiques :- NumPy, Pandas, Matplotlib...- Découvrir les concepts avancés du langage Python- Interfacer Python avec d'autres langages- Python et le génie logiciel

• Pré requis

Connaissance des bases de la programmation.

• Durée

4 jours

• Public

Developpeurs

• Plan de formation

Introduction rapide
Historique
Installation
Premier programme
Principales versions
Prise en main de IPython

Bases du langage
Opérateurs et expressions
Instructions de contrôle
Fonctions
Structures de données

Modules et packages
Programmation modulaire
Importation de modules
Du module au Package

Librairie standard
Gestion des fichiers
Gestion des répertoires
Interface avec le système

Programmation objets (bases)
Définition de classes
Héritages
Gestion des exceptions
Surcharge des opérateurs

Syntaxe avancée
Définitions fonctionnelles de listes
Itérateurs et générateurs
Décorateurs
Instructions « with » et Contextlib
Lambda fonctions

Aide au développement
Documentation de code
Tests unitaires
Debugger
Installation de packages (pip)

NumPy
Base de NumPy (tableaux et types)
Entrées/Sorties
Fonctions utiles
- corrélation de données
- polynômes
- programmation fonctionnelle
Manipulation de matrices

Matplotlib
Structure d'un graphe - éléments esthétiques
Layout et Annotations
Graphes en 3D
Graphes interactifs
Introduction à Seaborn
Introduction à VisPy (3D temps-réel)

Pandas
Manipulation de Series et DataFrames
Indexation, Catégories

Fonctions numériques et statistiques
Lecture & écriture de données
Transformation de données
Agrégations
Time-Series
Visualisation

Présentation de Scikit-image

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
Le module ctypes