**Prérequis :**

Cette formation s’adresse à des développeur Python ayant la maîtrise des fondamentaux de ce langage et une pratique quasi quotidienne. Débutant s’abstenir

**Objectifs :**

En s’appuyant sur les bibliothèques interne de Python :

* vous gagner du temps en développement en évitant l’écriture de fonction fastidieuses
* vous limitez les risques de régressions lors de vos migrations de versions, de vos phases de maintenance
* Acquérir les concepts de la programmation parallèle
* Sachez identifier les portions d’un programme qui sont parallélisables
* Posséder une vision claire de l’écosystème de calcul parallèle pour Python
* Développer des applications parallélisées (programmation asynchrone, multithreading, multiprocessing, distribuées sur plusieurs machines)

**Présentation**

Plonger au coeur de Python à travers ces 3 programmes :

Programme 1 : Python avancé – Programmation « pythonique »

Découvrez ces bibliothèques internes de Python qui vous permettront de gagner un temps précieux de mise en œuvre et apporterons une robustesse à vos applications.

Programme 2 : Python avancé – Performance et qualité

Appropriez-vous les outils internes pour rechercher des anomalies, localiser les goulots d’étranglements.   
Apprenez et mettez en œuvre des techniques de l’écosystème Python pour optimiser, accélérer et répartir vos traitements.

Superviser vos programmes à l'aide du monitoring

Programme 3 : Python avancé – Tester vos scripts

Mettez rapidement en place des tests unitaires simples, des tests d’intégration ou des tests de charges en s’appuyant sur des bibliothèques internes de Python et les bibliothèques externes « pytest »,« pymock » et « faker »

**Méthodes pédagogiques :**

Répartition cours / Travaux pratiques : 25 % / 75 %

Programme n°1   sur 3 jours – 21 heures:

Programme 1, jour 1 - les bibliothèques d’outils de base

Les fichiers de configuration

La bibliothèque « configparser »

La bibliothèque « json »

Le traitement des fichiers YAML

Les traces en Python

La fonction « print »

La bibliothèque « logging »

Intérêts et principes

Mise en oeuvre d’un premier exemple  
 par programmation

par fichier de configuration

Modifier les filtres

Créer son propose composant de trace

Point sur les traces en multi-threading, en multi-processing

Les arguments en ligne de commande

La bibliothèque « argparse »

La variable « sys.argv »

Exploration et mise en oeuvre  
 La bibliothèque externe « click »

Programme 1, jour 2 – Les structures de données avancées et leurs outils

Rappel sur les types en Python

Les types avancés

« Containers », « Collections », « Sequences », « Mapping »

Propriétés des types

mutable, subscriptable, hashable, trucmuchable, ...

Introduction aux annotations

La bibliothèque « collections »

Cas d’usage pour « defaultdict », « Counter », « deque », « namedtuple »

Mise en œuvre de ces classes

Cas d’usage pour « UserList »

Hériter de cette classe

Les fonctions de manipulation des collections

fonctions « map »/ »filter » vs générateurs à la volée

La bibliothèque « itertools »

Les bibliothèques « heapq » et « bisect »

Intérêts

Exemple de mise en œuvre

Programme 1, jour 3 – Techniques avancées

Les décorateurs (ou fonction englobante)

La fonction est un objet (comme les autres)

Contexte de mise en œuvre

Les closures

Variables « local », « non local », « global »

Attribut de fonction

La variable \_\_closure\_\_

Cas d’usage des décorateurs

Mise en œuvre

à partir d’une fonction

à partir d’une classe

Paramètres de décorateurs

Etudes des décorateurs de la bibliothèques « functools »

@lru\_cache

@wraps

Les « Context Managers »

Intégration sur une classe avec les « dunder » méthodes spécifiques

Intégration sur une fonction avec la biblothèque « contextlib »

Encore des fonctions

Les fonctions lambdas  
 Les fonctions de la bibliothèque « functools » : «reduce », « partial », etc ..

les fonctions internes « all », « any »

Et encore

L’opérateur splat (ou \*)

L’opérateur logique « or » lors des affectations

Programme 2   sur 3 jours – 21 heures:

Choisir les bonnes approches multi-programmation parmi l’écosystème Python. Utiliser les outils de mesures de performance, trouver les goulots d’étranglements, établir un arbre d’appels de fonctions

Programme 2, jour 1 - les bibliothèques d’outils

Les fichiers de configuration

La bibliothèque « configparser »

La bibliothèque « json »

Le traitement des fichiers YAML

Les traces en Python

La bibliothèque « logging »

Intérêts et principes

Mise en oeuvre d’un premier exemple  
 par programmation

par fichier de configuration

Modifier les filtres

Créer son propose composant de trace

les arguments en ligne de commande

La variable « argv » de la bibliothèque « sys.argv »

La bibliothèque « argparse »

Exploration et mise en oeuvre  
 La bibliothèque externe « click »

Programme 2, jour 2 – Mesurer les performances, principe de programmation multi-tâche, et première mise en ouvre

Rappel sur la gestion des erreurs

Concept et mise en oeuvre des exceptions

Mesurer les performances

Analyser les temps de réponses

Les bibliothèques « timeit » & « profile »

Examiner la consommation mémoire

Le garbage collector, la bibliothèque externe « memory\_profiler »

Explorer l’arbre d’appel de fonctions d’un script

Le module « trace » de la bibliothèque « sys »

La bibliothèque externe «

La programmation multi-taches : la bibliothèque « threading »

Rappel sur la gestion mémoire

La mémoire partagée

Principe

Contrôler son accès : Verrou, Sémaphore  
 Les taches élémentaires, les évènements

Cas d’usages et limitations

Les traces en multi-taches

Programme 2, jour 3 – Multi-processus, Taches asynchrones et Traitements répartis

La programmation multi-processus : la bibliothèque « multiprocessing »

Mémoire partagées

Cas d’usages et limitations

Les traces en multi-processus

La programmation asynchrone : la bibliothèque « asyncio »

Principe et cas d’usage

Les « futures »

Le lancement des futures

La boucle locale

Les traitements répartis

Mise en œuvre en python natif :

la bibliothèque « multiprocessing »

la bibliothèque « xmlRPC »

Autres mises en œuvre existantes

La bibliothèque externe « RPyc»

La bibliothèque externe « ray »

La bibliothèque externe « Celery »

La bibliothèque externe « Dask »

Programme 3

En cours …..