DataTips

Formation

PYTHON

DataTipsLearning.com

Tarif :
1500 € HT

Durée :
3 Jours
(21h)

Lieu :
PARIS/
Distancielle/
Votre Local

Contact :

contact@datatipslearning.com
+33 0751386021
Réf. DTPSL-PYTHON-2022
S'inscrire

Plus de détails :

Objectifs
Public / Prérequis
Programme détaillé
Méthode pédagogique

Python est un langage de programmation à usage général. Il a été créé en 1993 par Guido van Rossum. Il est aujourd'hui le langage le plus utilisé au monde. Il a été conçu et développé pour écrire des logiciels pour une grande variété de disciplines.

Python a été utilisé pour écrire des applications pour résoudre des problèmes en biologie, en chimie, en analyse financière, en analyse numérique, en robotique et dans de nombreux autres domaines. Il est également largement utilisé comme langage de script à l'usage des administrateurs informatiques, et de traitement de données et de dataviz.

Objectifs

- Présenter le langage Python.
- Apprendre les bases de Python.
- Comprendre le cycle du développement Python.
- Apprendre à modulariser un projet.
- Apprendre la programmation orientée objet.
- Présenter les librairies Python et leur utilisation.
- Apprendre le packaging et la distribution du code Python.
- Contrôler les performances des applications.
- Apprendre à traiter des données avec Pandas.
- Le multitraitement, multithreading et la programmation asynchrone.
- Comprendre la programmation fonctionnelle.
- Apprendre le test et l'automatisation avec Python.

Public

- Data Engineer
- Data Architectes
- Data Scientist
- Data Analyst

Prérequis

- Pas de connaissances particulières.

Programme détaillé

I. Introduction à Python
   1. Présentation de Python
   2. Installation de Python
   3. IDLE et le Shell Python
   4. Création, enregistrement et exécution d'un fichier Python
   5. IDLE sur plusieurs plates-formes

II. Les bases de python
   1. Variables et affectation

2. Fonctions intégrées
3. Fonctions User-Defined
4. Instructions if, else et elif
5. Boucles
6. Strings
7. Entrée/sortie de fichier
8. Enumérations
9. Texte Unicode vs Bytes

III. Environnements de développement Python modernes
1. L'écosystème de packaging de Python
2. Isolation de l'environnement d'exécution
3. Isolation de l'environnement au niveau de l'application
4. Isolation de l'environnement au niveau du système
5. Outils de productivité populaires

IV. Cycle de vie optimal du développement Python
1. Culture et communauté Python
2. Différentes phases d'un projet Python
3. Stratégies du processus de développement
4. Documentation du code Python
5. Développer un schéma de nommage efficace
6. Explorer les choix pour le contrôle de code source
7. Comprendre les stratégies de déploiement du code
8. Environnements de développement Python

V. Utilisation de la modularisation pour gérer des projets complexes
1. Présentation des modules et packages
2. Importation de modules
3. Charger et initialiser un module
4. Ecrire des modules réutilisables
5. Création d'un package
6. Accéder aux packages depuis n'importe quel endroit
7. Partager un package

VI. Programmation Python orientée objet avancée
1. Présentation des classes et des objets
2. Comprendre les principes de la POO
3. Polymorphisme
4. Utilisation de la composition comme approche de conception
5. Présentation du typage duck-typing en Python
6. Apprendre quand ne pas utiliser la POO en Python
7. Interfaces
8. Inversion de contrôle et injection de dépendances

VII. Les librairies Python pour la programmation avancée
1. Présentation des collections de données Python
2. Utiliser des itérateurs et des générateurs pour le traitement des

données
3. Manipuler des fichiers en Python
4. Gestion des erreurs et des exceptions
5. Utilisation du module de logging Python

VIII. Packaging et distribution de code Python
1. Packaging et distribution des libraires
2. Applications et services de packaging pour le Web
3. Création d'exécutable standalone

IX. Observation du comportement et des performances des applications
1. Capture des erreurs et des logs
2. Instrumenter le code avec des métriques personnalisées
3. Suivi d'application distribué

X. Traiter des données avec python
1. Fonctions avancées
2. Concepts avancés avec des structures de données
3. Le DataFrame Pandas
4. Utilisation de NumPy
5. Atteindre des performances optimales avec numexpr
6. Données étiquetées hautes performances avec xarray

XI. Multitraitement, multithreading et programmation asynchrone
1. Comprendre le multithreading en Python et ses limites
2. Implémenter le multitraitement
3. Utilisation de la programmation asynchrone pour les systèmes réactifs

XII. Programmation fonctionnelle
1. Le map, filter, et reduce
2. Objets partiels et fonctions partielles
3. Générateurs
4. Expressions génératrices
5. Décorateurs
6. Traiter une fonction comme un objet
7. Fonctions d'ordre supérieur
8. Fonctions anonymes
9. Les packages pour la programmation fonctionnelle
10. Closures

XIII. Programmation événementielle (Event-Driven)
1. Présentation de la programming événementielle
2. Différents styles de programmation événementielle
3. Architectures événementielles

XIV. Métaprogrammation
1. Présentation de la métaprogrammation

2. Utilisation de décorateurs pour modifier le comportement de la fonction avant utilisation
3. Intercepter le processus de création d'instance de classe
4. Métaclasses
5. Génération de codes

XV. Programmation Python pour le Cloud
1. Les options cloud pour les applications Python
2. Création de services Web Python pour le déploiement dans le cloud
3. Utilisation du cloud pour le traitement des données

XVI. Création d'une fonction serverless avec Python
1. Présentation des fonctions serverless
2. Présentation des options de déploiement pour les fonctions serverless
3. Créer des fonctions serverless

XVII. Optimisation de codes
1. Les causes communes des mauvaises performances
2. Profilage de code
3. Réduire la complexité avec les bonnes structures de données
4. Les compromis architecturaux

XVIII. Test et automatisation avec Python
1. Les différents niveaux de test
2. Les frameworks de test Python
3. Exécution de TDD
4. Présentation du CI automatisé
5. Les principes du développement test-driven
6. Ecrire des tests avec pytest
7. Quality automation
8. Test de mutation
9. Utilitaires de test utiles

Méthode pédagogique

La formation se compose d'une partie théorique, et également une partie pratique représentant 60% de de la formation.
La partie pratique contient plusieurs exercices sous forme de notebook Databricks avec les corrections, avec aussi un projet à la fin de la formation comme simulation d'une prod.

Chaque jour, une évaluation rapide des connaissances est effectuée avant de commencer les nouvelles parties de la formation.

A la fin, une synthèse globale est délivré aux stagiaires, renforcé par un projet prod.
Finalement, une évaluation QCM est proposée.

Un support de cours sera remis à chaque stagiaire comprenant les slides, les exercices et les corrigés et un git du projet prod.
Une feuille de présence est fournie en fin de formation avec une certificat de complétion de formation pour chaque stagiaire.

Le formateur est un Data Engineer expert, qui intervient sur le sujet depuis plusieurs années en formation mais aussi en conseil.