Smartfuture
Certifications | Training | Technology solutions

FORMATION & CERTIFICATION • Programmeur professionnel Python 1

Programmeur professionnel Python 1

Formation certifiante en présentiel et en ligne à la demande. Développez des compétences professionnelles en programmation Python : structures de données, fonctions, programmation orientée objet, gestion des fichiers, tests et bonnes pratiques de développement.

Distributeur officiel Certiport | Centre d'examen Pearson VUE | Formateurs certifiés Microsoft

5 jours 35 h	**Examen** Programmeur professionnel Python 1
Modalité Présentiel Distanciel	**Niveau** Intermédiaire à avancé

INSCRIPTION / RÉSERVATION

Je m'inscris maintenant

OBJECTIFS PÉDAGOGIQUES

- Bases Python et bonnes pratiques (PEP 8, structuration).
- Structures de données et programmation modulaire.
- Programmation orientée objet, fichiers et gestion d'erreurs.
- Bibliothèques standards et préparation à la certification Python 1.

PUBLIC CIBLE

- Développeurs débutants ou intermédiaires avec des bases en Python.
- Étudiants en informatique ou disciplines techniques cherchant à professionnaliser leur pratique.
- Ingénieurs, analystes, data engineers ou administrateurs souhaitant automatiser des tâches.
- Toute personne visant la certification professionnelle Python.

PRÉREQUIS

- Concepts de base : variables, conditions, boucles.
- Première pratique Python : scripts simples, listes.
- Notions générales d'algorithmique et logique.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Rappel des fondamentaux du langage Python

Environnement de développement

- Installation de Python, configuration de l'environnement (IDE, éditeurs, interpréteur).
- Scripts vs mode interactif, organisation d'un projet Python.
- Utilisation de pip et des environnements virtuels (venv).

Syntaxe et structures de base

- Variables, types de base (int, float, bool, str), conversions de type.
- Opérateurs arithmétiques, logiques et de comparaison.
- Structures de contrôle : if/elif/else, boucles for et while.
- Entrées/sorties simples et formatage de chaînes (f-strings).

Style de code et bonnes pratiques

- PEP 8 : conventions de nommage, indentation, lisibilité.
- Commentaires, docstrings et documentation du code.

Structures de données et collections avancées

Listes, tuples et sets

- Création, accès, modification et suppression d'éléments.
- Parcours, tri, recherche et fonctions intégrées.
- Notion d'immuabilité (tuples) et d'unicité (sets).

Dictionnaires et collections imbriquées

- Clés, valeurs, méthodes essentielles des dictionnaires.
- Listes de dictionnaires, dictionnaires de listes, structures imbriquées.

Compréhensions & itérations efficaces

- List comprehensions et dict/set comprehensions.
- Filtrage, transformations et expressions conditionnelles.
- Utilisation de enumerate, zip et autres fonctions utilitaires.

Fonctions, modules et packages

Fonctions en Python

- Définition et appel de fonctions, paramètres et valeurs de retour.
- Arguments positionnels, nommés, valeurs par défaut.
- *args et **kwargs, fonctions anonymes (lambda).
- Portée des variables et espaces de noms (local, global, nonlocal).

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Modularisation du code

- Création et import de modules personnalisés.
- Organisation d'un projet en packages (fichiers __init__.py).
- Gestion des imports relatifs et absolus.

Bibliothèque standard

- Exploration de modules courants : math, random, datetime, os, sys.
- Utilisation de pathlib pour la gestion avancée des chemins de fichiers.

Programmation orientée objet (POO) avec Python

Concepts de base de la POO

- Classes, objets, attributs et méthodes.
- Constructeur (__init__), représentation d'objet (__str__, __repr__).

Encapsulation, héritage et polymorphisme

- Attributs de classe vs attributs d'instance.
- Héritage simple, surcharge de méthodes.
- Notion de polymorphisme et classes abstraites (abc).

POO avancée et bonnes pratiques

- Propriétés (property), getters/setters.
- Introduction aux dataclasses.
- Utilisation d'objets pour structurer une application Python.

Fichiers, entrées/sorties et gestion des exceptions

Manipulation des fichiers

- Ouverture, lecture et écriture de fichiers texte.
- Gestion des fichiers binaires, encodage (UTF-8, etc.).
- Lecture et écriture de fichiers CSV et JSON.

Gestion des exceptions

- Principes des exceptions en Python.
- Bloc try/except/else/finally, exceptions intégrées les plus fréquentes.
- Création d'exceptions personnalisées.

Context managers

- Utilisation de with pour sécuriser l'accès aux ressources.
- Création de context managers personnalisés (protocoles __enter__ / __exit__).

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Bibliothèques, tests et outils de développement

Bibliothèques utiles au développeur Python

- Collections, itertools, functools : travailler efficacement avec les données.
- Module logging pour le suivi des applications.
- Notions de sérialisation (pickle, json).

Qualité du code et tests

- Introduction aux tests unitaires : principes de TDD.
- Frameworks de test (unittest, présentation de pytest).
- Écriture et exécution de tests, interprétation des résultats.

Débogage et optimisation

- Utilisation d'outils de débogage (print, logging, pdb, IDE).
- Notions de performance et de complexité (vue introductive).

Accès aux données et intégration

Accès à une base de données simple

- Introduction à sqlite3 pour des bases de données locales.
- Connexion, création de tables, requêtes simples, lecture et écriture de données.

Consommation de services web

- Requêtes HTTP avec la bibliothèque requests (niveau sensibilisation).
- Récupération et traitement de données JSON issues d'API.

Atelier de synthèse & préparation à la certification

Mini-projet de développement

- Conception d'une petite application Python (gestion, outil d'analyse, script d'automatisation…).
- Découpage en modules, choix des structures de données et de la POO.
- Gestion des erreurs, logs et fichiers.

Entraînement type examen

- Rappels des points clés du syllabus de certification.
- QCM, exercices pratiques et correction collective.

Plan d'action individuel

- Identifier ses axes de progression en Python.
- Conseils pour continuer à pratiquer et maintenir son niveau.