PYTHON & DATA : API, INTELLIGENCE ARTIFICIELLE

Python : POO et traitement de données

Passez à un Python professionnel : classes et dataclasses, héritage et protocoles, décorateurs et générateurs, fichiers CSV, JSON et XML, journalisation, tests pytest et projet installable.

DURÉE	TARIF	NIVEAU	LANGUE	GROUPE	FORMAT
3j (21h)	**1 275 € / participant** Inter-entreprise. Intra : sur devis, au plus 1 275 € / participant	**Intermédiaire**	**Français**	**3-10**	**Formation** Présentiel ou à distance

1 PUBLIC VISÉ

- Développeurs Python qui connaissent les bases du langage et veulent écrire un code structuré et professionnel
- Analystes et ingénieurs qui veulent automatiser le traitement de fichiers de données avec Python

2 PRÉREQUIS

- Écrire un script Python avec des fonctions, des listes, des dictionnaires, des boucles et la gestion des exceptions
- Aucune connaissance de la programmation orientée objet n’est requise

3 OBJECTIFS PÉDAGOGIQUES

- Écrire des classes Python avec attributs, propriétés, méthodes de classe et méthodes statiques, et des dataclasses pour les données
- Construire une hiérarchie avec héritage, classes abstraites et protocoles, et personnaliser les objets avec les méthodes spéciales (affichage, comparaison, itération)
- Écrire des décorateurs, des générateurs et des gestionnaires de contexte
- Lire, transformer et écrire des fichiers CSV, JSON et XML avec pathlib et datetime, en journalisant les opérations avec logging
- Tester ses classes avec pytest : assertions, fixtures et paramétrage
- Structurer un projet Python installable : pyproject.toml, environnement virtuel, annotations de type vérifiées par mypy, formatage et analyse avec Ruff

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) : Programmation orientée objet en Python

Module 1 : Classes, propriétés et dataclasses (3h30)

Classes et objets (1h30)

- Pourquoi la POO : organiser, réutiliser, maintenir
- class, __init__ et self ; attributs d’instance et de classe
- Méthodes d’instance, @classmethod et @staticmethod

- Exercice d’entraînement : classe Produit avec prix, stock et méthodes métier

Propriétés et dataclasses (2h)

- @property : lire et modifier un attribut avec contrôle
- Les dataclasses : @dataclass, valeurs par défaut, frozen, __post_init__
- Annoter ses classes et ses fonctions avec les types
- TP évalué : classes Produit et Panier avec des propriétés qui valident les données, une dataclass pour les lignes de panier et le calcul du total

Module 2 : Héritage, protocoles et méthodes spéciales (3h30)

Héritage et classes abstraites (1h30)

- super() et redéfinition de méthode
- Héritage multiple et ordre de résolution (MRO)
- Classes abstraites : abc.ABC et @abstractmethod
- Exercice d’entraînement : hiérarchie de comptes bancaires

Polymorphisme, protocoles et méthodes spéciales (2h)

- Le duck typing ; typing.Protocol ; isinstance
- Méthodes spéciales : __str__, __repr__, __eq__, __lt__, __len__, __iter__
- TP évalué : sérialisation polymorphe en plusieurs formats derrière un protocole, et classe Panier affichable, comparable et itérable

Jour 2 (7h) : Python expressif et fichiers de données

Module 3 : Décorateurs, générateurs et gestionnaires de contexte (3h30)

Décorateurs (1h30)

- Les fonctions sont des objets
- Écrire un décorateur ; décorateur avec arguments ; functools.wraps
- Usages courants : chronométrage, cache, nouvelle tentative
- Exercice d’entraînement : décorateur @chrono qui affiche la durée d’une fonction

Générateurs et gestionnaires de contexte (2h)

- yield et yield from ; expressions génératrices
- Compréhensions de dictionnaires et d’ensembles ; itertools
- Gestionnaires de contexte : __enter__ et __exit__, contextlib.contextmanager
- TP évalué : décorateur @retry, traitement d’un gros fichier par une chaîne de générateurs, et gestionnaire de contexte qui chronomètre un bloc

Module 4 : Fichiers de données et modules standards (3h30)

CSV, JSON et XML (1h30)

- csv : reader, DictReader, writer, DictWriter ; encodages et délimiteurs
- json : load, dump ; objets personnalisés
- XML avec ElementTree ; configuration TOML avec tomllib
- Exercice d’entraînement : convertir un flux XML en JSON

Fichiers, dates et journalisation (2h)

- pathlib, shutil et glob

- datetime et fuseaux horaires avec zoneinfo
- logging : niveaux, handlers, formats
- Variables d'environnement
- TP évalué : importer, filtrer et exporter un catalogue (CSV vers JSON), avec gestion des erreurs de lecture, dates converties et journal des opérations

Jour 3 (7h) : Tests, projet professionnel et projet de synthèse

Module 5 : Tests et structure d'un projet (3h)

Tester avec pytest (1h30)

- Pourquoi tester ; premier test et assert
- Fixtures ; @pytest.mark.parametrize
- Mesurer la couverture avec pytest-cov
- Exercice d'entraînement : tester la classe Panier

Structurer un projet Python (1h30)

- Structure src/ et tests/ ; pyproject.toml ; pip install -e .
- Environnement virtuel : venv, ou l'outil uv
- Ruff pour formater et analyser le code ; mypy pour vérifier les types
- TP évalué : transformer le code du jour 2 en projet installable avec pyproject.toml, des tests pytest des classes principales (fixtures, paramétrage), Ruff et mypy sans erreur

Module 6 : Projet de synthèse, évalué (4h)

Réalisation (2h30)

- Projet de synthèse, réalisé individuellement : outil d'analyse de données à partir de fichiers CSV et JSON : classes avec héritage et dataclasses, lecture, transformation et export, décorateur et générateur, journalisation, tests pytest, projet installable

Revue et retour (1h30)

- Revue de code collective : conception des classes, tests, lisibilité
- Retour individuel du formateur sur le projet rendu

Le contenu peut être ajusté selon le niveau, les prérequis et les besoins des participants.

5 COMPÉTENCES VISÉES

- Concevoir des classes Python idiomatiques
- Écrire un code Python expressif : décorateurs, générateurs, gestionnaires de contexte
- Traiter des fichiers de données avec Python
- Fiabiliser et structurer un projet Python

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou à distance, en classe virtuelle. Le formateur alterne entre méthode démonstrative, méthode interrogative (analyse critique de code existant et des choix de conception) et méthode active (exercices d'entraînement, TP évalués et projet de synthèse).

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Dépôt GitHub de démonstration avec exercices et corrections par module
- Environnement de développement : VS Code + Python 3.12 ou plus récent + pytest, Ruff et mypy
- Pour la classe virtuelle : visioconférence (Microsoft Teams ou équivalent), partage d’écran, chat en direct
- Supports remis par e-mail et sur la plateforme pédagogique LaPolaris (ressources, émargement)

En formation intra-entreprise sur le site du client, celui-ci s’assure et s’engage à disposer de toutes les ressources matérielles nécessaires à son bon déroulement (salle, postes de travail, logiciels, accès à Internet), conformément aux prérequis indiqués dans le programme et aux moyens prévus dans la convention.

8 MODALITÉS D'ÉVALUATION

- Questionnaire de positionnement individuel renseigné par chaque participant avant le démarrage de la formation, analysé par le formateur et donnant lieu, le cas échéant, à un ajustement du déroulé
- Exercices d’entraînement réalisés par chaque participant sur son poste, corrigés en séance
- Évaluation des acquis par les TP évalués, réalisés individuellement sur poste : un par module, puis le projet de synthèse décrit dans le dernier module du programme ; chaque objectif est évalué dans au moins un TP
- Grille de suivi renseignée par le formateur pour chaque participant et pour chacun des objectifs de l’action, au regard des critères d’évaluation ci-dessous
- Questionnaire d’appréciation renseigné par les participants en clôture de formation, et enquête à froid dans un délai de trois à six mois

9 CRITÈRES D'ÉVALUATION

- Classes aux attributs et méthodes cohérents ; propriété qui valide la donnée ; dataclass pour les objets de données ; annotations de type
- Hiérarchie avec super() et classe abstraite ; protocole respecté par plusieurs classes ; objets affichables, comparables et itérables
- Décorateur écrit avec functools.wraps et réutilisé sur plusieurs fonctions ; générateur qui traite un gros fichier sans tout charger en mémoire ; gestionnaire de contexte qui libère toujours la ressource
- Fichiers lus et écrits avec le bon encodage ; erreurs de lecture gérées ; dates converties ; opérations journalisées au bon niveau
- Tests qui passent ; fixtures et paramétrage utilisés ; cas limites testés ; couverture mesurée
- Projet installable avec pip install -e . ; dépendances déclarées dans pyproject.toml ; Ruff et mypy sans erreur

10 INDICATEURS DE RÉSULTATS

Les indicateurs de résultats de cette formation (taux de satisfaction et taux de réussite) seront publiés sur sa fiche en ligne à l’issue des premières sessions réalisées.

11 MODALITÉS DE VALIDATION

Une attestation de fin de formation est remise à chaque participant à l'issue du parcours. Elle mentionne **l'intitulé** et la **nature de l'action**, sa **durée** en heures, sa **période de réalisation**, **les objectifs de l'action** et le **niveau d'atteinte** de chacun d'eux, exprimé en trois valeurs : **atteint**, **partiellement atteint**, **non atteint**. **Une attestation d'assiduité**, établie à partir des émargements, est délivrée dans les mêmes conditions.

Formation non certifiante : elle ne délivre pas de certification enregistrée au RNCP ou au Répertoire spécifique.

12 SUIVI ET ACCOMPAGNEMENT

- Émargement électronique signé par chaque participant et par le formateur à chaque demi-journée, en présentiel comme à distance
- Traçabilité des activités pédagogiques réalisées
- Attestation d'assiduité délivrée en fin de formation
- Suivi individuel via les QCM de fin de journée, les exercices corrigés et les TP évalués

13 CONDITIONS D'ACCÈS

Formation accessible sans prérequis administratif particulier. Cette formation n'est pas éligible aux financements publics ou mutualisés, notamment au compte personnel de formation et à une prise en charge par un opérateur de compétences.

14 DÉLAIS D'ACCÈS

Inscription possible jusqu'à 15 jours ouvrés avant le début de la session pour permettre la préparation logistique et, si besoin, l'adaptation du contenu au contexte client. Pour toute demande spécifique, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations peuvent être accessibles aux personnes en situation de handicap. La question d'un besoin d'aménagement (matériel, temporel ou pédagogique) est posée lors de l'analyse du besoin et du questionnaire de positionnement ; vous pouvez aussi nous le signaler à tout moment, y compris pendant la formation, afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris
TÉL. +33762584798
EMAIL contact@lapolaris.fr
WEB lapolaris.fr