PYTHON & DATA : API, INTELLIGENCE ARTIFICIELLE

Tests automatisés en Python avec pytest

Apprenez à écrire des tests automatisés pour votre code Python avec pytest : tests unitaires, fixtures, paramétrage, mocks et mesure de la couverture de code.

DURÉE	TARIF	NIVEAU	LANGUE	GROUPE	FORMAT
3j (21h)	**1 275 € / participant** Inter-entreprise. Intra : sur devis, au plus 1 275 € / participant	**Intermédiaire**	**Français**	**1-6**	**Formation** À distance

1 PUBLIC VISÉ

- Développeurs Python qui veulent fiabiliser leur code avec des tests automatisés
- Développeurs back-end qui veulent intégrer l'écriture de tests à leur travail quotidien

2 PRÉREQUIS

- Écrire en Python des fonctions et des classes, gérer les exceptions et importer des modules
- Aucune expérience des tests automatisés n'est requise

3 OBJECTIFS PÉDAGOGIQUES

- Organiser un projet de tests pytest, en séparant les tests unitaires des tests d'intégration : dossier tests, conftest.py, pytest.ini
- Écrire des tests unitaires lisibles selon le principe Arrange, Act, Assert, y compris pour les exceptions attendues
- Créer des fixtures réutilisables, avec le bon scope et un nettoyage après usage, et les partager avec conftest.py
- Paramétrer un test pour couvrir les cas nominaux, les cas limites et les cas d'erreur, et filtrer les tests avec des marqueurs
- Remplacer les dépendances externes (API, fichiers, variables d'environnement, date et heure) par des mocks et vérifier les appels reçus
- Produire un rapport de couverture avec pytest-cov, compléter les tests des lignes non couvertes et fixer un seuil minimum

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) : Organiser ses tests et écrire des tests lisibles

Module 1 : Les types de tests et l'organisation d'un projet pytest (2h)

Les types de tests et la place de pytest (1h)

- Tests unitaires, d'intégration et end-to-end : ce que vérifie chacun
- pytest, unittest et doctest : différences et choix
- Ce qu'un test automatisé apporte : non-régression, documentation, confiance dans les modifications
- Exercice d'entraînement : classer les tests d'un projet fourni selon leur type

Installer pytest et organiser le projet (1h)

- Installer pytest dans un environnement virtuel
- Organiser le projet : dossier tests, conftest.py, pytest.ini
- Conventions de nommage : fichiers test_*.py, fonctions test_*
- Lancer pytest et lire le résultat : options -v, -x, -k, --tb
- Étendre pytest avec des plug-ins : installation et exemples (pytest-mock, pytest-cov, pytest-xdist)
- TP évalué : organiser le projet de démonstration en séparant les tests unitaires des tests d'intégration, puis lancer la suite

Module 2 : Écrire des tests unitaires lisibles (2h)

Les assertions pytest (1h)

- L'instruction assert : pytest détaille l'écart entre la valeur obtenue et la valeur attendue
- Comparer des nombres décimaux avec pytest.approx
- Vérifier les exceptions attendues avec pytest.raises et les avertissements avec pytest.warns
- Écrire le test avant le code : le cycle du TDD (test qui échoue, code minimal, remaniement)
- Exercice d'entraînement : développer en TDD une classe métier, avec ses cas nominaux et ses cas d'erreur

Des tests lisibles et maintenables (1h)

- Le principe Arrange, Act, Assert
- Un comportement vérifié par test
- Nommer un test pour qu'il décrive le comportement attendu
- Tester le comportement, pas l'implémentation : éviter les faux positifs
- Les erreurs courantes : tests qui dépendent les uns des autres, tests trop longs
- TP évalué : réécrire une suite de tests difficile à lire selon le principe Arrange, Act, Assert et ajouter les tests des exceptions attendues

Module 3 : Les fixtures (3h)

Créer et utiliser des fixtures (1h30)

- Le décorateur @pytest.fixture et l'injection par le nom du paramètre
- Les scopes : function, class, module, session
- Les fixtures fournies par pytest : tmp_path, capsys, caplog
- Exercice d'entraînement : fixture d'une base de données SQLite en mémoire

Fixtures avancées et partage (1h30)

- Nettoyer après usage avec yield
- Fixtures paramétrées et fixtures qui dépendent d'autres fixtures
- Partager les fixtures entre fichiers avec conftest.py
- TP évalué : fixtures d'un client HTTP simulé et d'un utilisateur connecté, avec le bon scope, nettoyées après usage et partagées par conftest.py

Jour 2 (7h) : Paramétrer, filtrer et isoler les tests

Module 4 : Paramétrage et marqueurs (3h)

Paramétrer les tests (1h30)

- @pytest.mark.parametrize : plusieurs cas pour une seule fonction de test
- Nommer les cas pour un rapport lisible : ids
- Combiner plusieurs paramétrages

- Exercice d'entraînement : paramétrer le test d'une fonction de calcul

Classer et filtrer avec des marqueurs (1h30)

- Les marqueurs fournis par pytest : skip, skipif, xfail
- Créer ses propres marqueurs et les déclarer dans pytest.ini
- Lancer seulement certains tests : pytest -m, pytest -k
- Lancer les tests en parallèle avec pytest-xdist
- TP évalué : paramétrer les tests d'une fonction de validation (cas nominaux, limites et d'erreur), marquer les tests lents et les exclure de l'exécution courante

Module 5 : Remplacer les dépendances par des mocks (4h)

Les bases des mocks (2h)

- Pourquoi isoler le code testé de ses dépendances externes
- unittest.mock : Mock, MagicMock et patch
- pytest-mock : la fixture mocker
- Où remplacer un objet : à l'endroit où il est utilisé
- Vérifier les appels reçus : assert_called_once_with, call_args_list
- Exercice d'entraînement : remplacer un appel HTTP fait avec requests et vérifier les appels

Mocks avancés (2h)

- Remplacer des variables d'environnement avec monkeypatch.setenv
- Isoler les accès aux fichiers avec tmp_path et monkeypatch
- Fixer la date et l'heure : datetime.now, time.sleep
- side_effect : simuler une exception ou des retours successifs
- autospec : garder un mock fidèle à l'objet remplacé
- TP évalué : tester un service d'envoi d'e-mails sans envoyer de message, en remplaçant le serveur d'envoi, la date et une variable d'environnement, et en vérifiant les appels reçus

Jour 3 (7h) : Mesurer la couverture et projet de synthèse

Module 6 : Mesurer et améliorer la couverture (3h)

Mesurer la couverture (1h30)

- Installer et configurer pytest-cov
- Produire un rapport dans le terminal et au format HTML
- Lire le rapport : lignes et branches non couvertes
- Exclure le code non testable avec # pragma: no cover
- Exercice d'entraînement : produire et lire le rapport de couverture d'un module existant

Améliorer la couverture et fixer un seuil (1h30)

- Choisir les parties à tester en priorité
- Fixer un seuil minimum avec --cov-fail-under et l'enregistrer dans la configuration
- Un taux de couverture élevé ne garantit pas des tests utiles
- TP évalué : compléter les tests d'un module jusqu'au seuil fixé et produire le rapport HTML

Module 7 : Projet de synthèse, évalué (4h)

- Écriture individuelle d’une suite de tests complète sur un module Python fourni : fonctions utilitaires et classe de service avec dépendances externes
- Projet organisé : dossier tests, conftest.py, pytest.ini
- Tests lisibles, paramétrés et marqués, couvrant les cas nominaux, les cas limites et les cas d’erreur
- Fixtures partagées et dépendances externes remplacées par des mocks
- Rapport de couverture HTML atteignant le seuil fixé
- Revue de code collective : lisibilité, isolation, pertinence des assertions
- Retour individuel du formateur sur les tests rendus

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

5 COMPÉTENCES VISÉES

- Concevoir une suite de tests pytest organisée et maintenable pour un projet Python
- Écrire des tests unitaires et d’intégration lisibles couvrant les cas nominaux, les cas limites et les cas d’erreur
- Isoler le code testé de ses dépendances grâce aux fixtures et aux mocks
- Analyser un rapport de couverture et compléter les tests des zones non couvertes

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée à distance, en classe virtuelle. Le formateur alterne entre méthode démonstrative, méthode interrogative (analyse des stratégies de test et des choix d’isolation) 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
- Exercices pratiques avec corrigés pour chaque module
- Environnement de développement : VS Code + Python 3.12+ + pytest, pytest-mock, pytest-cov et pytest-xdist
- 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, le client s’assure et s’engage à mettre à disposition de chaque participant un poste de travail avec les logiciels nécessaires et un 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

- Projet de tests organisé (dossier tests, conftest.py, pytest.ini), tests unitaires et tests d'intégration séparés, suite exécutée sans erreur avec pytest
- Tests structurés selon le principe Arrange, Act, Assert, un comportement par test, exceptions attendues vérifiées avec pytest.raises
- Fixtures au scope adapté, nettoyées après usage et partagées par conftest.py sans duplication entre fichiers
- Tests paramétrés couvrant les cas nominaux, les cas limites et les cas d'erreur ; tests lents marqués et exclus sur demande
- Dépendances externes remplacées par des mocks, appels reçus vérifiés, sans lier les tests à l'implémentation
- Rapport de couverture HTML produit, lignes non couvertes complétées par des tests, taux au moins égal au seuil fixé dans la configuration

10 INDICATEURS DE RÉSULTATS

Cette formation a été créée en 2026. Ses indicateurs de résultats (taux de satisfaction, taux de réalisation, nombre de stagiaires formés) seront publiés sur la fiche en ligne de cette formation 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
- 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