Fiche du cours

60 h

Titre :

PYS350 - Programmation Python pour les sciences

Description :

Cours pratique de programmation Python appliquée aux sciences : calcul numérique (NumPy/SciPy), analyse de données (Pandas/xarray), visualisation (Matplotlib), unités & incertitudes (pint/uncertainties), modélisation (ODE simples, ajustement de modèles), traitement du signal & de l'image (bases), formats scientifiques (CSV/HDF5/NetCDF), reproductibilité & qualité (Jupyter, conda, tests). Chaque séance relie code, concepts et phénomènes réels (physique, bio/éco, environnement

Objectifs :

- Structurer du code scientifique propre (fonctions, modules, notebooks → scripts) avec conda/venv.*
- Manipuler des tableaux NumPy, vectoriser et profiler des calculs ; utiliser SciPy (optimisation, intégration).*
- Charger, nettoyer et joindre des données avec Pandas/xarray ; tracer des figures de qualité (Matplotlib).*
- Gérer unités & incertitudes (pint, uncertainties) et ajuster des modèles (régression linéaire/non linéaire).*
- Résoudre des ODE simples (croissance/logistique, RC/oscillateur) et aborder signal & image (FFT/filtrage, mesures).*
- Travailler avec HDF5/NetCDF, documenter et tester le code (pytest), préparer un mini-déploiement (CLI/Docker – aperçu).

Chapitres :

1. Environnement & qualité : conda/venv, Jupyter/VS Code, structure projet, pytest, organisation notebooks*
2. NumPy I : ndarrays, dtypes, slicing, broadcasting, vectorisation, petit profilage*
3. Pandas I : DataFrame, nettoyage (NA/outliers), jointures, agrégations, séries temporelles (bases)*
4. Visualisation : Matplotlib (axes, légendes, barres d'erreur), styles "publication", export figures*
5. Unités & incertitudes : pint, uncertainties, propagation simple, vérifs dimensionnelles*
6. SciPy – Numérique : intégration, interpolation, optimisation (curve_fit, minimize), validation d'ajustement*

7. Modèles & ODE : croissance/décroissance, logistique, RC/oscillateur ; solve_ivp, pas de temps & stabilité (notions)*
8. Signal (bases) : FFT, filtres simples, fenêtrage ; extraction de pics/puissances (données expérimentales)*
9. Image (aperçu) : scikit-image (filtrage, seuillage, mesure d'objets), limites et artefacts*
10. Données scientifiques : HDF5/NetCDF, xarray (grilles/temps), géo/atmo/océan (exemples)*
11. Performance & packaging : vectorisation avancée, Numba (aperçu), mini-CLI, pyproject.toml*
12. Capstone : étude de bout en bout (choix du domaine : physique/bio/éco/env) → données + modèle + figures + rapport

À la fin :

Vous saurez programmer en Python pour la science : vectoriser des calculs, analyser des données, ajuster & valider des modèles, gérer unités/incertitudes et produire des figures reproductibles. Vous livrerez une mini-étude complète (code + données + rapport) — prête pour un portfolio ou pour poursuivre en physique computationnelle, data science ou ML scientifique.