Roadmap Formation Développement Python

Du Natif aux Frameworks Web

Objectif : Maîtriser Python du niveau débutant aux frameworks web modernes (Django/Flask) en 6 mois intensifs

Python Natif	**Frameworks Web**	**Écosystème**
Syntaxe & POO	Flask & FastAPI	Data Science
Structures de données	Django	APIs & Déploiement

Formation Professionnelle en Développement Python

28 juillet 2025

Table des matières

1 Vue d'ensemble de la Formation Python

Pourquoi Python ?

Python est le langage idéal pour débuter en programmation grâce à sa syntaxe claire et lisible. Il offre une polyvalence exceptionnelle : développement web, data science, intelligence artificielle, automatisation. Cette roadmap vous mènera du niveau débutant à un niveau professionnel en 6 mois.

1.1 Parcours d'Apprentissage

— **Durée totale :** 6 mois (24 semaines)
— **Charge de travail :** 20-25h par semaine
— **Répartition :** 40% théorie, 60% pratique
— **Projets :** 6 projets évolutifs + 1 application finale
— **Version Python :** Python 3.11+

1.2 Prérequis

— Bases de la programmation (variables, boucles, fonctions)
— Compréhension des concepts POO
— Familiarité avec les structures de données
— Expérience avec au moins un langage de programmation

2 MOIS 1 : Python Avancé et Écosystème

2.1 Semaines 1-2 : Fonctionnalités Avancées

2.1.1 Programmation Fonctionnelle

— Fonctions lambda et closures
— map(), filter(), reduce() avancés
— List/Dict/Set comprehensions complexes
— Générateurs et yield expressions
— Décorateurs avec paramètres
— Métaclasses et descripteurs

2.1.2 Gestion Avancée des Données

— Context managers personnalisés
— Sérialisation avancée (pickle, JSON, XML)
— Expressions régulières complexes
— Manipulation de fichiers volumineux
— Streaming et processing en temps réel

2.2 Semaines 3-4 : Architecture et Design Patterns

2.2.1 Design Patterns Avancés

— Singleton, Factory, Observer patterns
— Dependency Injection
— Strategy et Command patterns
— Adapter et Facade patterns
— MVC/MVP/MVVM architectures

2.2.2 Programmation Asynchrone

— async/await et coroutines
— asyncio event loop
— Concurrent.futures
— Threading vs Multiprocessing
— Async context managers

</> Projet du Mois 1

Framework ORM Personnalisé

— Création d'un mini-ORM avec métaclasses
— Query builder avec method chaining
— Connection pooling et lazy loading
— Décorateurs pour caching et validation
— Architecture plugin avec dependency injection
— Tests unitaires avec mocking avancé

3 MOIS 2 : Python Avancé et Modules

3.1 Semaines 1-2 : Fonctionnalités Avancées

3.1.1 Programmation Fonctionnelle

— Fonctions lambda
— map(), filter(), reduce()
— List/Dict/Set comprehensions
— Générateurs et yield
— Décorateurs avancés

3.1.2 Gestion des Fichiers et Données

— Lecture/écriture de fichiers
— Context managers (with)

— JSON et sérialisation
— CSV et données tabulaires
— Expressions régulières (regex)

3.2 Semaines 3-4 : Modules et Packages

3.2.1 Bibliothèque Standard

— os et pathlib
— datetime et time
— collections (defaultdict, Counter)
— itertools et functools
— logging et debugging

3.2.2 Packages Externes Essentiels

— requests pour HTTP
— BeautifulSoup pour web scraping
— pandas pour données
— matplotlib pour graphiques
— pytest pour tests

</> Projet du Mois 2

Analyseur de Données Web
— Web scraping avec requests/BeautifulSoup
— Collecte de données depuis APIs
— Nettoyage et analyse avec pandas
— Visualisations avec matplotlib
— Sauvegarde en différents formats
— Tests unitaires avec pytest

4 MOIS 3 : Développement Web avec Flask

4.1 Semaines 1-2 : Flask Fondamentaux

4.1.1 Introduction à Flask

— Installation et premier serveur
— Routes et méthodes HTTP
— Templates avec Jinja2
— Fichiers statiques (CSS, JS, images)
— Formulaires et validation

4.1.2 Base de Données

— SQLite et SQLAlchemy
— Modèles et relations
— Migrations avec Flask-Migrate
— CRUD operations
— Sessions et cookies

4.2 Semaines 3-4 : Flask Avancé

4.2.1 Authentification et Sécurité

— Flask-Login pour sessions
— Hachage de mots de passe
— Protection CSRF
— Rôles et permissions
— Variables d'environnement

4.2.2 APIs et JSON

— API REST avec Flask
— Sérialisation JSON
— Gestion d'erreurs HTTP
— Documentation API
— Tests d'API

</> Projet du Mois 3

Blog Personnel avec API

— Application web complète
— Système d'authentification
— CRUD pour articles et commentaires
— Interface admin
— API REST pour mobile
— Déploiement sur Heroku

5 MOIS 4 : FastAPI et APIs Modernes

5.1 Semaines 1-2 : FastAPI Fondamentaux

5.1.1 Introduction à FastAPI

— Installation et premier serveur
— Type hints et validation automatique

— Documentation automatique (Swagger)
— Modèles Pydantic
— Gestion des erreurs

5.1.2 Base de Données avec FastAPI

— SQLAlchemy avec FastAPI
— Modèles et schémas
— Dependency Injection
— Sessions de base de données
— Migrations Alembic

5.2 Semaines 3-4 : FastAPI Avancé

5.2.1 Authentification et Sécurité

— JWT tokens
— OAuth2 avec scopes
— Middleware de sécurité
— CORS et headers
— Rate limiting

5.2.2 Performance et Déploiement

— Programmation asynchrone (async/await)
— Background tasks
— Caching avec Redis
— Tests avec pytest-asyncio
— Containerisation Docker

</> Projet du Mois 4

API E-commerce

— API REST complète
— Authentification JWT
— Gestion produits/commandes
— Paiements (simulation)
— Documentation Swagger
— Tests automatisés
— Déploiement Docker

6 MOIS 5 : Django Framework

6.1 Semaines 1-2 : Django Fondamentaux

6.1.1 Architecture Django

— MVT (Model-View-Template)
— Projet vs Applications
— Settings et configuration
— URLs et routing
— Views et templates

6.1.2 Modèles et ORM

— Définition de modèles
— Relations (ForeignKey, ManyToMany)
— QuerySet et ORM
— Migrations
— Admin interface

6.2 Semaines 3-4 : Django Avancé

6.2.1 Formulaires et Vues

— Django Forms
— Class-based views
— Generic views
— Middleware personnalisé
— Signaux Django

6.2.2 Authentification et Sécurité

— Système d'authentification Django
— Permissions et groupes
— Sécurité CSRF/XSS
— Django REST Framework
— Pagination et filtrage

</> Projet du Mois 5

Plateforme de Formation en Ligne

— Application Django complète
— Gestion cours/étudiants/instructeurs
— Système de paiement
— Interface admin avancée
— API REST avec DRF
— Tests et déploiement

7 MOIS 6 : Spécialisation et Projet Final

7.1 Semaines 1-2 : Data Science avec Python

7.1.1 Analyse de Données

— NumPy pour calculs numériques
— Pandas pour manipulation de données
— Matplotlib/Seaborn pour visualisation
— Jupyter Notebooks
— Statistiques descriptives

7.1.2 Machine Learning Basics

— Scikit-learn introduction
— Régression et classification
— Preprocessing des données
— Évaluation de modèles
— Intégration ML dans web apps

7.2 Semaines 3-4 : DevOps et Production

7.2.1 Déploiement et Monitoring

— Docker et containerisation
— CI/CD avec GitHub Actions
— Déploiement cloud (AWS/Heroku)
— Monitoring et logging
— Performance optimization

7.2.2 Bonnes Pratiques

— Code quality (PEP 8, Black)
— Documentation (Sphinx)
— Tests avancés (coverage, mocking)
— Sécurité en production
— Maintenance et updates

</> Projet Final

Plateforme SaaS Complète

— Application web multi-tenant
— API REST avec authentification
— Dashboard avec analytics
— Intégration ML/AI
— Paiements et abonnements
— Déploiement production
— Documentation complète

8 Ressources et Outils

8.1 Environnement de Développement

Outils Essentiels

— **IDE :** PyCharm Professional, VS Code
— **Version Control :** Git, GitHub
— **Package Management :** pip, pipenv, poetry
— **Testing :** pytest, unittest, coverage
— **Linting :** pylint, flake8, black

8.2 Ressources d'Apprentissage

Documentation et Communauté

— Documentation officielle Python
— Real Python (tutorials)
— Python.org et PEPs
— Stack Overflow
— Reddit r/Python
— PyPI pour packages

9 Évaluation et Certification

9.1 Critères d'Évaluation

— **Projets (70%)** : Fonctionnalité, qualité du code, documentation
— **Tests techniques (20%)** : Algorithmes, résolution de problèmes
— **Présentation (10%)** : Communication, démonstration

9.2 Portfolio Professionnel

— Repository GitHub avec 6+ projets
— Applications web déployées
— Contributions open source
— Blog technique
— Profil LinkedIn optimisé

10 Préparation aux Certifications Python

10.1 Stratégie de Certification

Progression Certifiante

Chaque mois de formation inclut une préparation spécifique aux certifications Python reconnues dans l'industrie, avec examens blancs et projets validants.

10.2 Certifications Ciblées par Niveau

10.2.1 Niveau Fondamental (Mois 2-3)

— **PCEP (Python Certified Entry-Level Programmer)** : Bases Python
— **PCAP (Python Certified Associate Programmer)** : Programmation avancée

10.2.2 Niveau Professionnel (Mois 4-5)

— **PCPP1 (Python Certified Professional Programmer 1)** : Modules avancés
— **Django Developer Certification** : Framework web
— **AWS Certified Developer - Associate** : Cloud Python

10.2.3 Spécialisations (Mois 6)

— **Microsoft Azure AI Fundamentals** : IA et ML
— **Google Professional Data Engineer** : Data Science
— **Certified Kubernetes Application Developer** : DevOps

10.3 Planning de Préparation

10.3.1 Mois 1-2 : Fondations

— Révision syntaxe Python (3h/semaine)
— Exercices PCEP (1h/jour)
— Examen blanc PCEP (fin mois 2)

10.3.2 Mois 3-4 : Certification Intermédiaire

— Préparation PCAP intensive (5h/semaine)
— Projets pratiques validants
— Passage PCEP (début mois 3)
— Passage PCAP (fin mois 4)

10.3.3 Mois 5-6 : Certifications Professionnelles

— Préparation PCPP1 (4h/semaine)
— Préparation AWS Developer (3h/semaine)
— Projets cloud et déploiement
— Passage certifications (fin mois 6)

10.4 Ressources de Préparation

Matériel d'Étude

Python Institute :
— Python Institute Study Materials
— OpenEDG Python Essentials
— Practice Tests et Mock Exams
Cloud Certifications :
— AWS Training and Certification
— A Cloud Guru courses
— Hands-on Labs et Sandboxes
Frameworks :
— Django Official Documentation
— Real Python tutorials
— FastAPI documentation

10.5 Examens Blancs et Validation

10.5.1 Programme d'Évaluation

— **Fréquence :** 1 examen blanc par certification/mois

— **Projets validants :** Code review par experts
— **Peer programming :** Sessions collaboratives
— **Portfolio GitHub :** Projets documentés

10.5.2 Critères de Validation

— Score minimum 80% aux examens blancs
— Projets fonctionnels et déployés
— Code respectant les standards PEP 8
— Documentation complète

11 Perspectives de Carrière

11.1 Rôles Accessibles

— **Python Developer** : Applications web et desktop
— **Backend Developer** : APIs et services
— **Data Analyst** : Analyse et visualisation
— **DevOps Engineer** : Automatisation et infrastructure
— **Full-Stack Developer** : Web complet

11.2 Spécialisations

— **Data Science/ML** : Intelligence artificielle
— **Web Development** : Django/Flask expert
— **Automation** : Scripts et outils
— **Scientific Computing** : Recherche et simulation

12 Conclusion

Objectifs Atteints

À l'issue de cette formation de 6 mois, vous maîtriserez :
— Python natif et programmation orientée objet
— Développement web avec Flask, FastAPI et Django
— Bases de données et APIs REST
— Bonnes pratiques et déploiement
— Spécialisations (Data Science ou DevOps)

Cette roadmap Python vous prépare à devenir un développeur polyvalent, capable de créer des applications web modernes, des APIs robustes et d'explorer des domaines avancés comme la data science.