

Python développeurs

Syntaxe



Kévin Duranty
Formateur Data & Webmarketing



Planning de la formation

Lundi 01/12

Découverte & bases

1. **Configuration de l'environnement de développement** (venv, notebook)
2. **Déclaration de variable** et types de données avancés
3. **Structures de contrôle** : conditions et boucles (for, while)
4. **Utilisation de fonction standards** : print, input, range
5. **Déclaration de fonction**, gestion des arguments multiples
6. **Gestion des exceptions** : try, except, raise
7. **Cas pratique** : Création d'un programme de gestion de mot de passe.

Mardi 02/12

Fonctions, Modules & Documentation

1. **Import de bibliothèques** : os, requests
2. **Installation et manipulation d'une bibliothèque** : FastAPI
3. **Cas pratique** : Conception d'une API avec FastAPI
4. **Programmation orientée objet** : attribut, méthode et héritage
5. **Conception de modules** et packages python

Mercredi 03/12

POO & Interface graphique

1. Gestion de matrice de données avec la **librairie Numpy**
2. Gestion de fichiers de données avec la **librairie Pandas**
3. Exemples de projets python (automatisation, scraping, IA, interface web) et limites du langage
4. **Cas pratique** : Conception d'une interface de gestion de données avec Streamlit et pandas.

Dépôt GitHub : <https://github.com/Quera-fr/Python-developpeurs-Syntaxe>

Dossier Google Drive : https://drive.google.com/drive/folders/1bm4FqjW5S78jzjgbXwvj2HsFsc3QAF0n?usp=drive_link

Ressources pédagogiques

Jour 1 – Bases & syntaxe Python

Lundi 01/12

Découverte & bases

- **Introduction au langage Python.**
- **Données et fonctions standards.**
- **Opérateurs logiques**, Conditions et Boucles.
- Manipulation de **bibliothèques standards.**
- **Collections de données** : listes, tuples, dictionnaires, ensembles.

[Accédez au dossier](#)

Logiciel et bibliothèques

- [Python3.13.7](#) : l'exécutable d'installation et ses bibliothèques de base.
- [Git Bash](#) : un terminal qui fournit un environnement proche de Linux.
- [Visual Studio Code](#) : l'éditeur de code moderne, léger et extensible.
- [OS](#) et [Time](#) : Bibliothèques standards permettant de manipuler le système d'exploitation et gérer les données sous forme de dates.

Les IDE (**Interface de développement Intégrée**) les plus populaires :

- [Jupyter Lab](#): Environnement utilisé dans le domaine de la science des données et de l'apprentissage automatique. Il prend en charge le langage Python et offre une interface utilisateur conviviale pour la création de documents appelés "Notebooks".
- [Google Colaboratory](#) : basée sur le cloud, permet d'écrire et d'exécuter du code Python dans un environnement de notebook. Accès gratuit à différentes ressources matérielles GPU/CPU/TPU, idéal pour l'apprentissage machine et la recherche. Permet également le travail collaboratif à l'instar des ressources Google : Docs, Sheets, Slides, etc...
- [Anaconda](#) : Permet de gérer facilement environnements, librairies et lancer des IDE comme JupyterLab ou Spyder.

Jour 2 – Fonctions & structuration

Mardi 02/12

Fonctions, Exceptions, Tests & Documentation

- **Import de bibliothèques** : os, requests
- **Installation et manipulation** d'une bibliothèque : **FastAPI**
- **Cas pratique** : Conception d'une API avec **FastAPI**
- **Programmation orientée objet** : attribut, méthode et héritage
- **Conception de modules et packages python**

[Accédez au dossier](#)

Bibliothèques :

- **OS** : Fournit des outils pour interagir avec le système d'exploitation (gestion de fichiers, répertoires, variables d'environnement).
- **Request** : Permet d'effectuer facilement des requêtes HTTP (GET, POST, PUT, DELETE) pour communiquer avec des API externes.
- [FastAPI](#) : Bibliothèque moderne et populaire permettant de créer des API REST sécurisées et documentées.

Exemples utiles :

- [API RATP](#)
- [API EDF](#)
- [API Région Ile-De-France](#)

Projet : Conception d'une API documentée et sécurisée.

Jour 3 – POO & bases de données

Mercredi 03/12

POO & bibliothèques & bases de données

- Gestion de matrice de données avec la librairie Numpy
- Gestion de fichiers de données avec la librairie Pandas
- Exemples de projets python (automatisation, scraping, IA, interface web) et limites du langage
- Cas pratique : Conception d'une interface de gestion de données avec Streamlit et pandas.

[Accédez au dossier](#)

Bibliothèques :

- [Numpy](#) : Bibliothèque fondamentale pour le calcul scientifique, offrant des tableaux multidimensionnels performants et un large ensemble de fonctions mathématiques.
- [Pandas](#) : Outil incontournable pour la manipulation et l'analyse de données, proposant des structures comme les DataFrame pour travailler efficacement avec des données tabulaires.
- [Seaborn](#) : Bibliothèque de visualisation basée sur Matplotlib, permettant de créer rapidement des graphiques statistiques clairs, élégants et adaptés à l'exploration de données.
- [Streamlit](#) : Bibliothèque Python permettant de créer facilement des applications web interactives et des tableaux de bord.

Projet : Conception d'un package de gestion de base de données (POO + SQLite).

Introduction

1. Introduction à Python



Qu'est-ce que [Python](#) ?

Python est un langage de programmation polyvalent, **interprété** et facile à apprendre.

- **Langage de haut niveau** : syntaxe expressive, proche de l'anglais. Facile à lire, comprendre, écrire.
- **Interprété** : pas de compilation séparée, exécution directe par l'interpréteur. Débogage plus fluide.
- **Typage dynamique** : pas besoin de déclarer les types à l'avance, ils sont déterminés au moment de l'exécution.
- **Multiplateforme** : fonctionne sur Windows, macOS, Linux, Android, iOS, Raspberry Pi, serveurs...
- **Gratuit & Open Source**, développé par [Guido van Rossum](#) (Informaticien né aux Pays-Bas) depuis 1989 (première version publique en 1991), maintenu via la Python Software Foundation.
- **Grande communauté & écosystème riche** : bibliothèque standard très fournie + milliers de librairies externes (NumPy, Pandas, Flask, Django...) facilitant des domaines variés.

Le nom **Python** est inspiré de **Monty Python's Flying Circus**, une **série comique britannique** des années 1970 que Guido van Rossum appréciait.

1. Introduction à Python

Une des langues les plus faciles à apprendre. Ressources en ligne : [OpenClassrooms](#), [Udemy](#), [Youtube](#).

Communauté active ([Stackoverflow](#)), langage de haut niveau.

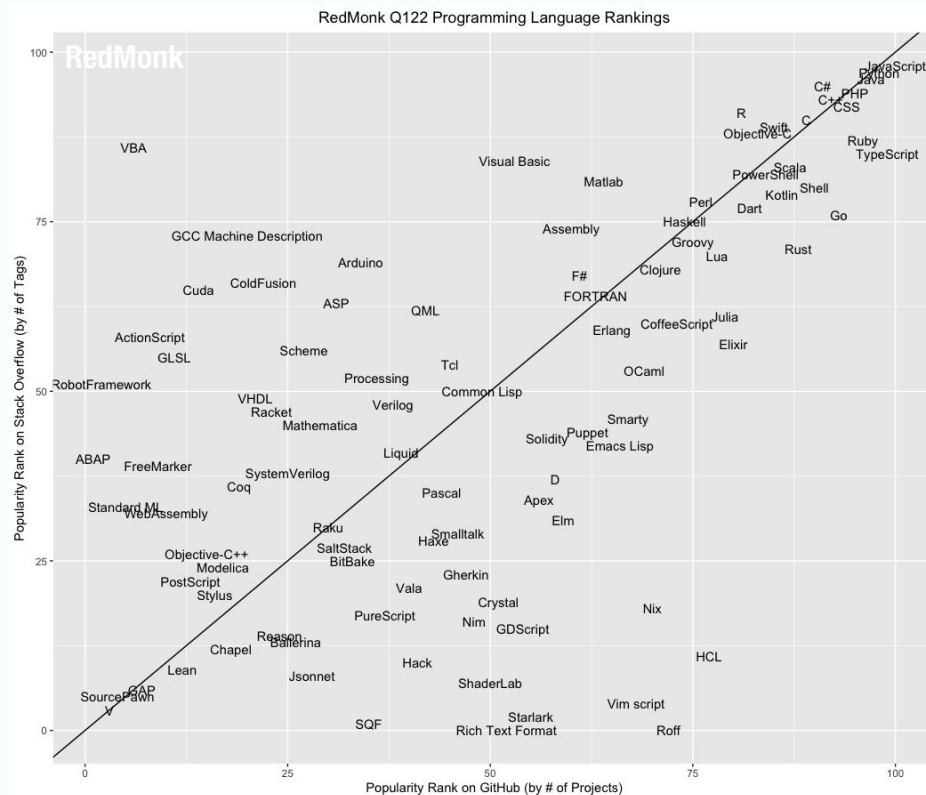
Langage enseigné au lycée en [Seconde générale et technologique](#) depuis 2018.

1er langage le plus utilisé par la communauté de développeurs d'après [TIOBE](#).

Versions Python 2 vs Python 3 : Python 2 (fin de vie en 2020) n'est plus maintenu. On utilise désormais exclusivement Python 3.x.

Interpréteurs / distributions courants :

- *CPython* (standard, en C) ;
- *PyPy* (JIT, performant) ;
- *Anaconda* (destinée à la data science, gestion d'environnement)



1. Introduction à Python

Cas d'utilisation

Développement web (back-end, APIs) avec frameworks comme Django, Flask, FastAPI.

Science des données & Data science : analyse, visualisation, traitement de données avec Pandas, NumPy, Matplotlib, Seaborn...

Machine Learning / Intelligence Artificielle : frameworks comme TensorFlow, Keras, PyTorch rendent Python leader dans ce domaine.

Automatisation / Scripting : tâches répétitives (manipulation de fichiers, scraping, administration), prototypage rapide.

Web scraping / extraction de données : BeautifulSoup, Scrapy permettent d'extraire des informations de pages web efficacement.

Applications desktop (GUI) : interfaces graphiques via Tkinter, PyQt, Kivy.

Jeux : prototypage, petits jeux avec Pygame, Pyglet ; certains jeux commerciaux utilisent Python pour scripting.

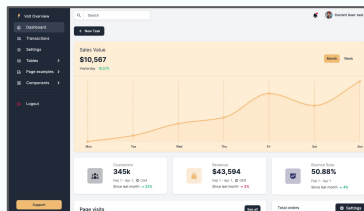
Entreprises & ERP / e-commerce : Odoo, Tryton ou d'autres solutions métiers intégrées.

Cloud, APIs, microservices : Python très utilisé côté serveur pour orchestrer des services.

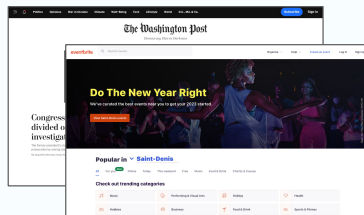
IoT, embarqué, fintech, blockchain, cybersécurité : domaines en croissance où Python intervient pour prototypage, scripts, outils.

1. Introduction à Python

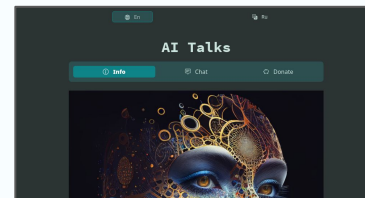
Exemples types d'applications Web



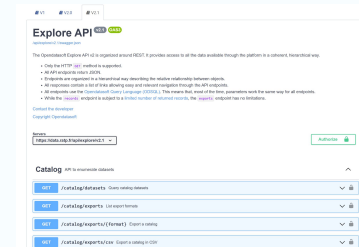
[Flask](#) est un micro-framework web Python qui permet de créer des applications web de manière simple et légère.



[Django](#) est un puissant framework web open source écrit en Python, conçu pour simplifier le développement d'applications web complexes et robustes.



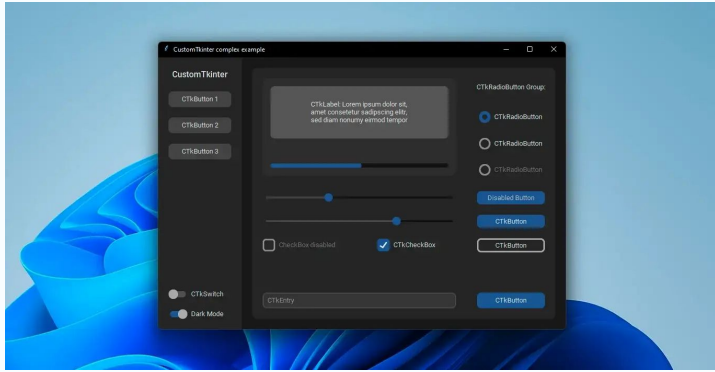
[Streamlit](#) est une bibliothèque open source de Python conçue pour simplifier la création d'applications web interactives et de tableaux de bord de données.



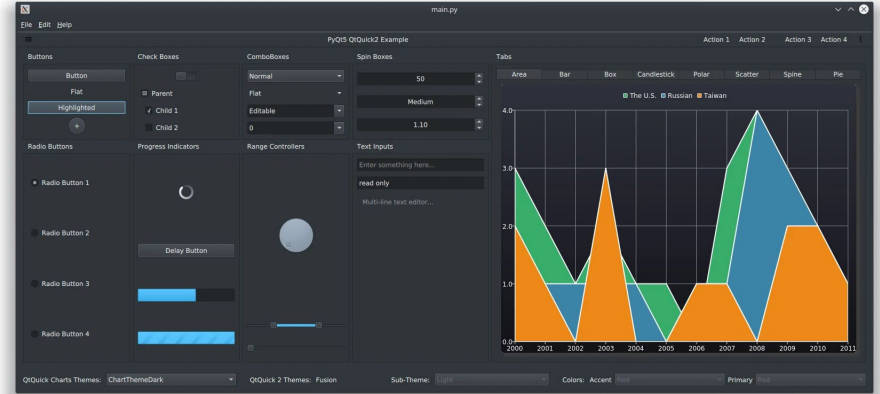
[FastAPI](#) est un framework web conçu pour créer des APIs REST rapidement tout en profitant des annotations de types, de la validation automatique des données et de la documentation interactive intégrée.

1. Introduction à Python

Exemples types d'applications bureautique



[Tkinter](#) est une bibliothèque standard de Python utilisée pour créer des interfaces graphiques utilisateur (GUI, pour Graphical User Interface).



PySide est une bibliothèque Python qui fournit des liaisons (bindings) officielles vers le framework [Qt](#), permettant de créer des interfaces graphiques multiplateformes sous licence LGPL.