Engineer

Python

langage utilisé par la plupart des outils de ce site

Ce que c’est

Python est un langage de programmation généraliste dont le code source est exécuté par un interpréteur, plutôt que compilé à l’avance en un binaire distinct. Sa documentation le décrit comme interprété, interactif et orienté objet, avec des modules, des exceptions, un typage dynamique et des types de données de haut niveau intégrés. Le trait que la plupart des gens remarquent en premier est que l’indentation fait partie de la syntaxe : un bloc est ce qui est indenté sous la ligne qui l’ouvre. La disposition vue par le lecteur est donc la structure exécutée par l’interpréteur. Guido van Rossum l’a commencé pendant les vacances de Noël 1989, en s’appuyant sur son travail sur le langage ABC au CWI d’Amsterdam. Il l’a publié en février 1991 et nommé d’après les scripts de Monty Python’s Flying Circus qu’il lisait alors.

Le typage est dynamique. Un nom est lié à un objet pendant l’exécution, et le type appartient à l’objet, pas au nom : la même variable peut donc contenir un nombre à une ligne et une chaîne à la suivante. Les annotations facultatives, normalisées par la PEP 484, permettent à une fonction de déclarer ce qu’elle attend et renvoie. Mais cette PEP précise qu’aucune vérification de type n’a lieu pendant l’exécution et que Python restera un langage à typage dynamique. Les annotations sont lues par des outils distincts, des vérificateurs de types et des éditeurs, tandis que l’interpréteur les conserve sans les imposer.

Le langage est livré avec une vaste bibliothèque standard, que le tutoriel décrit par l’expression « piles incluses » : fichiers, JSON, SQLite, expressions régulières, sous-processus, analyse des arguments et bien d’autres outils, sans rien installer. Le reste provient de paquets tiers, généralement installés avec pip depuis le Python Package Index. Comme un interpréteur ne peut contenir qu’une version de chaque paquet, la bibliothèque standard fournit aussi `venv`, qui crée un environnement autonome par projet : une application peut ainsi utiliser la version 1.0 d’une bibliothèque et une autre la 2.0.

Les nouvelles versions suivent un calendrier. Depuis Python 3.9, la PEP 602 prévoit une version apportant de nouvelles fonctionnalités chaque mois d'octobre, après environ dix-sept mois de développement, puis cinq ans de maintenance. À partir de la 3.13, cela signifie deux ans de versions correctives régulières, environ tous les deux mois, puis trois ans de correctifs de sécurité uniquement. Python 3.15.0 est paru le 9 octobre 2026. Python 3.10 a atteint sa fin de vie le 1er octobre 2026, et Python 2.7, dernière version de la deuxième série, ne reçoit plus de support depuis le 1er janvier 2020. L'interpréteur de référence, CPython, possède depuis longtemps un verrou global qui n'autorise qu'un seul thread à exécuter du code Python à la fois. La PEP 703 a ajouté une variante de l'interpréteur sans ce verrou, et la PEP 779 a associé la version 3.14 à la phase où cette variante sans verrou global est officiellement prise en charge, mais reste facultative.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Python ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. La partie invisible pour les utilisateurs : où résident les données, quelles règles s'appliquent et comment les autres systèmes peuvent y accéder.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 6 pipelines ETL/ELT conçus et exploités — 6 pipelines ETL sur trois moteurs de bases de données

6 pipelines ETL/ELT conçus et exploités Contactez-nous →

Pourquoi c'est important

Python est souvent le chemin le plus court entre une question et un programme fonctionnel. Un script qui lit un fichier, interroge une base et écrit un rapport peut être réalisé avec la seule bibliothèque standard, exécuté immédiatement et modifié sans étape de construction. C'est pourquoi Python sert si souvent de liant dans les pipelines de données, les outils de construction et les bancs de test.

La lisibilité est un objectif de conception délibéré, pas un effet secondaire. Le Zen de Python, la PEP 20, affirme que la lisibilité compte, et la PEP 8, guide de style de la bibliothèque standard, tranche les questions dont une équipe débattrait autrement, du nommage à la ligne de 79 caractères.

La souplesse a un coût qui apparaît tardivement. Un attribut mal orthographié, un `None` à la place d'une chaîne ou un argument du mauvais type ne devient une erreur que lorsque cette ligne est exécutée, éventuellement en production. Les équipes qui utilisent Python pour un travail durable ajoutent donc les contrôles

laissés de côté par le langage : un linter, des annotations avec un vérificateur de types dans la chaîne de validation, et des tests qui exécutent réellement les chemins du code.

Le calendrier compte autant que le code. Chaque version de l'interpréteur possède une date de fin fixe : une installation à laquelle personne ne touche sort donc du support de sécurité selon un calendrier publié. Un projet qui indique sa version minimale et évolue pendant les deux années de corrections de bogues échange une urgence contre une mise à niveau annuelle planifiée.

À quoi cela ressemble en pratique

Dans le portfolio, Python faisait fonctionner un générateur de CV et de portfolio piloté par une base de données à une étape documentée : 41 modules et 10 580 lignes produisaient six formats, de HTML et PDF à Markdown et JSON, à partir d'une base SQLite de 23 tables. Un pipeline de 19 étapes construisait cette base, chacune pouvant être répétée sans risque. Les dépendances d'exécution avaient été limitées à exactement deux, un moteur de modèles et un moteur de rendu PDF, au motif qu'un générateur de documents impossible à installer dans cinq ans n'a rien préservé. En contrepartie, tous les documents dépendaient alors de ce code Python, sans autre moyen d'en produire un. Ces chiffres décrivent l'implémentation à cette étape documentée.

Python a aussi remporté une comparaison directe dans un pipeline de consommation électrique horaire, réalisé de trois façons, en Python, en JQ et en SQL sur PostgreSQL, pour transformer des relevés irréguliers en une ligne par heure et par zone, la consommation nette étant la production plus les importations moins les exportations. La version composée de fonctions Python pures était la plus rapide : elle traitait une journée de données en 34 ms et un jeu de données de 30 jours en 180 ms.

Ce site est construit par Hugo. Dans l'instantané de l'outillage consigné en octobre 2026, ses 22 scripts auxiliaires, soit 8 408 lignes au total, étaient écrits en Python. Tous n'utilisaient que la bibliothèque standard, sauf le script d'importation des dessins, qui avait besoin de Pillow pour les recadrer. Le fichier `.tool-versions` indiquait Python 3.14.7, avec pour règle que l'interpréteur pouvait être plus récent, mais jamais plus ancien. `task doctor` le vérifiait sur le binaire installé au lieu de faire confiance au fichier. Ruff était le linter, et basedpyright vérifiait les types au niveau `standard`, car son niveau par défaut avait signalé 358 problèmes concernant presque tous des annotations

manquantes dans du code volontairement simple. Dès sa première exécution, le vérificateur de types a trouvé une valeur susceptible d'être `None` utilisée comme clé de dictionnaire, un défaut que Ruff avait laissé passer sans diagnostic.

Où les problèmes surviennent

La plupart des problèmes Python proviennent d'un code que l'interpréteur accepte sans signaler d'erreur et dont le comportement diffère de ce que sa lecture laisse attendre.

Un argument par défaut conserve la mémoire des appels précédents. Une fonction renvoie une liste qui contient encore des éléments d'appels antérieurs : les résultats dépendent donc de l'ordre des appels. Une valeur par défaut est évaluée une seule fois, lors de l'exécution de l'instruction `def` . Une valeur mutable, comme une liste ou un dictionnaire, est donc un objet partagé entre tous les appels qui omettent l'argument. Utilisez `None` par défaut et créez la liste dans le corps de la fonction ; la règle B006 de Ruff signale ce motif.

```
def add_tag(tag, tags=None):
    if tags is None:
        tags = []  # a new list on every call
    tags.append(tag)
    return tags
```

Les projets cassent leurs dépendances respectives. Mettre à jour une bibliothèque pour un projet en casse un autre, ou `pip install` s'arrête avec une erreur concernant un environnement géré en dehors de pip. Dans les deux cas, des paquets sont installés directement dans un interpréteur partagé : les projets se disputent une seule copie de chaque bibliothèque, et la PEP 668 permet à un distributeur de système d'exploitation d'interdire à pip de modifier son propre Python. Donnez à chaque projet son environnement avec `python -m venv .venv` , installez-y les paquets et consignez les dépendances dans un fichier pour pouvoir reconstruire l'environnement.

Les annotations de type sont prises pour des vérifications. Une fonction annotée pour recevoir un `int` reçoit une chaîne provenant d'un fichier JSON, et l'échec apparaît trois appels plus loin. Les annotations décrivent une intention ; l'interpréteur ne les impose pas. Exécutez un vérificateur de types dans la chaîne

de validation pour que les incompatibilités entre portions de code annotées fassent échouer la construction. Validez les données à leur entrée dans le programme, puisqu'aucune annotation ne peut garantir les entrées.

Un interpréteur dépasse sa période de support. Un serveur conserve la version installée jusqu'à l'arrêt des versions de sécurité et à son retrait des versions prises en charge par les paquets. Chaque version apportant de nouvelles fonctionnalités bénéficie de cinq ans, et les cinq ans de la 3.10 se sont terminés le 1er octobre 2026. Indiquez la version minimale dans un fichier lu par une vérification, passez à une version supérieure pendant les deux années de corrections de bogues de chaque version et exécutez la suite de tests avec la nouvelle avant de basculer.

Pour aller plus loin

- Documentation Python — Le tutoriel, la référence de la bibliothèque et celle du langage pour la version actuelle.
- Le tutoriel Python — Le langage depuis ses principes de base, avec l'avertissement sur les arguments par défaut mutables.
- PEP 8 – Guide de style du code Python — Les conventions de disposition et de nommage suivies par la plupart du code Python.
- PEP 602 – Cycle annuel des versions de Python — Pourquoi une version apportant de nouvelles fonctionnalités paraît chaque mois d'octobre et combien de temps elle est maintenue.
- PEP 484 – Annotations de type — La syntaxe des annotations et la raison pour laquelle l'interpréteur laisse leur vérification à d'autres outils.
- Environnements virtuels et paquets — Créer un environnement par projet avec `venv` et y installer les paquets avec pip.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que Python ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/python/