Logo

Formation

introduction python

Logo

Présenté par

elbachir

Rappels et Contexte sur Python

Origine et évolution de Python

Python a été créé par Guido van Rossum et publié pour la première fois en 1991. Il a été conçu pour être un langage de programmation facile à lire et à écrire, avec une syntaxe claire et concise. Python a évolué au fil des années pour devenir l'un des langages de programmation les plus populaires, grâce à sa simplicité et à sa polyvalence.

- `Créé par Guido van Rossum en 1991`
- `Inspiré par des langages comme ABC, Modula-3, et C`
- `Évolution continue avec des versions majeures comme Python 2 et Python 3`

Les avantages de Python par rapport à d'autres langages

Python se distingue par plusieurs avantages qui en font un choix privilégié pour de nombreux développeurs et entreprises.

- Lisibilité : La syntaxe de Python est claire et intuitive, ce qui facilite la lecture et la maintenance du code.
- Polyvalence : Python peut être utilisé pour une grande variété de tâches, du développement web à l'analyse de données en passant par l'intelligence artificielle.
- Communauté active : Python bénéficie d'une large communauté de développeurs qui contribuent à son écosystème en créant des bibliothèques et des frameworks.
- Portabilité : Python est compatible avec de nombreux systèmes d'exploitation, ce qui en fait un choix idéal pour le développement multiplateforme.

Avantage	Description
Lisibilité	Syntaxe claire et intuitive
Polyvalence	Utilisable dans de nombreux domaines
Communauté	Large communauté de développeurs
Portabilité	Compatible avec de nombreux systèmes d'exploitation

Les domaines d'application de Python

Python est utilisé dans de nombreux domaines, grâce à sa polyvalence et à la richesse de son écosystème.

Domaine	Exemples d'outils
Développement Web	Django, Flask
Data Science	Pandas, NumPy, Matplotlib
Intelligence Artificielle	TensorFlow, Keras, PyTorch
Automatisation	Selenium, BeautifulSoup
Jeux vidéo	Pygame

Versions de Python : Python 2 vs Python 3

Python 2 et Python 3 sont deux versions majeures du langage Python, avec des différences significatives.

- `Python 2 : Publié en 2000, Python 2 a été largement utilisé pendant de nombreuses années. Cependant, il a atteint la fin de son support en 2020.`
- `Python 3 : Publié en 2008, Python 3 introduit de nombreuses améliorations et corrections de bugs par rapport à Python 2. Il est recommandé d'utiliser Python 3 pour tous les nouveaux projets.`

Caractéristique	Python 2	Python 3
Support	Terminé en 2020	Actuellement supporté
Syntaxe	Moins intuitive	Plus claire et intuitive
Fonctionnalités	Moins de fonctionnalités modernes	Introduit de nombreuses améliorations

Installation et Configuration de l'Environnement

Installation de Python sur différents systèmes d'exploitation

Python est compatible avec plusieurs systèmes d'exploitation. Voici comment l'installer sur les plus courants :

- Windows : Téléchargez l'installateur depuis le site officiel de Python (python.org) et suivez les étapes de l'assistant d'installation. Assurez-vous de cocher l'option 'Add Python to PATH' pour faciliter l'utilisation de Python depuis la ligne de commande.
- macOS : Python est préinstallé sur macOS, mais il est recommandé d'installer la dernière version via Homebrew. Utilisez la commande `brew install python` dans le terminal.
- Linux : Python est souvent préinstallé sur les distributions Linux. Pour installer ou mettre à jour Python, utilisez le gestionnaire de paquets de votre distribution (par exemple, `sudo apt-get install python3` sur Ubuntu).

Utilisation de pip pour installer des packages

pip est le gestionnaire de paquets par défaut de Python. Il permet d'installer, mettre à jour et supprimer des packages Python.

- Installation d'un package : Utilisez la commande `pip install nom_du_package` pour installer un package spécifique. Par exemple, `pip install numpy` installe la bibliothèque NumPy.
- Mise à jour d'un package : Pour mettre à jour un package, utilisez `pip install --upgrade nom_du_package`.
- Suppression d'un package : Pour supprimer un package, utilisez `pip uninstall nom_du_package`.
- Liste des packages installés : La commande `pip list` affiche tous les packages installés dans l'environnement actuel.

Configuration d'un environnement virtuel avec venv

Les environnements virtuels permettent d'isoler les dépendances d'un projet. `venv` est un module intégré à Python pour créer et gérer ces environnements.

- Création d'un environnement virtuel : Utilisez la commande `python -m venv nom_du_repertoire` pour créer un nouvel environnement virtuel dans le répertoire spécifié.
- Activation de l'environnement : Sur Windows, utilisez `nom_du_repertoire\Scripts\activate`. Sur macOS/Linux, utilisez `source nom_du_repertoire/bin/activate`.
- Désactivation de l'environnement : Tapez simplement `deactivate` dans le terminal.
- Installation de packages dans l'environnement : Une fois activé, utilisez `pip install` comme d'habitude pour installer des packages uniquement dans cet environnement.

Introduction aux IDE (PyCharm, VS Code, Jupyter Notebook)

Les IDE (Environnements de Développement Intégrés) facilitent le développement en Python en offrant des fonctionnalités avancées comme la complétion de code, le débogage et la gestion de projets.

- PyCharm : Recommandé pour les projets complexes et les développeurs professionnels.

- `VS Code : Idéal pour les développeurs qui travaillent avec plusieurs langages ou qui préfèrent un éditeur léger.`
- `Jupyter Notebook : Parfait pour les scientifiques des données et les analyses interactives.`

IDE	Description	Avantages
PyCharm	IDE dédié à Python développé par JetBrains.	Puissant, support complet pour Python, intégration avec des outils comme Git et Docker.
VS Code	Éditeur de code léger et extensible développé par Microsoft.	Léger, support pour de nombreux langages, extensions disponibles pour Python.
Jupyter Notebook	Environnement interactif basé sur le web.	Idéal pour l'analyse de données et la visualisation, supporte le code Python en temps réel.

Les Fondamentaux de Python

Variables et types de données

En Python, une variable est un conteneur pour stocker des données. Les types de données définissent la nature des valeurs que peut contenir une variable. Voici les principaux types de données en Python :

- `int : Entiers, comme 5 ou -3.`
- `float : Nombres à virgule flottante, comme 3.14 ou -0.001.`
- `str : Chaînes de caractères, comme 'Bonjour' ou 'Python'.`
- `bool : Booléens, True ou False.`
- `list : Liste ordonnée et modifiable, comme [1, 2, 3].`
- `tuple : Liste ordonnée et immuable, comme (1, 2, 3).`
- `set : Collection non ordonnée et sans doublons, comme {1, 2, 3}.`
- `dict : Dictionnaire de paires clé-valeur, comme {'nom': 'Alice', 'âge': 25}.`

Opérateurs

Les opérateurs en Python permettent de manipuler des valeurs et des variables. Ils sont classés en plusieurs catégories :

Catégorie	Opérateurs	Exemples
Arithmétiques	+, -, *, /, //, %, **	5 + 3, 10 / 2, 2 ** 3
Comparaison	==, !=, >, <, >=, <=	5 == 5, 10 > 20
Logiques	and, or, not	True and False, not True
Affectation	=, +=, -=, *=, /=	x = 5, x += 3

Structures de contrôle

Les structures de contrôle permettent de diriger le flux d'exécution du programme en fonction de conditions ou de répéter des blocs de code.

- `if : Exécute un bloc de code si une condition est vraie.`

- `elif : Exécute un bloc de code si la condition précédente est fausse et que la condition actuelle est vraie.`
- `else : Exécute un bloc de code si toutes les conditions précédentes sont fausses.`
- `match (Python 3.10) : Structure de contrôle pour les correspondances de motifs.`

Boucles

Les boucles permettent de répéter un bloc de code plusieurs fois. Python propose deux types de boucles :

- `for : Répète un bloc de code pour chaque élément d'une séquence.`
- `while : Répète un bloc de code tant qu'une condition est vraie.`
- `break : Interrompt la boucle immédiatement.`
- `continue : Passe à l'itération suivante de la boucle.`

Fonctions et Modules

Définition et appel de fonctions

Une fonction en Python est un bloc de code réutilisable qui effectue une tâche spécifique. Elle est définie à l'aide du mot-clé `def`, suivi du nom de la fonction et de ses paramètres entre parenthèses. Pour exécuter une fonction, on l'appelle par son nom suivi de parenthèses contenant les arguments nécessaires.

- Définition : `def nom_fonction(param1, param2):`
- Appel : `nom_fonction(arg1, arg2)`

Paramètres et arguments

Les paramètres sont les variables définies dans la fonction, tandis que les arguments sont les valeurs passées lors de l'appel de la fonction. Python supporte plusieurs types d'arguments :

- Positionnels : Les arguments sont passés dans l'ordre des paramètres.
- Nommés : Les arguments sont passés avec le nom du paramètre.
- Par défaut : Les paramètres ont une valeur par défaut si aucun argument n'est fourni.
- *args : Permet de passer un nombre variable d'arguments positionnels.

- `**kwargs : Permet de passer un nombre variable d'arguments nommés.`

Portée des variables : local vs global

La portée d'une variable détermine où elle peut être utilisée dans le code. En Python, il existe deux types de portée :

- `Locale : Une variable définie dans une fonction est locale à cette fonction et ne peut être utilisée qu'à l'intérieur de celle-ci.`
- `Globale : Une variable définie en dehors de toute fonction est globale et peut être utilisée partout dans le code.`

Création et importation de modules

Un module en Python est un fichier contenant des définitions et des instructions Python. Il permet d'organiser le code en le divisant en plusieurs fichiers. Pour utiliser un module, il faut l'importer dans un autre fichier.

- `Création : Un module est simplement un fichier `.py` contenant du code Python.`
- `Importation : On utilise le mot-clé `import` pour importer un module.`

- Alias : On peut donner un alias à un module lors de l'importation avec `import module as alias`.
- Importation spécifique : On peut importer uniquement certaines fonctions ou variables d'un module avec `from module import fonction`.

Structures de Données Avancées

Listes : méthodes courantes

Les listes en Python sont des collections ordonnées et modifiables. Voici quelques méthodes couramment utilisées :

- append(x) : Ajoute un élément x à la fin de la liste.
- extend(iterable) : Ajoute tous les éléments de l'itérable à la fin de la liste.
- pop([i]) : Supprime et retourne l'élément à la position i (par défaut, le dernier élément).
- sort(key=None, reverse=False) : Trie les éléments de la liste en place.

Dictionnaires : accès, ajout, suppression, itération

Les dictionnaires sont des collections non ordonnées de paires clé-valeur. Voici comment les manipuler :

- Accès : Utilisez la clé pour accéder à la valeur correspondante.
- Ajout : Assignez une valeur à une nouvelle clé pour l'ajouter.
- Suppression : Utilisez del ou pop pour supprimer une paire clé-valeur.
- Itération : Parcourez les clés, les valeurs ou les paires clé-valeur avec des méthodes comme keys(), values(), et items().

Compréhensions de listes et de dictionnaires

Les compréhensions permettent de créer des listes ou des dictionnaires de manière concise.

- Liste : [expression for item in iterable if condition]
- Dictionnaire : {key: value for item in iterable if condition}

Utilisation de collections (deque, defaultdict, Counter)

Le module collections offre des structures de données spécialisées :

- `deque : Une file doublement terminée pour des ajouts/suppressions rapides aux deux extrémités.`
- `defaultdict : Un dictionnaire qui fournit une valeur par défaut pour les clés manquantes.`
- `Counter : Un dictionnaire pour compter les occurrences d'éléments.`

Gestion des Fichiers et Exceptions

Ouverture, lecture et écriture de fichiers

Python offre des fonctions intégrées pour manipuler les fichiers. Pour ouvrir un fichier, on utilise la fonction `open()`. Cette fonction prend deux arguments principaux : le chemin du fichier et le mode d'ouverture (lecture, écriture, etc.). Une fois le fichier ouvert, on peut lire son contenu avec `read()` ou écrire dedans avec `write()`. Il est important de fermer le fichier après utilisation avec `close()` pour libérer les ressources.

- `open('fichier.txt', 'r')` : Ouvre un fichier en mode lecture.
- `read()` : Lit tout le contenu du fichier.
- `write('texte')` : Écrit du texte dans le fichier.
- `close()` : Ferme le fichier.

Gestion des exceptions avec try, except, finally

La gestion des exceptions permet de gérer les erreurs qui surviennent pendant l'exécution du code. Le bloc `try` contient le code susceptible de générer une exception. Si une exception est levée, le bloc `except` est exécuté. Le bloc `finally` est optionnel et s'exécute toujours, que l'exception soit levée ou non. Cela est utile pour libérer des ressources ou effectuer des nettoyages.

- `try:` : Bloc où l'on place le code susceptible de générer une exception.
- `except:` : Bloc exécuté si une exception est levée.
- `finally:` : Bloc exécuté dans tous les cas, après `try` et `except`.

Utilisation de with pour la gestion des ressources

Le mot-clé `with` est utilisé pour gérer les ressources de manière propre et sécurisée. Il garantit que les ressources sont libérées automatiquement, même si une exception est levée. Cela évite d'avoir à appeler manuellement `close()` pour les fichiers ou d'autres ressources. Le bloc `with` est particulièrement utile pour la gestion des fichiers.

- `with open('fichier.txt', 'r') as fichier:` : Ouvre un fichier et garantit sa fermeture automatique.
- Le fichier est fermé automatiquement à la fin du bloc `with`.

Sauvegarde et chargement de données : JSON, pickle

Python propose des modules pour sauvegarder et charger des données dans des formats structurés. Le module `` permet de convertir des objets Python en chaînes JSON et vice versa. Le module `pickle` permet de sérialiser et désérialiser des objets Python, ce qui est utile pour sauvegarder des structures de données complexes. Cependant, `pickle` est spécifique à Python et n'est pas sécurisé pour des données provenant de sources non fiables.

- `.dumps(obj)` : Convertit un objet Python en chaîne JSON.
- `.loads(str)` : Convertit une chaîne JSON en objet Python.
- `pickle.dump(obj, fichier)` : Sérialise un objet Python dans un fichier.
- `pickle.load(fichier)` : Désérialise un objet Python à partir d'un fichier.

Introduction à la Programmation Orientée Objet (POO)

Concepts de base : classes et objets

La Programmation Orientée Objet (POO) est un paradigme de programmation qui organise le code autour de 'objets', qui sont des instances de 'classes'. Une classe est un modèle ou un plan pour créer des objets. Elle définit les attributs (données) et les méthodes (comportements) que les objets auront.

- Classe : Un modèle ou un plan pour créer des objets.
- Objet : Une instance spécifique d'une classe.

Attributs et méthodes : instance, classe

Les attributs et méthodes peuvent être de deux types : d'instance ou de classe. Les attributs d'instance sont spécifiques à chaque objet, tandis que les attributs de classe sont partagés par toutes les instances de la classe. Les méthodes d'instance opèrent sur les données d'un objet spécifique, tandis que les méthodes de classe opèrent sur la classe elle-même.

- `Attributs d'instance : Uniques à chaque objet.`
- `Attributs de classe : Partagés par tous les objets de la classe.`
- `Méthodes d'instance : Opèrent sur les données d'un objet spécifique.`
- `Méthodes de classe : Opèrent sur la classe elle-même.`

Encapsulation : propriétés et méthodes privées

L'encapsulation est un concept clé en POO qui consiste à restreindre l'accès direct aux données d'un objet. Cela se fait en utilisant des propriétés et des méthodes privées. Les propriétés privées ne peuvent être accédées ou modifiées que via des méthodes publiques, ce qui permet de contrôler comment les données sont manipulées.

- Propriétés privées : Ne peuvent être accédées ou modifiées que via des méthodes publiques.
- Méthodes publiques : Permettent de manipuler les propriétés privées de manière contrôlée.

Héritage et polymorphisme

L'héritage permet à une classe de hériter des attributs et méthodes d'une autre classe, favorisant la réutilisation du code. Le polymorphisme permet à des objets de différentes classes de répondre à la même méthode de manière différente, ce qui rend le code plus flexible et extensible.

- Héritage : Une classe peut hériter des attributs et méthodes d'une autre classe.
- Polymorphisme : Des objets de différentes classes peuvent répondre à la même méthode de manière différente.

Conclusion

Python est un langage puissant et polyvalent

Python est reconnu pour sa simplicité et sa lisibilité, ce qui en fait un choix idéal pour les débutants comme pour les développeurs expérimentés. Sa polyvalence lui permet d'être utilisé dans une multitude de domaines, allant du développement web à l'analyse de données en passant par l'intelligence artificielle.

- `Simplicité et lisibilité`
- `Polyvalence dans divers domaines`
- `Large communauté et support`

Idéal pour une variété de projets

Que vous travailliez sur un petit projet personnel ou sur une application d'entreprise complexe, Python offre les outils et les bibliothèques nécessaires pour répondre à vos besoins. Sa compatibilité avec d'autres langages et technologies en fait un choix flexible pour l'intégration dans des systèmes existants.

- `Projets personnels et professionnels`

- Compatibilité avec d'autres technologies
- Bibliothèques et frameworks variés

Revue des bases essentielles et concepts avancés

Cette introduction a couvert les concepts fondamentaux de Python, tels que les variables, les structures de contrôle, les fonctions et les classes. Nous avons également exploré des concepts plus avancés comme les générateurs, les décorateurs et la gestion des exceptions, qui sont essentiels pour écrire du code Python efficace et robuste.

- Variables et types de données
- Structures de contrôle (boucles, conditions)
- Fonctions et classes
- Générateurs et décorateurs
- Gestion des exceptions

Continuez à explorer et à pratiquer

La maîtrise de Python nécessite de la pratique et de l'exploration continue. Engagez-vous dans des projets réels, participez à des communautés en ligne, et explorez les nombreuses ressources disponibles pour approfondir vos connaissances et compétences en Python.

- Pratique continue
- Participation à des projets réels
- Engagement dans des communautés en ligne
- Exploration de ressources supplémentaires