


La structure lexicale d'un langage

Nous allons vous montrer qu'un langage informatique, y compris Python, dispose de sa propre grammaire interne. Lorsque l'interpréteur Python parcourt le code, il cherche à catégoriser tout ce qu'il rencontre. Nous avons suffisamment présenté de concepts dans ce chapitre pour lister les catégories principales :

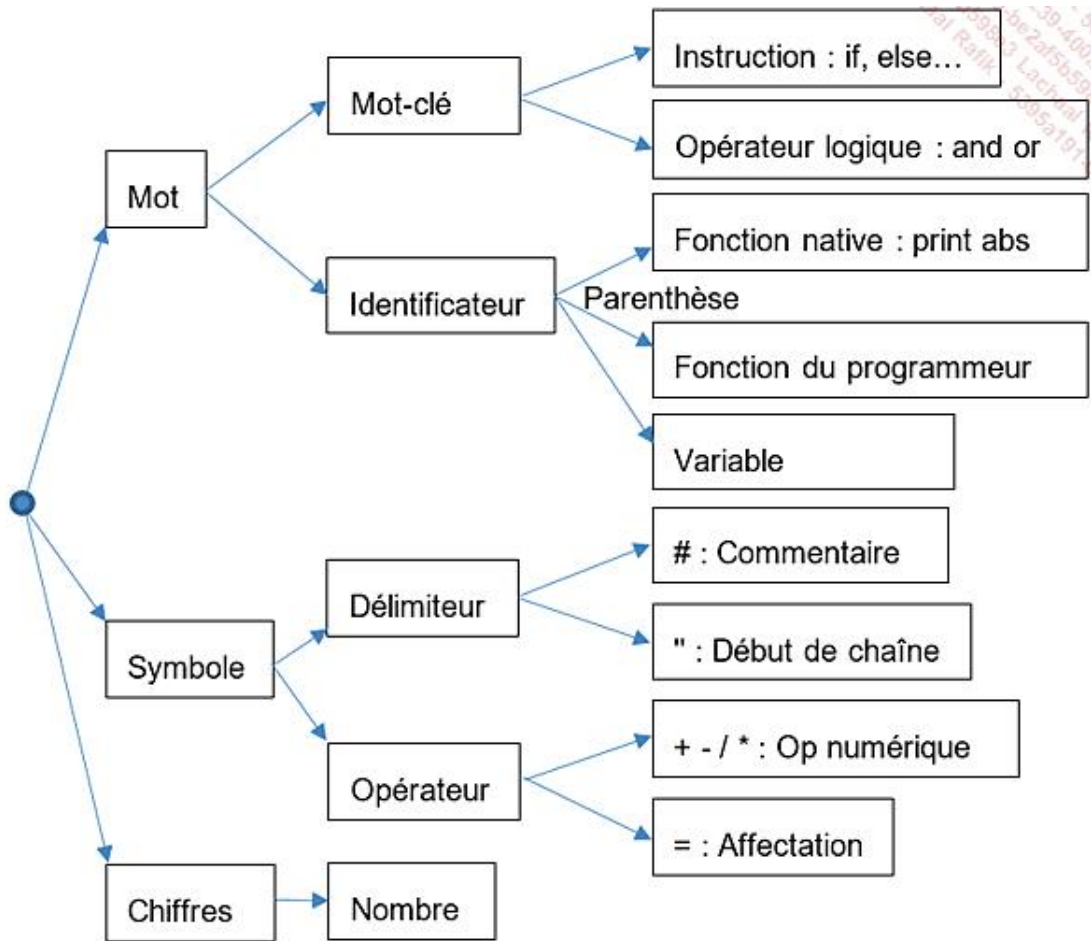
- Commentaires `# Bonjour`
- Indentations
- Délimiteurs `: ()`
- Mots-clés
 - Instruction `if then else`
 - Opérateurs logiques `and or not`
- Constantes `"Bonjour", 17, 3.14`
- Opérateurs
 - Calcul `+ - / *`
 - Affectation `=`
- Identificateurs
 - Variable créée par le développeur
 - Fonction native `print abs`
 - Fonction créée par le développeur

Nous détaillons chaque point ci-après :

- Les commentaires consistent à écrire des informations utiles pour le programmeur, mais ne produisant aucune action lors de l'exécution.
- L'indentation permet à l'interpréteur d'identifier les sous-parties du code associées à certaines instructions comme l'instruction `if`.
- Les délimiteurs sont importants pour structurer le code, on oublie souvent leur existence tellement ils passent inaperçus. Par exemple, vous trouvez le symbole `:` à la fin d'une condition `if`, les parenthèses `()` qui délimitent une expression, le symbole `'` qui marque le début et la fin d'une chaîne... On peut dire que les délimiteurs correspondent à la ponctuation dans un langage naturel.
- Les mots-clés représentent une liste de 30 mots réservés en Python. Ces mots ne peuvent pas être utilisés pour créer des noms de variables ou des noms de fonctions. Dans cette liste, vous connaissez les mots-clés `if/then/else`. Certains mots-clés sont des opérateurs logiques, comme `or`, `and` et `not`.
- Les constantes représentent des valeurs écrites directement à l'intérieur du programme. On peut rencontrer des chaînes de caractères : `"Bonjour"` ou des numériques, comme `44` ou `3.14`. En anglais, on utilise le terme *literals*.
- Les opérateurs peuvent être arithmétiques : `+` `-` `*`. On trouve aussi l'opérateur d'affectation défini par le symbole `=`.
- Les identificateurs correspondent à tout ce qui n'est pas une constante, un mot-clé, un délimiteur ou un opérateur... Il s'agit ici de la liberté laissée au développeur de créer ses propres noms. On parle donc des "nouveaux mots" dans le code correspondant aux noms des variables et des fonctions.

 Comment un développeur fait-il pour savoir si un nom correspond à celui d'une variable ou d'une fonction ? C'est simple : s'il est suivi d'une parenthèse ouvrante, il a affaire à une fonction, sinon à une variable.

Nous cherchons à faire passer l'idée que le langage informatique se structure suivant des règles grammaticales. Si vous connaissez l'existence de ces règles, cela vous aidera à décoder la structure d'un programme. Cependant, ces règles sont nombreuses et complexes, la présentation faite ici reste très simplifiée, même si elle est utile. Voici par exemple le cheminement que fait un programmeur dans son esprit lorsqu'il lit du code :



➤ Il n'est pas toujours possible d'associer un symbole à un unique rôle. En effet, si le symbole = est intrinsèquement associé à l'affectation, le symbole (accompagné du symbole) joue un rôle différent suivant le contexte : il peut décrire l'appel d'une fonction avec l'écriture `print()` ou la création d'un tuple sous la forme `(4,5,6)` ou encore une expression de la forme `5*(a+b)`.