

## A Langage Python

Cette annexe liste limitativement les éléments du langage Python (version 3 ou supérieure) dont la connaissance est exigible des étudiants. Aucun concept sous-jacent n'est exigible au titre de la présente annexe. Aucune connaissance sur un module particulier n'est exigible des étudiants. Toute utilisation d'autres éléments du langage que ceux que liste cette annexe, ou d'une fonction d'un module, doit obligatoirement être accompagnée de la documentation utile, sans que puisse être attendue une quelconque maîtrise par les étudiants de ces éléments.

### Traits généraux

- Typage dynamique : l'interpréteur détermine le type à la volée lors de l'exécution du code.
- Principe d'indentation.
- Portée lexicale : lorsqu'une expression fait référence à une variable à l'intérieur d'une fonction, Python cherche la valeur définie à l'intérieur de la fonction et à défaut la valeur dans l'espace global du module.
- Appel de fonction par valeur : l'exécution de  $f(x)$  évalue d'abord  $x$  puis exécute  $f$  avec la valeur calculée.

### Types de base

- Opérations sur les entiers (`int`) : `+`, `-`, `*`, `//`, `**`, `%` avec des opérandes positifs.
- Opérations sur les flottants (`float`) : `+`, `-`, `*`, `/`, `**`.
- Opérations sur les booléens (`bool`) : `not`, `or`, `and` (et leur caractère paresseux).
- Comparaisons `==`, `!=`, `<`, `>`, `<=`, `>=`.

### Types structurés

- Structures indicées immuables (chaînes, tuples) : `len`, accès par indice positif valide, concaténation `+`, répétition `*`, tranche.
- Listes : création par compréhension `[e for x in s]`, par `[e] * n`, par `append` successifs; `len`, accès par indice positif valide; concaténation `+`, extraction de tranche, copie (y compris son caractère superficiel); `pop` en dernière position.
- Dictionnaires : création `{c1 : v1, ..., cn : vn}`, accès, insertion, présence d'une clé `k in d`, `len`, `copy`.

### Structures de contrôle

- Instruction d'affectation avec `=`. Dépaquetage de tuples.
- Instruction conditionnelle : `if`, `elif`, `else`.
- Boucle `while` (sans `else`). `break`, `return` dans un corps de boucle.
- Boucle `for` (sans `else`) et itération sur `range(a, b)`, une chaîne, un tuple, une liste, un dictionnaire au travers des méthodes `keys` et `items`.
- Définition d'une fonction `def f(p1, ..., pn), return`.

### Divers

- Introduction d'un commentaire avec `#`.
- Utilisation simple de `print`, sans paramètre facultatif.
- Importation de modules avec `import module`, `import module as alias`, `from module import f, g, ...`
- Manipulation de fichiers texte (la documentation utile de ces fonctions doit être rappelée; tout problème relatif aux encodages est éludé) : `open`, `read`, `readline`, `readlines`, `split`, `write`, `close`.
- Assertion : `assert` (sans message d'erreur).