

# Éléments de Python

*Un langage facile d'abord où l'indentation n'est pas une option*

Valérie Ménissier-Morain

Document LU3IN024 – Version du 12 janvier 2022

Il s'agit de Python 3.

Lorsqu'une commande admet un paramètre nécessaire pour les explications il apparaît en italique bleu tel que *<paramètre>*. En revanche les parties optionnelles seront entourées de doubles crochets et écrites en rouge, ainsi *[[ partie optionnelle ]]*.

## Table des matières

<b>1</b>	<b>Affectation</b>	<b>1</b>
<b>2</b>	<b>Structures de contrôles</b>	<b>2</b>
2.1	Conditions . . . . .	2
<b>3</b>	<b>Commentaires</b>	<b>2</b>
<b>4</b>	<b>Opérations arithmétiques</b>	<b>2</b>
<b>5</b>	<b>Séquences</b>	<b>2</b>
5.1	Base . . . . .	3
5.1.1	Opérations . . . . .	3
5.1.2	Exemple d'utilisation : le découpage d'une séquence en colonnes . . . . .	3
5.1.3	Parcours . . . . .	3
5.2	Chaînes de caractères . . . . .	3
5.2.1	Conversions chaînes ↔ nombres . . . . .	3
5.2.2	Caractères et code ASCII . . . . .	4
5.2.3	Remplacement caractère par caractère . . . . .	4
5.3	Listes . . . . .	4
5.3.1	Méthodes . . . . .	4
5.3.2	Fonctions . . . . .	4
5.3.3	Conversion chaîne de caractères ↔ liste de caractères . . . . .	4
5.3.4	Range . . . . .	5
<b>6</b>	<b>Dictionnaires</b>	<b>5</b>
6.1	Syntaxe . . . . .	5
6.2	Exemple d'utilisation d'un dictionnaire : le calcul de fréquences dans une chaîne de caractères . . . . .	5
<b>7</b>	<b>Définition de listes, d'ensembles et de dictionnaires par compréhension</b>	<b>5</b>
<b>8</b>	<b>Entrées-sorties</b>	<b>5</b>
8.1	Clavier/écran . . . . .	5
8.2	Fichiers . . . . .	6
8.3	Ligne de commande . . . . .	6
8.4	Lire aussi bien dans un fichier dont on passe le nom en paramètre du programme que sur l'entrée standard . . . . .	6
<b>9</b>	<b>Utilisation de bibliothèques</b>	<b>6</b>

## 1 Affectation

*<variable>* = *<valeur>*

## 2 Structures de contrôles

```
<mot-clef> [[ paramètres ]]:  
    <instructions indentées (1 tabulation ou 4 caractères)>  
  
def <fonction> ([[ paramètres ]]):  
    <corps de la fonction>  
  
if <condition>:  
    <instructions_vrai>  
[[ elif <condition2>:  
    <instructions2_vraie> ]]  
[[ else:  
    <instructions_faux> ]]  
  
while <condition>:  
    <instructions>  
  
for <variable> in <structure à parcourir>:  
    <instructions pour chaque élément>
```

### 2.1 Conditions

Valeurs True, False  
Opérateurs **not**, **and**, **or**

## 3 Commentaires

```
# <commentaire>  
Un commentaire spécial en début de fonction : le docstring  
  
def <fonction> ([[ <paramètres> ]]):  
    """<signature de la fonction  
    hypothèses d'application  
    description observationnelle de la fonction.>"""  
  
    <corps de la fonction>
```

qui produit automatiquement l'accès à cette information :

```
% help(<fonction>)  
<signature de la fonction  
hypothèses d'application  
description observationnelle de la fonction.>
```

## 4 Opérations arithmétiques

+, -, \*, / (division flottante), // (division entière), % (modulo), \*\* (puissance).

## 5 Séquences

Une séquence *<s>* est un ensemble ordonné de valeurs, indicées de 0 à **len** (*<s>*) exclus

Les chaînes de caractères et les listes sont des séquences donc tout ce qui suit s'applique à ces deux structures de données.

## 5.1 Base

### 5.1.1 Opérations

- Test d'appartenance de l'élément `<e>` dans la séquence `<s>` : `<e> in <s>` et `<e> not in <s>`
- Concaténation : `<s1>+<s2>` la séquence formée des éléments de la séquence `<s1>` puis de ceux de la séquence `<s2>`
- Élément en position `<i>` : `<s>[<i>]`, origine des indices 0, si `<i>` est négatif (`-<k>`) c'est une abréviation pour `len(<s>)-<k>`
- Tranche `<s>[<i>:<j>]` ou tranche ajourée `<s>[<i>:<j>:<pas>]` : la séquence des valeurs de la séquence `<s>` de l'indice `<i>` à l'indice `<j>` exclus par pas de `<pas>`
  - si `<i>` est absent, c'est 0 par défaut
  - si `<j>` est absent, c'est `len(<s>)` par défaut
  - si `<pas>` est absent, c'est 1 par défaut
  - si `<pas>` est négatif `-k`, `<s>[<i>:<j>:$-k$]` renvoie la séquence `<s>[<i>:<j>:$k$]` en ordre inverse
- Longueur, valeur minimale, valeur maximale d'une séquence `<s>` : `len(<s>)`, `min(<s>)`, `max(<s>)`
- `<s>.index(<e>[[,<i>[[,<j>]]]])` indice de la première occurrence de l'élément `<e>` dans la séquence `<s>` (à partir de l'indice `<i>` et avant l'indice `<j>`; si l'élément `<e>` ne fait pas partie de la séquence `<s>` alors il y a une erreur
- Nombre d'occurrences de l'élément `<e>` dans la séquence `<s>` : `<s>.count(<e>)`

### 5.1.2 Exemple d'utilisation : le découpage d'une séquence en colonnes

La `<i>`-ème colonne d'une séquence `<s>` découpée en `<k>` colonnes : `<coli>=<s>[<i>:<k>]`.

La liste des colonnes d'une séquence `<s>` découpée en `<k>` colonnes :

`<cols>=[<s>[<i>:<k>] for <i> in range(<k>)]`. Ceci utilise la notion de définition de liste par compréhension que nous verrons à la fin de la partie sur les listes.

### 5.1.3 Parcours

```
for <e> in <s>:
    <instructions pour e>
```

Deux types de séquences particulières : les chaînes de caractères et les listes

## 5.2 Chaînes de caractères

- Littérales : `'<texte>'` ou `"<texte>"`
- Multi-lignes : `'''<texte>'''` ou `"""<texte>"""`
- Les chaînes ne sont pas modifiables
- Les caractères sont des chaînes de longueur 1
- Caractère en position `<i>` :
  - consultation `<s>[<i>]`,
  - affectation/modification `<s>[<i>]=<valeur>`
- changer `<s>.lower()` les majuscules en minuscules, `<s>.upper()` les minuscules en majuscules, `<s>.swapcase()` les minuscules en majuscules et les majuscules en minuscules, `<s>.capitalize()` les initiales des mots en majuscules
- `<s>.islower()`, `<s>.isupper()`, `<s>.isalpha()`, `<s>.isdigit()`, `<s>.isalnum()` indique si la chaîne de caractères `<s>` est respectivement en minuscules, en majuscules, contient uniquement des lettres de l'alphabet, des chiffres, des lettres de l'alphabet et des chiffres
- `<s>.count(<texte>)` renvoie le nombre d'occurrences de la sous-chaîne `<texte>` dans la chaîne de caractères `<s>`
- `<s>.find(<texte>)[<i>:<j>]` renvoie la position de la sous-chaîne `<texte>` dans la chaîne de caractères `<s>` (à partir de la position `<i>` et jusqu'à la position `<j>`) et -1 si `<texte>` n'est pas contenu dans `<s>` (à la différence de la fonction `index` qui renvoie une erreur dans ce cas)
- `<s>.replace(<avant>,<après>)` renvoie une copie de `<s>` où chaque occurrence de `<avant>` a été remplacée par `<après>` (ne modifie pas `<s>`)

### 5.2.1 Conversions chaînes ↔ nombres

- Conversion d'un nombre `<n>` en chaîne de caractères : `str(<n>)`
- Conversion d'une chaîne de caractères `<s>` en nombre : `eval(<s>)`

$$123 \xrightleftharpoons[\text{eval}]{\text{str}} '123'$$

## 5.2.2 Caractères et code ASCII

**ord** (<c>) : le code ASCII associé à un caractère <c>

**chr** (<code>) : le caractère correspondant à un code ASCII

'A'  $\xleftrightarrow{\text{ord}}$  65  
 $\xleftrightarrow{\text{chr}}$

## 5.2.3 Remplacement caractère par caractère

Pour appliquer à une chaîne de caractères <s> une transformation caractère par caractère

`<s>.translate(str.maketrans(<orig>, <dest>))`

`maketrans` fabrique la table de traduction de chaque caractère de la chaîne <orig> au caractère à la même position de la chaîne <dest>;  
`translate` transforme la chaîne <s> en utilisant cette table de traduction.

Exemple pour le décalage :

```
alph='ABCDEFGHIJKLMNOPQRSTUVWXYZ'  
alph_dec='DEFGHIJKLMNOPQRSTUVWXYZABC'  
'SALUT'.translate(str.maketrans(alph, alph_dec))
```

ou substitution mono-alphabétique :

```
alph_subst='OULSGCZQIYFEBMPDJWXKVANHRT'  
'SALUT'.translate(str.maketrans(alph, alph_subst))
```

Pour décrire l'alphabet sans se fatiguer

```
import string  
alph = string.ascii_uppercase
```

et le décalage de <dec> appliqué à l'alphabet : `alph_dec=alph[<dec>:] + alph[:<dec>]`

## 5.3 Listes

- Constantes : [], ou [<élément1>, <élément2>, ...]
- Élément en position <indice> :
  - consultation <liste>[<indice>],
  - affectation/modification <liste>[<i>]=<valeur>
- Longueur : `len(<liste>)`
- Test d'appartenance <e> in <liste>
- Parcours : `for <e> in <liste>`
- Concaténation : +
- Évidemment on peut faire des listes de listes et accéder par exemple au <indice2>-ème élément du indice1-ème élément d'une liste <liste> : <liste>[<indice1>][<indice2>]

### 5.3.1 Méthodes

- <liste>.append(<e>) ajout d'un <e> à la <liste>
- <liste>.remove(<e>) suppression de la première occurrence d'<e> de la <liste>
- <liste>.pop(<indice>) suppression de l'élément en position <indice> de la <liste>
- <liste>.insert(<indice>, <e>) ajout de l'<e> en position <indice> dans la <liste>
- <liste>.reverse() inverse l'ordre des éléments de la liste <liste>
- <liste>.sort() trie la <liste> (en place)
- <liste>.index(<e>) renvoie l'indice de la première occurrence de l'<e> dans la <liste>
- <liste>.count(<e>) renvoie le nombre d'occurrences de l'<e> dans la <liste>

### 5.3.2 Fonctions

- `min(<liste>)`, `max(<liste>)`, `sum(<liste>)`
- `sorted(<liste>)` : renvoie la liste ordonnée (nouvelle liste)

### 5.3.3 Conversion chaîne de caractères ↔ liste de caractères

- <s>.split(<séparateur>) renvoie une liste en découpant <s> selon le <séparateur>
- <séparateur>.join(<liste>) renvoie une chaîne de caractères en concaténant les éléments de la <liste> avec le <séparateur>

### 5.3.4 Range

Des listes particulières :

- **range** (*<longueur>*) la liste des entiers de 0 à *<longueur>* exclus,
- **range** (*<début>*, *<fin>*) la liste des entiers de *<début>* à *<fin>* exclus,
- **range** (*<début>*, *<fin>*, *<pas>*) la liste des entiers de *<début>* à *<fin>* exclus en sautant de *<pas>* à chaque fois (*<début>*, *<début>*+*<pas>*, *<début>*+2*<pas>*,...)

## 6 Dictionnaires

C'est un outil très puissant mais nous nous servirons surtout du fait qu'on peut utiliser n'importe quel type pour les indices, y compris des caractères. Cela correspond à la notion classique de listes d'association.

### 6.1 Syntaxe

- Constantes : { }, ou { commande(*<clef1>*:*<valeur1>*,... ) }
- Valeur associée à la *<clef>* :
  - consultation *<dictionnaire>*[*<clef>*],
  - affectation/modification *<dictionnaire>*[*<clef>*]=*<valeur>*
- Test d'appartenance *<clef>* **in** *<dictionnaire>* ou *<clef>*, *<valeur>* **in** *<dictionnaire>*.items()
- Parcours : **for** *<clef>* **in** *<dictionnaire>* ou **for** *<clef>*, *<valeur>* **in** *<dictionnaire>*.items()

### 6.2 Exemple d'utilisation d'un dictionnaire : le calcul de fréquences dans une chaîne de caractères

```
D=dict()
for car in s:
    if car in D:
        D[car]=D[car]+1
    else:
        D[car]=1
```

## 7 Définition de listes, d'ensembles et de dictionnaires par compréhension

[*<fonction>*(*<variable>*) **for** *<variable>* **in** *<s>*]  
(resp. { *<fonction>*(*<variable>*) **for** *<variable>* **in** *<s>* })  
(resp. { *<variable>*:*<fonction>*(*<variable>*) **for** *<variable>* **in** *<s>* })  
crée la liste (resp. l'ensemble) (resp. le dictionnaire) des images par la fonction *<fonction>* de tous les éléments *<variable>* de la structure de données *<s>* (chaîne de caractères, liste, ensemble, dictionnaire, etc.)

## 8 Entrées-sorties

### 8.1 Clavier/écran

Lecture *<variable>*=**input** (*<message d'invite>*)

Écriture **print** (*<valeurs a afficher séparées par des virgules>*) les valeurs seront affichées séparées par des espaces et il y aura un saut de ligne après l'affichage.

Pour supprimer le saut de ligne on le demande explicitement par **print** (*<valeurs>*, end="")

On peut évidemment faire de l'affichage formaté comme en C notamment avec la notation

```
print ("...%<format>..." %..., <valeur a afficher>, ...)
```

à l'ancienne (obsolète) ou plus moderne

```
print ("...{<num>:<format>}..." .format(..., <valeur a afficher>, ...))
```

où *<format>* est ce qui suit le caractère % en C et *<valeur a afficher>* est placé à la position *<num>* dans la liste des valeurs à afficher. *<num>* peut être omis, dans ce cas on fait référence aux valeurs une seule fois et dans l'ordre où elles apparaissent.

## 8.2 Fichiers

Ouverture `<fichier>=open(<nom>, <mode>)` où `<mode>` est 'r' (Read), 'w' (Write) ou 'a' (Append).

Fermeture : `<fichier>.close()`

Parcours de toutes les lignes du fichier :

```
for <ligne> in <fichier>:
    <instructions à appliquer à la ligne>
```

Écriture : `<fichier>.write(<texte>)` écrit `<texte>` dans le `<fichier>`. Le fichier est créé s'il n'existe pas.

On peut faire de l'affichage formaté avec `write`, de la même façon qu'avec `print` dans l'affichage sur la sortie standard.

## 8.3 Ligne de commande

Dans la bibliothèque `sys`, la fonction `argv` fournit la liste des arguments. Par exemple pour parcourir et imprimer la liste des arguments :

```
import sys
for arg in sys.argv:
    print arg
```

## 8.4 Lire aussi bien dans un fichier dont on passe le nom en paramètre du programme que sur l'entrée standard

```
import sys

if len(sys.argv) == 2:
    fichier = sys.argv[1]
    f = open(fichier, "r")
else:
    f = sys.stdin
for ligne in f:
    ...
```

## 9 Utilisation de bibliothèques

Après `import <bibliothèque>` on peut utiliser la `<fonction>` de la `<bibliothèque>` en écrivant

`<bibliothèque>.<fonction>`

Après `from <bibliothèque> import <fonction>` ou `from <bibliothèque> import *` en écrivant directement `<fonction>`