Éléments de Python *

Nathalie Sznajder

Document LU2IN005

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]]

1 Préambule

Pour lancer l'interpréteur sur un fichier *<fichier>*, il faut utiliser la commande `python` *<fichier>*.
Attention ! En Python, l'indentation du code est indispensable (4 espaces ou une tabulation) pour délimiter les blocs d'instruction.

2 Affectation de variables

```
<variable> = <valeur>
```

3 Structures de contrôles

Les structures de contrôle permettent d'effectuer des séquences d'instruction de façon particulière : plusieurs fois (avec les boucles **`for`** et **`while`**) ou de façon conditionnelle (avec la structure de contrôle **`if`**), ou encore uniquement sur l'appel d'une *fonction*, si la structure de contrôle est une définition de fonction.
Elles se définissent de façon générale ainsi :

```
<mot-clef> [[ paramètres ]]:
    <instructions indentées (1 tabulation ou 4 caractères)>
```

Plus particulièrement, la définition d'une fonction se fait ainsi, où *<fonction>* est le nom donné à la fonction, et *<corps de la* représente la suite d'instructions qui sera effectué à l'appel de la fonction. La partie entre guillemets est la signature de la fonction, elle est décrite en annexe du projet.

```
def <fonction> ([[ paramètres ]]):
  """ type des paramètres -> type du résultat
      rend ...
  """
    <corps de la fonction>
```

Les instructions conditionnelles se font ainsi :

```
if <condition>:
    <instructions_vrai>
[[ elif <condition2>:
    <instructions2_vraie> ]]
[[ else:
    <instructions_faux> ]]
```

Les boucles effectuent une séquence d'instructions de façon répétée - tant que la condition *<condition>* est vraie, dans le cas de la boucle **`while`**, autant de fois qu'une variable *<variable>* parcourt un certain nombre de valeurs d'une *<structure>* dans le cas de la boucle **`for`**.

*Merci à Valérie Ménissier-Morain pour son document qui a très largement inspiré celui-ci

```
while <condition>:
    <instructions>

for <variable> in <structure à parcourir>:
    <instructions pour chaque élément>
```

Conditions
Valeurs `True`, `False`
Opérateurs **`not`**, **`and`**, **`or`**
Test d'égalité ==

4 Commentaires

Les commentaires sont des lignes ignorées par l'interpréteur.
`# <commentaire>`

5 Opérations arithmétiques

+, −, ⋆, / (division flottante), // (division entière), % (modulo), ⋆⋆ (puissance).

6 Séquences

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

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

6.1 Base

Opérations

- Test d'appartenance de l'élément *<e>* à 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>*`)`

Parcours

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

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

6.2 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>*`.isalphanum()` 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>*)

6.2.1 Conversions chaînes ↔ nombres

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

$$123 \underset{\textbf{eval}}{\overset{\textbf{str}}{\rightleftharpoons}} \texttt{'123'}$$

6.2.2 Caractères et code ASCII

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

$$\texttt{'A'} \underset{\textbf{chr}}{\overset{\textbf{ord}}{\rightleftharpoons}} 65$$

Pour décrire l'alphabet sans se fatiguer

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

6.3 Listes

— La liste vide est notée `[]`,
— Une liste de plusieurs éléments est notée : `[`*<élément1>*`,`*<élément2>*`,...]`
— élément en position *<indice>* :
  — accéder à l'élément *<indice>* d'une liste : *<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>*`]`

6.3.1 Méthodes

- `<liste>.append(<e>)` ajout d'un `<e>` à la fin de la `<liste>`
- `<liste>.remove(<e>)` suppression de la première occurrence d'`<e>` de la `<liste>`
- `<liste>.pop(<indice>)` suppression (et retour) de l'élément en position `<indice>` de la `<liste>`
- `<liste>.insert(<indice>,<e>)` ajout de l'élément `<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 `<e>` dans la `<liste>`

6.3.2 Fonctions

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

6.3.3 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>`

6.3.4 *Range*

Des listes particulières :

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

7 Ensembles

Un ensemble (de type **`set`**) est une collection non ordonnée d'éléments tous distincts. Comme ils sont non ordonnés, on ne peut pas accéder à un élément par son indice. On ne peut pas construire d'ensemble d'ensembles, mais on peut construire des listes d'ensembles.

- Pour construire un ensemble de plusieurs éléments : `{<élément1>, <élément2>,...}`
- Pour construire un ensemble à partir d'une liste `<liste>` : **`set`**`(<liste>)`. Les doublons de `<liste>` sont supprimés.
- Cardinalité d'`<ensemble>` : **`len`**`(<ensemble>)`.
- Test d'appartenance : `<e>` **`in`** `<ensemble>` et non appartenance `<e>` **`not in`** `<ensemble>`.
- Parcours : **`for`** `<e>` **`in`** `<ensemble>`.

7.1 Comparaisons

- `<ensemble1> <= <ensemble2>` teste si `<ensemble1>` est inclus dans `<ensemble2>`.
- `<ensemble1> < <ensemble2>` teste si `<ensemble1>` est strictement inclus dans `<ensemble2>`.
- `<ensemble1> == <ensemble2>` teste si `<ensemble1>` et `<ensemble2>` contiennent exactement les mêmes éléments.

7.2 Méthodes

- `<ensemble1>.isDisjoint(<ensemble2>)` renvoie `True` si `<ensemble1>` et `<ensemble2>` n'ont aucun élément en commun.
- `<ensemble1>.isSubset(<ensemble2>)` renvoie `True` si `<ensemble1>` est inclus dans `<ensemble2>`.
- `<ensemble1>.isSuperset(<ensemble2>)` renvoie `True` si `<ensemble2>` est inclus dans `<ensemble1>`.

— *<ensemble1>*`.union(`*<ensemble2>*`)` renvoie un nouvel ensemble égal à l'union des éléments de *<ensemble1>* et *<ensemble2>* (se généralise à plusieurs arguments).
— *<ensemble1>*`.intersection(`*<ensemble2>*`)` renvoie un nouvel ensemble égal à l'intersection des éléments de *<ensemble1>* et *<ensemble2>* (se généralise à plusieurs arguments).
— *<ensemble1>*`.difference(`*<ensemble2>*`)` renvoie un nouvel ensemble égal à l'ensemble des éléments de *<ensemble1>* n'appartenant pas à *<ensemble2>* (se généralise à plusieurs arguments).
— *<ensemble>*`.add(`*<élément>*`)` ajoute *<élément>* à *<ensemble>*.
— *<ensemble>*`.remove(`*<élément>*`)` supprime *<élément>* d'*<ensemble>*. Lève `KeyError` si *<élément>* n'est pas dans *<ensemble>*.
— *<ensemble>*`.discard(`*<élément>*`)` supprime *<élément>* d'*<ensemble>* s'il est présent.

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 évidememnt 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.

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>*