

Syntaxe : types structurés

1 Opérateurs sur les listes, tuples et chaînes

- `L+M` : concaténation des deux collections ;
- `L*n` : répétition `n` fois d'une liste `L` ;
- `x in liste` : renvoie `True` si l'élément `x` est présent dans `liste`, `False` sinon ;
- `x not in liste` : renvoie `True` si l'élément `x` n'est pas présent dans `liste`, `False` sinon.

2 Accès aux éléments et extraction d'une tranche de liste, tuple ou chaîne

1. `L[p]` renvoie l'élément d'indice `p` de `L`
2. `L[p:n]` renvoie une nouvelle liste constituée des éléments de `L` d'indice `p` inclus à `n` exclu
3. `L[p:n:pas]` renvoie une nouvelle liste constituée des éléments de `L` d'indice `p` inclus à `n` exclu, tous les `pas`
4. `L[:]` renvoie une nouvelle liste constituée de tous les éléments de `L`
5. `L[p:]` renvoie une nouvelle liste constituée de tous les éléments de `L` à partir de l'élément d'indice `p` inclus jusqu'à la fin
6. `L[:n]` renvoie une nouvelle liste constituée de tous les éléments de `L` depuis le premier jusqu'à l'élément d'indice `n` exclu
7. `L[:pas]` renvoie une nouvelle liste constituée des éléments de `L`, tous les `pas`

2.1 Parcours d'une liste, tuple ou chaîne

A partir des syntaxes d'accès à un élément de liste, il est possible de parcourir un après l'autre les éléments d'une liste. Deux approches sont possibles :

	<pre>L=[2,4,8,16,32,64,128] for i in range(len(L)): print(L[i])</pre>		<pre>L=[2,4,8,16,32,64,128] for x in L: print(x)</pre>
---	---	---	--

2.1.1 Définition de liste par compréhension

Sur cette base, il est possible de construire une liste « par compréhension » . C'est une méthode qui consiste à définir une liste à l'aide d'une expression dépendante d'une variable qui parcourt un itérable.

La syntaxe générale est à adapter suivant le contexte :

`[expression(i) for i in range(n)]` ou bien `[expression(x) for x in L]`

3 Fonctions et méthodes sur des listes

3.1 Fonctions

1. `len(liste)` : renvoie la longueur, c'est à dire le nombre d'éléments, d'une liste
2. `min(liste)` : renvoie le plus petit élément d'une liste (dans la mesure où ils sont comparables)
3. `max(liste)` : renvoie le plus grand élément d'une liste (dans la mesure où ils sont comparables)
4. `sum(liste)` : renvoie la somme de tous les éléments d'une liste (dans la mesure où ils sont additionnables)
5. `sorted(liste)` : renvoie une nouvelle liste des éléments de `liste` triés par ordre croissant
6. `del(liste[i])` : supprime le i^{eme} élément de la `liste`
7. `del(liste)` : supprime `liste` de la mémoire

3.2 Méthodes

1. `L.append(x)` : ajoute l'élément `x` à la fin de la liste `L`
2. `L.pop(i)` : retire le i^{eme} élément et renvoie sa valeur
3. `L.extend(T)` : concatène la liste `T` à la fin de la liste `L`
4. `L.index(x)` : renvoie le plus petit indice de l'élément `x` si `x` est présent. Sinon : erreur.
5. `L.remove(x)` : retire le premier élément `x` de la liste `L`, si `x` est présent. Sinon : erreur.
6. `L.insert(i,x)` : insère l'élément `x` à l'indice `i` dans `L`. Si cet indice dépasse la taille de la liste, l'élément est ajouté à la fin.
7. `L.count(x)` : renvoie le nombre de fois où l'élément `x` est présent dans la liste `L`
8. `L.reverse()` : renverse l'ordre des éléments de la liste `L`
9. `L.sort()` : trie les éléments de la liste `L` par ordre croissant
10. `L.clear()` : vide la liste `L` de tous ses éléments
11. `L.copy()` : renvoie une copie de la liste `L` (voir ci-dessous)

4 Fonctions sur les tuples et les chaînes de caractères

5 Méthodes sur les chaînes de caractères

Une chaîne de caractère est non modifiable. Les méthodes renvoient donc une nouvelle chaîne, qu'il faudra affecter à une variable. On ne présente ici que celles utiles dans le cadre de ce cours. Soit *S* une chaîne de caractère quelconque.

- *S.replace(old,new)* : renvoie une copie de la chaîne de caractères en remplaçant toutes les occurrences de la chaîne *old* par *new*.
- *S.split(sep)* : renvoie une liste de mots correspondante à *S* découpée selon le séparateur *sep*
- *S.join(liste)* : renvoie une chaîne de caractères formée des éléments de la liste, séparés entre eux par la chaîne *S*
- *S.isalpha()* : renvoie **True** si tous les caractères de *S* sont des lettres, **False** sinon.
- *S.isdigit()* : renvoie **True** si tous les caractères de *S* sont des chiffres, **False** sinon
- *S.upper()* : renvoie une chaîne en remplaçant les minuscules de *S* par des majuscules
- *S.lower()* : renvoie une chaîne en remplaçant les majuscules de *S* par des minuscules
- *S.find(sub)* : renvoie le plus petit indice de la chaîne de caractères *sub* recherchée dans la chaîne principale *S*. Cette fonction renvoie -1 si la recherche n'a pas abouti.
- *S.index(sub)* : renvoie l'indice de l'élément *x* si *x* est présent. Sinon : erreur.
- *S.count(sub)* : renvoie le nombre d'occurrences de la chaîne de caractères *sub* dans la chaîne principale *S*
- *S.lstrip(car)* : renvoie une copie de la chaîne *S* où les caractères *car* situé en début de liste ont été retirés

6 Comparaison

Pour comparer deux valeurs, python utilise l'ordre croissant. Pour comparer deux lettres, python utilise l'ordre ASCII.

Pour comparer des chaînes, des listes ou des tuples entre eux, Python utilise l'ordre lexicographique :

1. On compare la première lettre (ou premier élément) ;
2. Si elles sont différentes, le résultat de la comparaison de ces 2 lettres s'impose ;
3. Si elles sont égales, on analyse la lettre (ou l'élément) suivant ;
4. Si il n'y a plus d'élément dans une des 2 chaînes, alors celle-ci est la plus petite.

7 Dictionnaires

7.1 Manipulation de dictionnaires

Usage	Syntaxe
créer un dictionnaire vide	<code>d = {}</code>
créer un dictionnaire non vide	<code>d = {'a' : 12, 'droit' : 'au but', 42 : [1,2,3]}</code>
lire une valeur sur la base de sa clé	<code>d['a']</code>
insérer une paire clé-valeur	<code>d[True]=888</code>
modifier une valeur sur la base de sa clé	<code>d[True]=889</code>
supprimer un élément sur la base de sa clé	<code>del(d[42])</code>

7.2 Fonctions et méthodes sur les dictionnaires

- `len(d)` renvoie le nombre d'élément du dictionnaire ;
- `key in d` teste si la clé `key` est présente dans le dictionnaire `d` ;
- `d.keys()` renvoie la liste des clefs ;
- `d.values()` renvoie la liste des valeurs ;
- `d.items()` renvoie la liste des couples (clé,valeur) ;
- `d.copy()` renvoie une copie du dictionnaire `d`.

7.3 Parcours d'un dictionnaire

Affichage des clés	Exemple
<pre>for cle in dico.keys(): print(cle)</pre>	a droit True
<pre>for cle in dico: print(cle)</pre>	a droit True
Affichage des valeurs	Exemple
<pre>for cle in dico: print(dico[cle])</pre>	12 au but 889
<pre>for val in dico.values(): print(val)</pre>	12 au but 889
Affichage des clés et des valeurs	Exemple
<pre>for cle,val in dico.items(): print(cle,val)</pre>	a 12 droit au but True 889