Listes

Plutôt que, pour manipuler beaucoup d'objets, créer autant de variables que d'objets voulus, on peut les stocker dans une liste.

Une liste est un type Python (`list`) ; on peut définir une liste en donnant la liste de ses éléments, entre crochets, séparés par des virgules.

EXEMPLE

Les objets `[1,"a",1.2]` et `[]` sont des listes. La dernière s'appelle *liste vide*.

Comme pour les chaînes de caractères, on peut accéder aux éléments d'une liste ; si la liste s'appelle `mylist`, alors `mylist[n]` renvoie le n-ième élément de la liste (attention, les listes sont numérotées à partir de 0). De plus, `mylist[deb:fin]` renvoie les éléments de `mylist` aux places `deb`, `deb`+1,...,`deb+fin`-1. On peut aussi spécifier un pas si on ne veut prendre qu'un objet sur 2, sur 3, *etc.* avec `mylist[deb:fin:pas]`.

EXEMPLE

Soit `l = [1,2,3,4,5,6]`. Alors `l[3]` est 4, `l[2:4]` est `[3,4]`, `[1:5:2]` est `[2,4]`.

Comme pour les chaînes de caractèreson peut omettre un des paramètres : s'ils ne sont pas précisés, `deb` est le premier élément de la liste, `fin` est le dernier + 1, et `pas` vaut 1.

EXEMPLE

Avec la liste précédente, `l[:3]` est `[1,2,3]`, `l[2:]` est `[3,4,5,6]` et `[::2]` est `[1,3,5]`.

Les listes se comportant comme des ensembles, on pourra donc faire des boucles `for` sur les listes. La variable de boucle prend alors successivement toutes les valeurs de la liste.

Exercice 1. Écrire une fonction qui calcule le nombre d'éléments d'une liste.

Exercice 2. Écrire une fonction qui prend en paramètres une liste et un objet, et qui renvoir `True` si l'objet est dans la liste, et `False` sinon.

La modifier pour qu'elle renvoie aussi l'indice où se trouve l'élément s'il y est.

Exercice 3. Écrire une fonction qui calcule la somme des éléments d'une liste, puis la moyenne des éléments d'une liste.

Ces fonctions existent en fait déjà en Python : `len(l)` est la longueur de `l`, et `x in l` teste si `x` est dans `l`.

Les listes sont un type mutable : on peut modifier une liste avec les commandes :

- `l[k]=x` : remplace l'élément d'indice `k` par `x`
- `l[i:j]=liste` : remplace la sous-liste `l[i:j]` par la liste `[liste]`
- `del(l[i])` : supprime l'élément d'indice `i` (c'est équivalent à `l[i:i+1]=[]`
- `l.append(x)` : ajoute `x` à la fin de la liste (c'est équivalent à `l[len(l):]=[x]`)

Attention, pour gagner de l'espace mémoire, Python a tendance à ne pas dupliquer les listes ; lors d'une affectation du type `l2=l`, `l2` est vu comme un deuxième nom pour `l`, et pas une nouvelle liste. En particulier, si `l` est modifiée, `l2` aussi.

Essayez le code suivant pour le constater :

```
l=[1,2,3]
l2=l
l.append(4)
print(l)
print(l2)
```

Exercice 4. Écrire une fonction qui échange le premier et le dernier élément d'une liste.

Exercice 5. Écrire une fonction qui calcule le miroir d'une liste. (Par exemple, la liste miroir de [1,2,5,3,4] est [4,3,5,2,1]).

Exercice 6. Écrire une fonction qui prend en paramètre un entier n, et renvoie la liste des termes de la suite de Fibonacci d'indices de 0 à n.

On peut aussi définir les listes par compréhension, c'est-à-dire en fonction d'une autre liste. Pour cela, on utilise la syntaxe `[f(i) for i in l if b(i)]`, où `f` est une fonction, `l` une liste et `b(i)` un booléen (dépendant éventuellement de `i`).

On peut aussi faire une liste avec plusieurs autres listes, avec `f(x,y) for x in l for y in l' if b(x,y)]`.

Exercice 7. Quelle liste est construite en appelant `[i*i for i in range(0:12) if i%2==0]` ?

Exercice 8. Écrire une fonction créant la liste de toutes les sommes possibles entre deux entiers naturels inférieurs à 15.

Exercice 9. Écrire une fonction qui prend en paramètre une liste de chaînes de caractères, et qui renvoie la liste de la première lettre de chaque chaîne.

Exercice 10.
a) Écrire une fonction qui prend en paramètres deux entiers n et p, et qui génère une liste de n entiers aléatoires entre 0 et p.

b) Écrire une fonction qui prend en paramètre un entier p et une liste d'entiers entre 0 et p, et qui renvoie le nombre d'éléments de $[\![0,]\!]p$ qui ne sont pas dans la liste.

c) Dans cette question, on prend $n = 100$ et $p = 99$. On veut savoir combien, en moyenne, d'entiers ne sont pas présents dans une liste aléatoire.

Pour cela, calculer la moyenne du nombre d'éléments non présents dans un grand nombre de listes aléatoires (par exemple 1000).