Dans ce chapitre, on s'attarde sur des fonctionnalités de Python pour la manipulation de tableaux.

1 Voir les tableaux comme des listes

Jusqu'à maintenant, on manipulait les tableaux, dans l'idée, comme on manipule les tableaux dans la plupart des langages de programmation :

1. Création du tableau de taille donnée (`t = [0]*10`) ;
2. Modification des valeurs (`t[4] = 2`) ;
3. Accès aux valeurs (`x = t[2]`).

Il existe cependant d'autres manières de créer des tableaux, dont la première repose sur la méthode `append`. Tester le programme suivant :

```
t = []
t.append(2)
t.append(4)
t.append(8)
```

On constate que la commande `t.append(x)` ajoute à la fin du tableau `t` la valeur `x`. Attention cependant, en utilisant cette commande, la longueur du tableau est modifiée !

2 Itérer sur les éléments d'un tableau

Pour parcourir un tableau, la manière habituelle (et commune dans l'idée aux autres langages) est la suivante :

```
for i in range(len(t)):
    print(t[i])
```

Tester le programme suivant :

```
for x in t:
    print(x)
```

On voit que ce programme produit exactement le même affichage. La déférence entre les deux boucle est l'**itérateur**. Dans le premier programme, il s'agissait de `i` l'**indice** de la boucle valant successivement 0, 1, 2 etc. alors que dans le deuxième programme, il s'agît de `x` prenant dans l'ordre toutes les **valeurs** du tableau `t[0]`, `t[1]`, `t[2]` etc.

Cette écriture est plus rapide et plus intuitive à utiliser (on peut y lire en français « pour tout x de t »). Elle ne sera cependant pas applicable si :

— on veut parcourir le tableau dans un ordre particulier ;

— de manière générale dès qu'on a besoin des positions des éléments du tableau.

3 Construction des tableaux par compréhension

Dernière méthode pour créer des tableaux, la compréhension. Il s'agit de décrire en une seule commande la taille et le contenu d'un tableau. Tester le programme suivant :

```
t = [3*i+2 for i in range(10)]
print(t)
```

On voit que le tableau `t` contient les valeurs de la forme $3i + 1$ pour tout i entre 0 et 9. Tester le programme suivant :

```
t = [3*i+2 for i in range(10) if i%2 == 0]
print(t)
```

Ici on a réduit les i pour lesquels on insère les valeurs dans le tableaux aux seuls nombre pairs entre 0 et 9.

Ici on a itéré sur un indice dans la construction par compréhension. On peut cependant choisir d'itérer sur les valeurs d'un autre tableau. Exemple à tester :

```
t1 = [4,5,6]
t2 = [x*x for x in t]
print(t2)
```