Chap. 2 Types structurés Thème 2

Tableaux

Sommaire du chapitre 2

- ▷ Types simples et types structurés
- ▷ p-uplets (ou tuples)
- ▷ **Tableaux**
- ▷ Chaînes de caractères
- ▷ Dictionnaires
- ▷ Valeurs et références

Un **tableau** est une collection ordonnée d'éléments de ***même*** type.

Remarques

- En Python, un **tableau** est appelé **liste**. Il est de type `list`.
- Contrairement au tuple, le tableau est modifiable par affectation, c'est-à-dire que le contenu d'un tableau peut changer après sa création. On dit qu'il est **muable** (« *mutable* » en anglais).

I Création d'un tableau

En langage Python, les tableaux sont notés entre **crochets** en listant les termes séparés par des virgules.

Exemples

Un tableau peut servir par exemple à représenter une suite de nombres, ou la suite des mots d'un texte.

Remarques

Un tableau vide est noté `[ ]`.

II Indexation des éléments du tableau

Pour un tableau de `n` éléments, noté `t`, les termes sont **indexés** de `0` à `n-1`.
L'accès aux éléments du tableau se fait par la **notation indexée** `t[i]`, où `i` est appelé l'**indice** de l'élément `t[i]`. Le premier élément du tableau est à l'indice 0, le nombre d'éléments d'un tableau est donné par `len(t)` :

Exemples

```
1 premiers = [1, 2, 3, 5, 7, 11, 13, 17, 19, 23]
2 print('le cinquième nombre premier est', premiers[4]) # 1er
      indice est 0
3 for i in range(len(premiers)) : # i va de 0 à len(premiers)-1
4     print('indice', i, '->', premiers [i])
```

Remarques

Si l'on tente d'accéder à un élément avec un indice en dehors du tableau, on obtient une erreur :

```
>>> premiers[20]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

III Énumération des éléments d'un tableau

On peut **énumérer** les éléments d'un tableau comme indiqué à la ligne 3 de l'exemple précédent, avec un indice qui va de `0` à sa longueur (`for i in range(len(premiers))`). Si l'on n'a pas besoin de la valeur de l'indice, on peut aussi énumérer les éléments directement avec l'instruction `for e in t`. Dans ce cas, la variable de boucle prend les valeurs des éléments du tableau :

```
1 for n in premiers : # n vaut les éléments successifs du tableau
2     print(n, 'est un nombre premier')
```

Comme pour les chaînes de caractères, on peut créer de nouveaux tableaux à partir de tableaux existants par **concaténation** (opérateur `+`) et par **répétition** (opérateur `*`) :

```
1 pairs = [0, 2, 4, 6, 8]
2 impairs = [1, 3, 5, 7, 9]
3 nombres = pairs + impairs # nombres vaut [0,2,4,6,8,1,3,5,7,9]
4 couleurs = ['trefle', 'pique'] + ['coeur', 'carreau']
5 zeros = [0] * 100          # tableau de 100 éléments qui valent tous 0
6 cycle = [1, 2, 3] * 10     # 10 fois la séquence 1, 2, 3
```

Remarques

On peut également utiliser l'énumération, la concaténation et la répétition avec les p-uplets.

IV Modification du tableau par affectation

Contrairement aux tuples, les tableaux sont des types **muables** : on peut modifier leur contenu. Pour modifier un élément de tableau, on utilise la notation indexée à gauche d'une affectation :

Exemples

```
t = [1, 2, 3]
for i in range(len(t)) :
    t[i] = t[i] * 2       # doubler la valeur du i-ème élément
# t vaut [2, 4, 6]
```

Remarques

Les éléments d'un tableau doivent être d'un **même** type, même si Python n'impose pas cette contrainte. Ce type peut être un type de base, comme dans les exemples ci-dessus, mais aussi un type construit, comme un tuple ou un tableau, ce qui permet de représenter des données complexes.

Par exemple, un tableau peut contenir un ensemble de coordonnées `(x, y)`, ou l'ensemble des cartes d'un jeu de cartes, ou une main d'un joueur.
Inversement, un tuple peut contenir un tableau comme élément. Par exemple on peut représenter le joueur d'un jeu de cartes par un tuple constitué de son nom, du tableau de ses cartes, et de son score :

```
# tableau de tuples
cartes = [("as","pique"), (10,"trefle"), ("valet","trefle"), (10,
    "coeur"), (5,"pique")]
#tuple avec le tableau 'cartes' comme second élément
joueur = ("Joe", cartes, 120)
```

Remarques

Dans le langage Python, les tableaux sont mis en œuvre par un type construit plus général appelé **list**. Ce type permet d'ajouter et de retirer des éléments d'une liste, alors que dans beaucoup de langages de programmation la taille d'un tableau est fixée à sa création.

Pour ajouter et retirer des éléments d'une liste, on utilise des **méthodes**. Une méthode est une fonction associée à un type d'objet. On l'appelle en utilisant la **notation pointée** `objet.methode(parametres)`. Les méthodes sont utilisées en particulier avec les types muables pour les fonctions qui modifient la valeur de l'objet. Les méthodes qui permettent de modifier le contenu d'une liste sont les suivantes :

```
premiers.append(13)  # ajoute le nombre 13 à la fin de la liste
mots.insert(1, "et") # insère "et" en 2è position (on démarre à 0)
p = points.pop(2)    # retire l'élément à l'indice 2 et retourne sa valeur
points.clear()       # retire tous les éléments de la liste
```

```
semaine =['lundi', 'mercredi', 'jeudi', 'vendredi', 'samedi']
semaine.append('dimanche')  # ajoute la chaîne 'dimanche' à la fin de la liste
semaine.append('lundi')
semaine.insert(1, "mardi")  # insère "mardi" en 2è position (on démarre à 0)
semaine.remove('lundi')     # supprime de la liste le premier élément dont la valeur est "lundi"
p = semaine.pop(2)          # retire l'élément à l'indice 2 et retourne sa valeur
semaine.clear()             # retire tous les éléments de la liste
```