Python

Tuples - Tableaux - Listes

Yannick CHISTEL

Lycée Dumont d'Urville - CAEN

Décembre 2019

Données multiples

Une variable contient une valeur, modifiable à tout moment dans le programme.
Il est parfois nécessaire de créer de très nombreuses variables du même type. Par exemple, les jours de la semaine, les mois de l'année, les dates...

- j1="lundi", j2="mardi", j3="mercredi", ...
- m1="janvier", m2="février", m3="mars", ...
- d1=1, d2=2, ..., d31=31

Les tableaux sont une structure en informatique qui permettent de stocker toutes les valeurs dans une même variable.

- jours="lundi", "mardi", "mercredi", ..., "dimanche"
- mois="janvier", "février", "mars", ..., "décembre"
- dates=1, 2, 3, ..., 31

En **python**, il en existe 2 types de tableaux : les **tuples** et les **listes**.

Tableaux python

Tuple et Liste

Un **tuple** ou une **liste** regroupe plusieurs valeurs dans une **même variable**.
Un **tuple** se note entre **parenthèses** contenant les valeurs séparées par des virgules.
Une **liste** se note entre **crochets** contenant les valeurs séparées par des virgules.

Exemple

Les différentes variables utilisées pour les jours de la semaine peuvent être rassemblées dans une même variable en **tuple** ou en **liste**.

1. Pour un **tuple** (avec des parenthèses) :
   jours=("lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche")
2. Pour une **liste** (avec des crochets) :
   jours=["lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche"]

Remarque importante

1. Un **tuple** n'est pas modifiable. On ne peut pas modifier les valeurs.
   Un tuple n'est pas **mutable**.
2. Une **liste** est modifiable. On peut modifier les valeurs.
   Une liste est **mutable**.

Tableaux python

Création d'un tableau

Pour créer un **tuple** ou une **liste**, il suffit d'écrire les valeurs séparées par des virgules entre **parenthèses** pour un **tuple** et entre **crochets** pour une **liste**.

Exemple

Les mois de l'année (de type "string") :
Sous forme de tuple : **mois**=("janvier", "février", ... , "novembre", "décembre")
Sous forme de liste : **mois**=["janvier", "février", ... , "novembre", "décembre"]

Longueur d'un tableau

Pour accéder à une valeur d'un tuple ou d'une liste, on utilise son indice (index en anglais). Il est donc important de connaître la **longueur** d'un tableau, c'est à dire le nombre de ses éléments.

En **python**, la longueur d'un tuple ou d'une liste est donnée avec la fonction **len**.

Exemple

len(mois) renvoie 12, donc le tableau contient 12 éléments **indicés** de 0 à 11.
mois[**0**] vaut "janvier", **mois**[**1**] vaut "février", ... , **mois**[**11**] vaut "décembre"

Parcourir un tableau

Méthode

On peut parcourir un tableau et donc récupérer ses valeurs en itérant sur les indices avec une boucle. Soit tab une variable de type tableau :

1. Avec une boucle **for** :
   for i **in range**(**len**(tab)) :
   ... tab[i] ...
2. Avec une boucle **while** :
   while k $<$ **len**(tab) :
   ... tab[k] ...
   k = k+1

Exemple

Afficher les mois de l'année avec une boucle for :

```
for i in range(len(mois)) :
    print(mois[i],end="-")
```

qui affichera :
janvier-février-mars-avril-mai-juin-juillet-septembre-octobre-novembre-décembre-

Valeurs d'une liste

Méthode 1

On peut modifier les valeurs d'une liste en affectant directement les nouvelles valeurs repérées par leurs indices.

Exemple

Soit la liste des jours de la semaine dans la variable jours :

jours = ("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche")

Si on souhaite remplacer les valeurs de la liste jours par des valeurs en anglais :
jours[0]="monday", jours[1]="tuesday", etc

Méthode 2

Si on souhaite construire un tableau de très grande taille, python permet une construction rapide avec des valeurs initiales :

variable tableau = [valeur initiale] * dimension du tableau

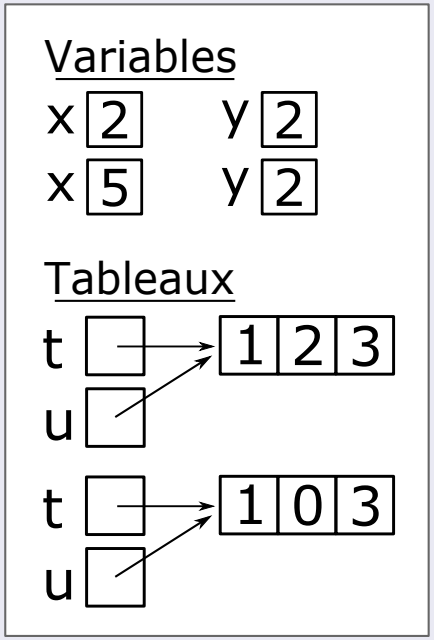
Exemple

Construire une liste de 100 nombres : **nombres** = [0] * 100
On a un tableau rempli de 100 zéros : [0,0,0,...0,0]

Tableaux et variables

Variables

1. Soit x et y deux variables.
   On affecte les variables : $x = 2$ et $y = x$. On a 2 espaces mémoires distincts qui ont la même valeur.
   Si maintenant on a $x = 5$ alors y vaut toujours 2.
2. Soit la variable tableau t contenant les valeurs 1, 2, 3.
   Si on déclare une seconde variable tableau u en lui affectant la valeur de la variable t : $u = t$.
   Dans ce cas, les variables t et u désignent le même tableau !
   Si on modifie la valeur $u[1] = 0$ alors $t[1] = 0$.



Remarque

Cette affectation par égalité de tableaux lie les deux tableaux. Mais à tout moment on peut délier les tableaux en affectant un nouveau tableau à l'une des variables.
Par exemple : $u = [7, 8, 9]$ du coup on redéfinit un nouveau tableau ! u et t ne sont plus liés.

Tableaux et fonctions

Présentation

1. Une fonction accepte un tableau comme paramètre. Dans ce cas, la fonction peut modifier le tableau.
2. Une fonction peut retourner un tableau de valeurs.

Exemple

```
def carres(n) :
    t=[0]*n
    for i in range(n) :
        t[i]=(i+1)**2
    return t
```

```
def quadro(t) :
    for i in range(len(t)) :
        t[i]=t[i]**2
    return t
```

1. Si une variable **c = carres(5)**, celle-ci a pour valeur le tableau [1,4,9,16,25].
2. Si on appelle la fonction **quadro(c)**, la variable c a pour valeur le tableau [1, 16, 81, 256, 625].

Attention

Dans les fonctions, les tableaux sont directement modifiés et cela nécessite de la réflexion et des tests.

Python

Seconde partie :

Utilisation avancée des tableaux

Parcourir un tableau

Tableaux et indices négatifs

On peut accéder aux derniers éléments d'un tableau avec des indices négatifs :

- liste[0] : première valeur du tableau
- liste[-1] : dernière valeur du tableau
- liste[-2] : avant-dernière valeur du tableau

Itérer sur les éléments

On peut accéder à un élément d'une liste avec une boucle for et la syntaxe suivante :

```
for e in liste :
    instruction avec e
```

Exemple

Soit t=[1,2,3,4,5,6]

1. **print(t[-1])** affiche la valeur 6
2. **print(t[-2])** affiche la valeur 5
3. **for** k **in** t :
   print(k, end=' ') affiche 1 2 3 4 5 6

Créer un tableau par **compréhension**

Création d'un tableau

La construction d'un tableau par compréhension introduit la boucle for à l'intérieur des crochets du tableau à construire. La syntaxe est de la forme :

[valeur **for** i **in range**(dimension du tableau)]

Exemple

1. Construire par compréhension une liste ordonnée de nombres :
   a) [i **for** i in range(5)] construit le tableau [0, 1, 2, 3, 4]
   b) [i**2 **for** i in range(5)] construit le tableau [0, 1, 4, 9, 16]
   c) [2*i+1 **for** i in range(5)] construit le tableau [1, 3, 5, 7, 9]
2. Construire un tableau à partir des valeurs d'un autre tableau :
   t=[2,3,5,7,11,13]
   p=[x**2 **for** x in t]
   Le tableau p a pour valeur [4, 9, 25, 49, 121, 169]
3. Construire un tableau dont les valeurs sont des chaines de caractères :
   direction=['nord','sud','est','ouest']
   DIRECTION=[e.upper() for e in direction]
   Le tableau DIRECTION a pour valeur ['NORD', 'SUD', 'EST', 'OUEST']

Copier un tableau : **list**

Méthode

On a vu que la copie d'un tableau ne se fait pas par affectation puisque les variables vont au final désigner le même tableau.
Pour déclarer une nouvelle variable en lui affectant les valeurs d'un tableau existant, on utilise la fonction **list** avec la syntaxe :
tableau2 = **list**(tableau1)

Exemple

t=[10,20,30,40,50]
u=**list**(t)
print(u) affiche [10,20,30,40,50] *le tableau est copié dans la variable u.*
for i **in range**(5) :
    u[i]=u[i]/10 *ici, chaque valeur de u est divisée par 10*
print(t) afffiche [10,20,30,40,50] *le tableau t n'a pas changé*
print(u) afffiche [1,2,3,4,5] *le tableau u a ses valeurs divisées par 10*

Modifier les tableaux

Ajouter des valeurs

Un tableau est de dimension fixée. Mais, en python, il est possible d'agrandir un tableau en ajoutant des valeurs. Deux méthodes sont possibles :

1. Par concaténation de deux tableaux existants avec l'opérateur + ;
2. En utilisant la fonction **append** qui permet d'ajouter une valeur en fin de tableau.

Exemple

1. Par concaténation de tableaux :
   t=[0,1,2]
   u=[3,4,5]
   s=t+u *le tableau u a pour valeur [0,1,2,3,4,5]*
2. Avec la fonction **append** : t=[0,1,2]
   t.append(3) *le tableau t a pour valeur [0,1,2,3]*
   t.append(4) *le tableau t a pour valeur [0,1,2,3,4]*
3. Avec la fonction **append** et une boucle **for** : t=[0,1,2]
   u=[3,4,5]
   for e in u :
   t.append(e) *le tableau t a pour valeur [0, 1, 2, 3, 4, 5]*

Des fonctions sur les tableaux

Présentation

Il existe des fonctions que l'on peut appliquer sur un tableau, pour ajouter des éléments (append) et pour connaître le nombre d'éléments (len). En voici d'autres fonctions (liste non exhaustive) :

- La fonction copy qui recopie les valeurs d'un tableau dans un autre ;
- La fonction pop qui supprime la dernière valeur d'un tableau ;
- La fonction insert qui insère une valeur pour un indice donné du tableau ;
- La fonction remove qui suprrime une valeur pour un indice donné du tableau ;
- La fonction sort trie le tableau dans l'ordre croissant.

Exemple

Soit un tableau : t=[0,1,2,3]

1. u=t.copy() *le tableau u est créé et a les mêmes valeurs que t*
2. u.pop() *la dernière valeur du tableau u est supprimée ; u vaut [0,1,2]*
3. u.insert(1,4) *la valeur 4 est insérée à l'indice 1 ; u vaut [0,4,1,2]*
4. u.remove(1) *si elle existe, la valeur indiquée est supprimée ; u vaut [0,4,2]*
5. u.sort() *le tableau u est trié ; u vaut [0,2,4]*

Tableau de tableaux

Présentation

Un tableau peut contenir tout type de valeurs : des entiers (int), des chaines de caratères (string), des nombres réels (float) et aussi des tableaux !

Pour accéder à une valeur de ce tableau, on utilise un premier indice pour sélectionner le tableau où se trouve la valeur puis un second indice pour obtenir la valeur dans le tableau sélectionné. Les indices sont notés entre crochets.

Exemple

1. t=[[4,5],[6,7],[8,9]]
   Le tableau t contient 3 tableaux de dimension 2 ; t est un tableau de dimension 3×2 ;
   le tableau [4,5] est d'indice 0, le tableau [6,7] d'indice 1 et le tableau [8,9] d'indice 2 ;
   les valeurs ont pour indice 0 et 1 pour chacun des trois tableaux ;
   print(t[1][0]) *tableau d'indice 1, valeur d'indice 0 ; affiche la valeur 6*
2. On initialise un tableau de tableaux par compréhension :
   t=[[0]*3 for i in range(3)] le tableau t vaut [[0,0,0], [0,0,0], [0,0,0]]