Algorithmique et programmation 1

Structures de données (1) les tableaux

L1 M-I-SPI – Université de Lorraine
Marie Duflot-Kremer
avec l'aide des collègues de Nancy et Metz

Transparents disponibles sur la plateforme de cours en ligne

Types de données

Pour le moment, pour stocker des informations, on dispose de :

-
-
-
-

... mais cela ne nous suffit pas.

On veut plus

- Un même objet pour stocker plusieurs données,
- soit plein de données de même nature
  - notes des étudiants,
  - durée de morceaux de musique,
  - difficulté de chaque sujet de TP
- soit quelques données de nature différente
  - Pour une personne : prénom, genre, taille
  - Pour un téléphone : modèle, capacité mémoire, taille écran

⇝ On va étudier deux sortes d'objets pour faire cela

Tableau - à quoi ça sert

- Stocker un n-uplet de valeurs de même nature,
- accéder facilement à ces valeurs :
  - pouvoir lire chaque donnée directement,
    - sans parcourir les autres,
    - un indice pour désigner chaque donnée,
  - pouvoir modifier chaque donnée indépendamment,
    - sans parcourir les autres,
    - sans tout réécrire,
- parcourir/manipuler les valeurs avec des boucles.

Tableau - nouveau type

Variables de type tableau à déclarer comme les autres.
Il faut dire :

- que c'est un tableau,
- le nombre d'éléments (=dimension),
- le type de ces éléments (le même pour tous !).

Exemple

Variables
distances : tableau de réels [12]
scores : tableau d'entiers [30]
référendum : tableau de booléens [100]

Pourquoi toutes ces infos ?

Dire que c'est un tableau sert à...

Dire le nombre d'éléments sert à...

Dire le type de chaque élément sert à...

Taille totale

Pour un tableau de n cases contenant chacune une donnée codée sur k octets on a besoin de octets.

Accès aux données

On peut accéder aux données en lecture :

- afficher la valeur de la case i du tableau tab,
  afficher(tab[i])
- lire la valeur de case la case i du tableau tab (et la ranger dans la variable max),
  max ← tab[i]

... mais aussi en écriture :

- initialiser tout un tableau
  tab ← [2, 1, 4, 6, 10]
- mettre le contenu de la variable max dans la case i du tableau tab.
  tab[i] ← max

Attention

Les cases sont numérotées à partir de 0 !

Exercice (1)

Écrire un algorithme qui déclare un tableau de 10 entiers, le remplit avec des valeurs aléatoires entre 1 et 99 puis calcule la somme des valeurs de chaque case.

```
# N'oubliez pas les commentaires du début
```

Exercice (2)

Ecrire une fonction qui calcule l'indice du tableau à partir duquel la somme dépasse strictement un certain seuil(donné en paramètre). La fonction retourne -1 si la somme ne dépasse pas.

Nombre d'éléments variable

Le problème

On veut un algorithme qui demande un nombre de notes à l'utilisateur puis les lui fait saisir et les stocke dans un tableau.

- Combien d'éléments à stocker ?
  - ⇝ on ne sait pas précisément,
  - ⇝ dépend du choix de l'utilisateur,
  - ⇝ il faut pourtant fixer la dimension !
- Comment faire ?
  - ⇝ on décide un nombre de notes maximum,
  - ⇝ on le prend comme dimension du tableau,
  - ⇝ on définit une variable pour le nombre effectif de notes.

Nombre d'éléments variable (2)

Le problème

On veut un algorithme qui demande un nombre de notes à l'utilisateur puis les lui fait saisir et les stocke dans un tableau.

```
# Algorithme saisie de notes
Variables
    nbnotes, index : entier
    tabnotes : tableau d'entiers [100]
Début
Fin
```

- dimension fixe : ici
- nombre effectif de cases variable : ici

Tableau à deux dimensions

- déclaration : quasi comme pour un tableau à une dimension

```
Variables
    distances : tableau d'entiers [8][10]
# On donne le nombre de lignes puis le nombre de colonnes
```

- lecture /mise à jour

```
    # d'abord le numéro de ligne puis le numéro de colonne
distance[2][3] ← 120
parcours ← distance[7][6]
```

- (en fait on a créé un tableau de... tableaux)

Tableaux à deux dimensions (2)

Écrire un algorithme qui déclare un tableau de 5 fois 7 entiers et l'initialise de telle sorte que la case ligne l colonne c contienne la valeur $l \times c$.

Des tableaux en Python ?

Le type tableau, avec

- taille fixe, non modifiable
- objets tous de même type

... **n'existe pas** en Python standard, **mais**

- il existe un type approchant : list
- on peut l'utiliser pour représenter/manipuler des tableaux

⇝ c'est ce qu'on va faire dans ce cours

Déclaration et initialisation

- Déclaration (dans l'en-tête)

```
# Variables
# T : list of int [10]
# Tbis : list of float [8]
```

- initialisation (quelque part dans le programme)

```
T = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Tbis = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

ou bien

```
T =[0]*10
Tbis =[1.0]*8
```

  - première solution : personnalisable

```
T =[0, 2, 4, 6, 8, 10, 9, 7, 5, 3]
```

  - deuxième solution : plus rapide

Déclaration et initialisation - types

Attention

Il faut initialiser le tableau avec des valeurs du bon type, par exemple :

- 0, -1 ou 10 pour `int`
- `True` ou `False` pour `bool`
- 0.0, -12.5 ou 1.0 pour `float`
- `None` (= "rien"), accepté pour tous les types

Déclaration et initialisation - rappels

En Python ailleurs

- l'ordinateur se moque de la déclaration des variables
- variable réellement créée au moment où elle est initialisée
- on peut mettre des valeurs de n'importe quel type n'importe où

En Python en AP1

- la déclaration des variables, en commentaires, est obligatoire
- il faut respecter le type annoncé pour les données

Copie de tableau

```
>>> tab = [4, 3, 2, 1, 0]
>>> print(tab)
[4, 3, 2, 1, 0]
>>> tabcopie = tab
>>> print(tabcopie)
[4, 3, 2, 1, 0]
>>> tabcopie[2] = 3
>>> print(tabcopie)

>>> print(tab)
```

Ne fonctionne pas

tabcopie n'est pas un nouveau tableau, la commande tabcopie = tab en fait un alias de tab. Ils désignent les mêmes données.

Copie de tableau - solutions

- On peut recopier un à un les éléments de tab [1]...

```
tab = [4, 3, 2, 1, 0]
tabcopie = [0]*5  # Initialisation
for indice in range(5):
    tabcopie[indice]=tab[indice]
tabcopie[2] = 3
print(tabcopie)
print(tab)
```

Ce programme va afficher :

- ... ou utiliser une méthode appelée copy()

```
tab = [4, 3, 2, 1, 0]
tabcopie = tab.copy()
tabcopie[2] = 3
print(tabcopie)
print(tab)
```

Ce programme va afficher :

1. sauf si ce sont eux-mêmes des tableaux

Et à deux dimensions

- Déclaration :

```
# tab : list of int [12][6]
```

- Initialisation :
  - un peu plus compliquée qu'à une dimension
  - tab=[0]*6*12 ne fonctionne pas (une seule ligne)
  - tab=[[0]*6]*12 non plus (problème d'alias)
  - on doit utiliser deux boucles

```
tab = [[0 for i in range(6)] for j in range(12)]
                        # attention à déclarer i et j
```

- Manipulation :

```
tab[4][1] = 2*tab[2][3]+6
```

C'est plus facile en Python

Afficher le contenu du tableau T de dimension n

en algo :

```
Pour i allant de 0 à n-1 Faire
    afficher(T[i],", ")                    # pas parfait
Finpour
```

en python :

```
print(T)
```

Parcourir toutes les cases en lisant leur valeur

Calcul de la somme de tous les éléments :

en algo :

en python :

Parcourir un tableau

Attention

for elem in T ne permet que de **lire** les éléments, pas de les modifier.

```
T = [5, 6, 7, 8]
print ("Tableau avant : ", T)
for elem in T:
    elem = elem + 1
print ("Tableau après : ", T)
```

affiche :

Solution

```
for i in range(4):
    T[i] = T[i]+1
```


Sources

G.Swinnen, *Apprendre à programmer avec Python 3 (3ème édition*, Eyrolles, 2012, Disponible en ligne à l'adresse `http://inforef.be/swi/python.htm`.