Langages de programmation

Cours 4

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-prog

Plan

- exécution et mise au point (*debugging*) avec VScode
- révision des programmes du cours 3
- alias et données modifiables
- gestion de la mémoire

dès maintenant: télécharger Python 3 en `http://www.python.org`

Tableaux multi-dimensionnels

- création d'une matrice pleine de zéros

```
def new_matrix (m, n, v) :
    a = []; z = [v]*n
    for i in range(m): a.append (z.copy())
    return a
```

à comprendre plus tard !

- carré magique

```
def magique (n) :
    a = new_matrix (n, n)
    i = n - 1
    j = n // 2
    for k in range (n*n) :
        a[i][j] = k+1
        if (k+1) % n == 0 :
            i = (i - 1) % n
        else :
            i = (i + 1) % n
            j = (j + 1) % n
    return a
```

n impair

```
>>> print_matrix (magique(3))

4 9 2
3 5 7
8 1 6
```

somme 15 sur lignes et colonnes

somme 15 sur les 2 diagonales

```
>>> print_matrix (magique(7))

 22  31  40  49   2  11  20
 21  23  32  41  43   3  12
 13  15  24  33  42  44   4
  5  14  16  25  34  36  45
 46   6   8  17  26  35  37
 38  47   7   9  18  27  29
 30  39  48   1  10  19  28
```

Recap

- mots clés en Python (déjà vus en rouge)

```
>>> help()
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

Tableau

- valeur d'un tableau (*list*)

```
>>> a = [1, 3, 4, 2, 3, 5, 9]

>>> b = a
>>> b
[1, 3, 4, 2, 3, 5, 9]

>>> b[3] = 42
>>> b
[1, 3, 4, 42, 3, 5, 9]

>>> a
[1, 3, 4, 42, 3, 5, 9]
```

a a été modifié ! !

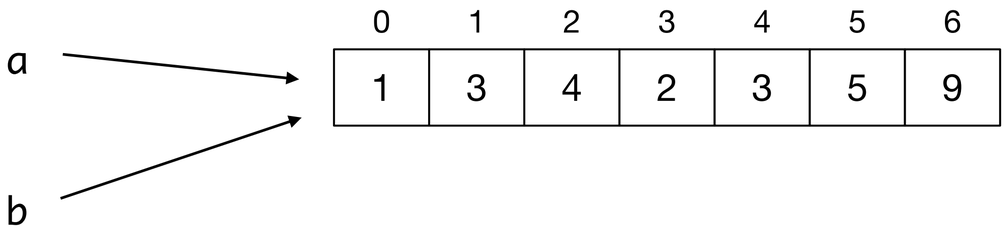
- pourquoi ?

Tableau

- alias

```
>>> a = [1, 3, 4, 2, 3, 5, 9]

>>> b = a
>>> b
[1, 3, 4, 2, 3, 5, 9]

>>> b[3] = 42
>>> b
[1, 3, 4, 42, 3, 5, 9]

>>> a
[1, 3, 4, 42, 3, 5, 9]
```

a et b sont des alias



alias et données modifiables = DANGER !

- la valeur de a est son adresse mémoire

```
>>> id (a)
4342380928
>>> id (b)
4342380928

>>> a == b
True
>>> a is b
True
```

```
>>> c = [1, 3, 4, 2, 3, 5, 9]
>>> id (c)
4342381056

>>> a == c
True
>>> a is c
False
```

Tableau

- valeur d'un tableau (*list*)

```
>>> a = [1, 3, 4, 2, 3, 5, 9]

>>> b = a
>>> b
[1, 3, 4, 2, 3, 5, 9]

>>> b [3] = 42
>>> b
[1, 3, 4, 42, 3, 5, 9]

>>> a
[1, 3, 4, 42, 3, 5, 9]
```

a a été modifié ! !

- la valeur de a est une **référence** vers l'objet qui la représente

- la valeur de a est grosso modo l'**adresse** de ce tableau

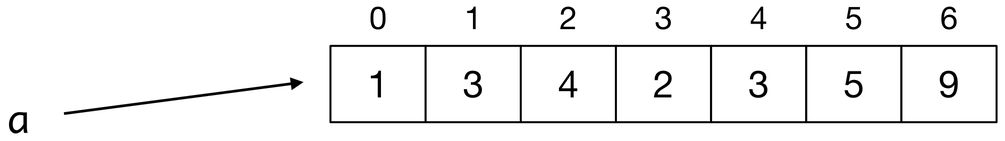
Tableau

- création de tableau

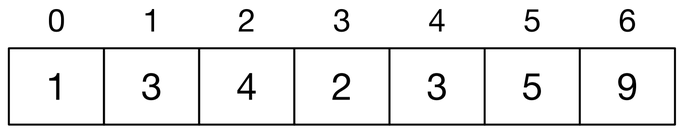
```
def copy_list (a) :
    r = [ ]
    for x in a :
      r = r + [x]
    return r
```

```
def new_list (n, v) :
    r = [ ]
    for i in range (n) :
        r = r + [v]
    return r
```

```
def new_matrix (m, n, v) :
    r = [ ]
    for i in range(m):
      r = r + [new_list (n, v)]
    return r
```




```
b = copy_list (a)
```




```
c = new_matrix (2, 3, 0)
```

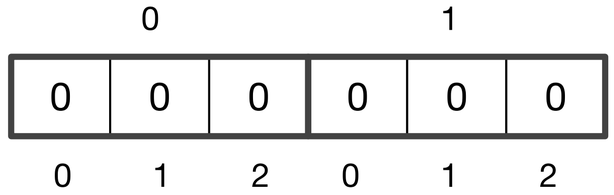
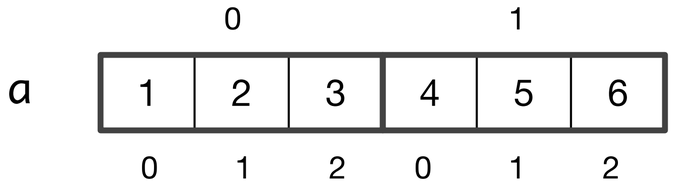


Tableau multi-dimensionnel

- une matrice est une liste de listes

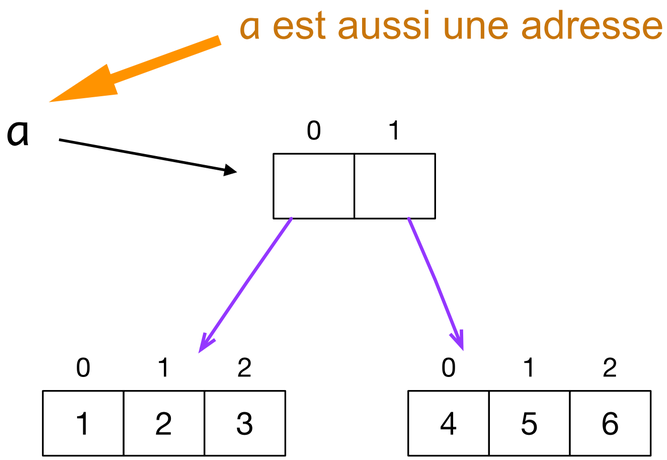
```
>>> a = [[1,2,3], [4,5,6]]

>>> a[0][0]
1
>>> a[0][1]
2
>>> a[0][2]
3
>>> a[1][0]
4
>>> a[1][1]
5
>>> a[1][2]
6
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$



- a est une **référence** vers une liste de 2 éléments

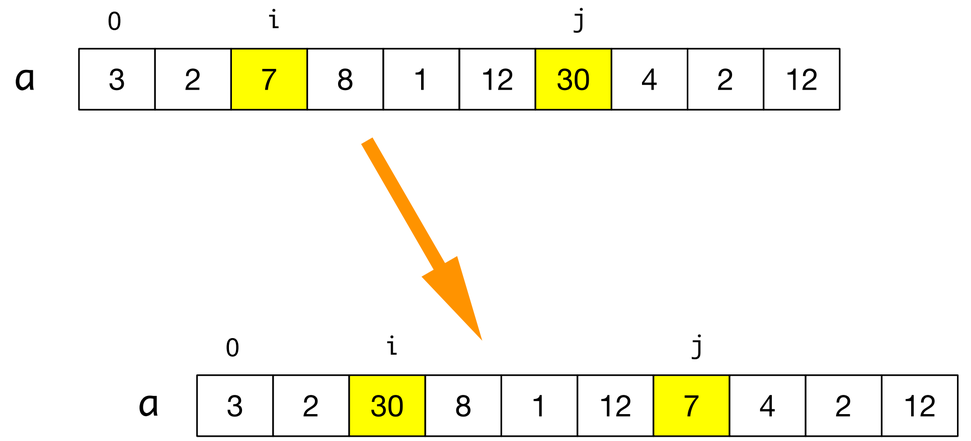
- chaque élément de a est une **référence** vers un tableau de 3 éléments



Tableau

Exercice Écrire la fonction `index_min_of (a)` qui retourne l'indice du minimum dans le tableau *a*

Exercice Écrire la fonction xchange `(a, i, j)` qui échange les éléments du tableau *a* aux indices `i` et `j`



Tableau

Exercice Écrire la fonction `index_min_of (a)` qui retourne l'indice du minimum dans le tableau *a*

Solution

```
def index_min_of (a) :
    n = len (a)
    m = MAX_INT; imin = -1
    for i in range (n):
        if a[i] < m :
            m = a[i]; imin = i
    return imin
```

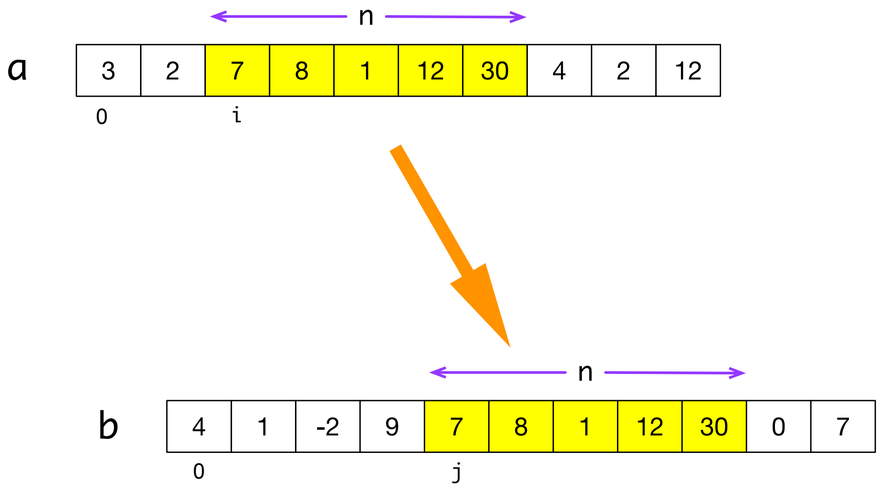
Exercice Écrire la fonction xchange `(a, i, j)` qui échange les éléments du tableau *a* aux indices `i` et `j`

Solution

```
def xchange (a, i, j) :
    t = a[i]
    a[i] = a[j]
    a[j] = t
```

Tableau

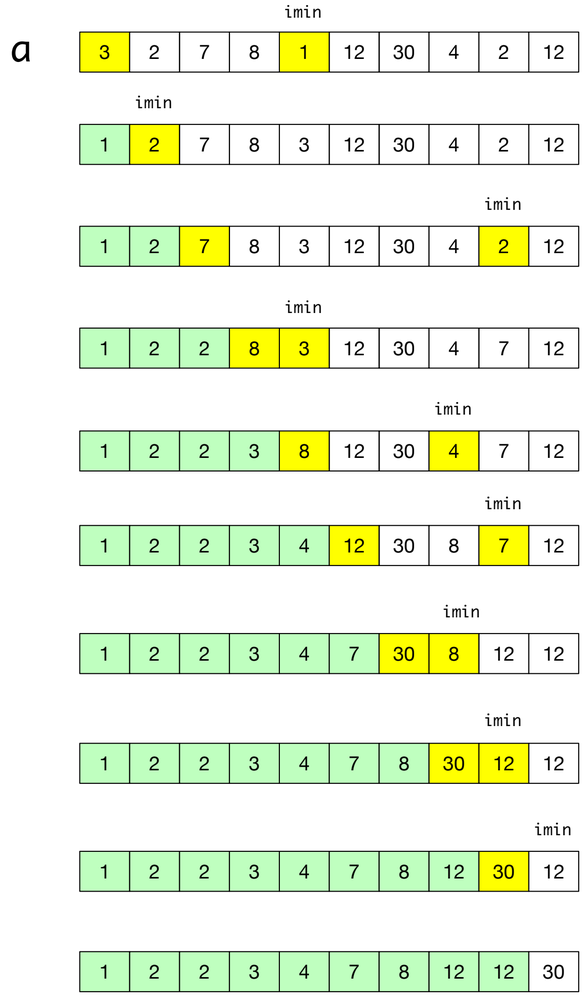
Exercice Écrire la fonction `copy_sublist (a, i, b, j, n)` qui copie les n éléments de *a* à partir de l'indice `i` dans le tableau b à partir de l'indice `j`



Tri sélection

- trier un tableau en ordre croissant
- on cherche l'indice `imin` du minimum dans le tableau
- on échange le premier élément et celui en `imin`
- et on recommence sur le reste du tableau

```
def tri_selection (a) :
    n = len (a)
    for i in range (n-1) :
        imin = i;
        for j in range(i+1, n) :
            if a[j] < a[imin] :
                imin = j
            t = a[i]; a[i] = a[imin]; a[imin] = t
```



Tri par insertion

- trier un tableau en ordre croissant

- comme on trie un jeu de cartes

```
def tri_insertion (a) :
    n = len (a)
    for i in range (1, n) :
        v = a[i]; j = i
        while j > 0 and a[j-1] > v :
            a[j] = a[j-1];
            j = j-1
        a[j] = v
```

Bonus Python

- le module `random` des nombres aléatoires

```
import random

random.randint (x, y)
```
← nombre entre x et y (exclu)

```
random.choice (a)
```
← un élément de a

```
random.sample (range (x, y), n)
```
← tableau de n éléments tous différents entre x et y (exclu)

- list comprehension

```
[i for i in range (x, y)]
```
← tableau des éléments de l'intervalle entre x et y (exclu)

- initialiser une liste avec des nombres aléatoires

```
def new_randlist (x, y, n) :
    return [ random.randint (x, y) for i in range (n)]
```
← tableau de n éléments entre x et y (exclu)

Un peu d'algorithmique

Exercice Le cours de l'action Google en bourse est affichée tous les jours de l'année 2020 dans le tableau g de 366 éléments (entiers positifs).

Ecrire un programme pour trouver les jours où on aurait dû acheter et vendre cette action pour avoir obtenu un gain maximum ?

Exercice Les enfants font une ronde en se tenant par la main. Chacun note sur un bout de papier son nom et celui de son voisin de gauche.

La récréation arrive. Les bouts de papier sont collectés dans une boîte.

Ecrire un programme pour remettre tous les élèves en place.

Le langage C

• vérifier la présence du **compilateur** `gcc` ou `cc` pour C en tapant dans la fenêtre Terminal de VScode:

`g++ --version` et `clang --version` ou `which cc` ou `which gcc`

• sinon installer l'application **XCode** (c'est bien long!) dans le AppStore et ensuite taper dans la fenêtre Terminal

`xcode-select –install`

• installer **l'extension C/C++** de VScode

• **premier** programme en C

```c
#include <stdio.h>

int main() {
    printf ("Hello world!");
}
```

• et on l'exécute avec VScode avec le même interface que déja vu pour Python

Prochain cours

- un bon tutorial Python: `http://www.programiz.com/python-programming`
- les langages procéduraux — suite
- alias
- représentation en mémoire des tableaux
- un peu d’algorithmique
- installer le langage C