Langages de programmation

Cours 3

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-prog

Plan

- installation VScode pour Python
- exécution et mise au point (*debugging*) avec VScode
- révision des programmes du cours 2
- alias et données modifiables
- gestion de la mémoire

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

Visual Studio Code

on suppose avoir téléchargé Python 3 en `http://www.python.org`

- télécharger **VScode** en `http://code.visualstudio.com/download`
  - bien choisir entre Windows, Mac et Linux (avec le bon type de processeur)
- un bon **tutoriel** en `http://youtu.be/VqCgcpAypFQ`
  - d'autres en `basics`, ou en `python-tutorial`, ou en `projet-python`
- on choisit son thème de **couleurs** en `Préférences > Thèmes de couleurs`
- un interface en **français** se trouve dans l'extension `French Language Pack`, en **chinois** dans `Chinese language Pack`
- on charge les extensions: `Python`, `Python for VSCode`, `Python Extension Pack`, `Code Runner`
- et voilà !

Premier programme sous VScode

Méthode 1

- créer une nouvelle fenêtre éditeur et taper

```
print ("Bonjour tout le monde")
```

- on l'exécute en cliquant sur le triangle en haut à droite

Méthode 2

- on change le texte du programme

```
msg = "Bonjour tout le monde"
print (msg)
```

- on met un point d'arrêt (*breakpoint*) avant la ligne 2
- on lance le débogueur avec la touche F5
- et on inspecte la variable `msg`

Méthode 3

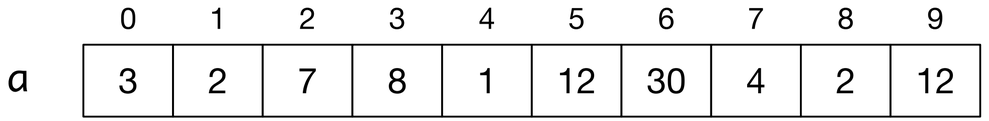
- dans la fenêtre Terminal, on appelle le *toplevel* python
- et on fait couper-coller du texte de la fenêtre éditeur

```
Python 3.8.8 (default, Feb 20 2021, 17:46:49)
[Clang 12.0.0 (clang-1200.0.32.27)] on darwin
Type "help", "copyright", "credits" or "license" ..
>>> print ("Bonjour tout le monde")
Bonjour tout le monde
>>>
```

Tableaux

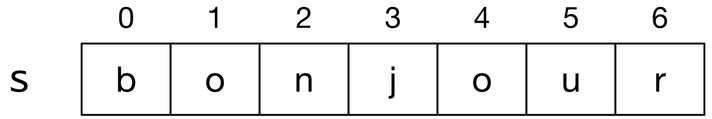
- en Python, les tableaux (modifiables) sont appelés « listes » !
- ils sont dans la classe `list`

```
>>> a = [3, 2, 7, 8, 1, 12, 30, 4, 2, 12]
>>> type (a)
<class 'list'>
```



- les chaînes de caractères sont des tableaux non modifiables de caractères

```
>>> s = "bonjour"
>>> type (s)
<class 'str'>
>>> s[0]
'b'
>>> s[5]
'u'
```



Tableaux

- itération sur une liste (tableau)

```
>>> s = 0
>>> for m in a :
      s = s + m

>>> s
81
```

itération sur tous les éléments de a

- idem avec une fonction

```
>>> def sum_of (x) :
      r = 0
      for m in x :
          r = r + m
      return r

>>> sum_of (a)
81
>>> b = [-3, 42, 23, 11, -30]
>>> sum_of (b)
43
```

x est l’argument de la fonction

Tableaux

- une autre fonction sur les tableaux

```
>>> def max_of (x) :
      r = -1
      for m in x :
         if m > r :
            r = m
      return r

>>> max_of (a)
30

>>> max_of ([ ])
-1
```

x est un tableau de nombres positifs

- impression et longueur d'un tableau

```
>>> print (a)
[3, 2, 7, 8, 1, 12, 30, 4, 2, 12]

>>> len (a)
10
```

- la même fonction avec un tableau de nombres arbitraires

```
>>> import sys
>>> MIN_INT = -sys.maxsize
>>>
>>> def max_of (x) :
      r = MIN_INT
      for m in x :
         if m > r :
            r = m
      return r
```

entier minimum sur 64 bits

Intervalles

- intervalles (*range*)

```
>>> for i in range (0, 10):
      print (i)

0
1
2
3
4
5
6
7
8
9
```

intervalle semi-ouvert

- abréviation et pas (*step*) des intervalles

```
>>> range (10)
range(0, 10)
>>> for i in range (0, 10, 2) :
      print (i)

0
2
4
6
8
```

2 est le pas

Exercice: quel est le sens de `range (10, 0, -1)` ?

- une autre itération sur les indices d'un tableau

```
def index_max_of (x) :
    n = len (x)
    m = MIN_INT; imax = -1
    for i in range (n):
        if x[i] > m :
            m = x[i]; imax = i
    return imax
```

Intervalles et n-uplets

- tranche (*slice*)

```
>>> a
[1, 3, 4, 2, 3, 5, 9]
>>> a[3:6]
[2, 3, 5]
>>> a[3:6:2]
[2, 5]
>>> a[3:]
[2, 3, 5, 9]
>>> a[:6]
[1, 3, 4, 2, 3, 5]
>>> a[::2]
[1, 4, 3, 9]
```

intervalle semi-ouvert

- un n-uplet (*tuple)* est une liste non modifiable

```
>>> b = (9, "novembre", 1989)
>>> b[0]
9
>>> b[1]
'novembre'
>>> b[2]
1989
```

Tableaux

- itération sur les indices d'un tableau

```
>>> def is_palindrome (s) :
        n = len(s)
        for i in range(n) :
            if s[i] != s[n-1-i] :
                return False
        return True

>>> is_palindrome("kayak")
True
>>> is_palindrome("kayok")
False
```

on peut optimiser avec `range(n//2)`

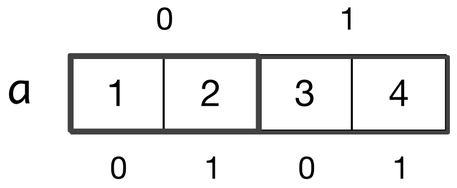
Tableaux multi-dimensionnels

- une matrice est une liste de listes

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

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

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



- addition de matrices

```
def add (a, b) :
    m = len (a); n = len (a[0])
    if m != len(b) or n != len(b[0]) :
        print("Erreur !"); return
    r = new_matrix (m, n)
    for i in range(m) :
        for j in range(n):
            r[i][j] = a[i][j] + b[i][j]
    return r
```

- une itération sur les matrices

```
def print_matrix (a) :
    for line in a :
        for elt in line :
            print (elt, end = ' ')
        print ()
```

impression sur une ligne

Tableaux multi-dimensionnels

- création d'une matrice pleine de zéros

```
def new_matrix (m, n) :
    a = []; z = [0]*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

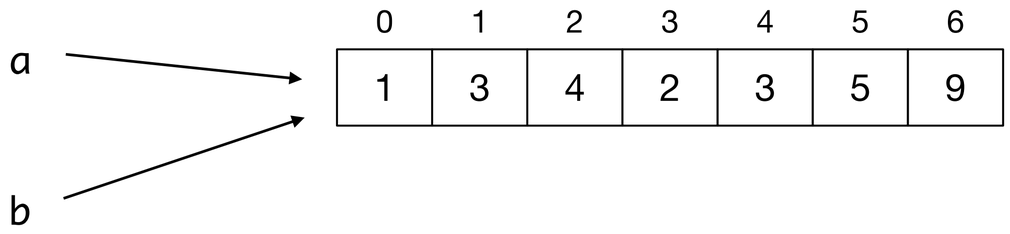
```
>>> a = [1, 3, 4, 2, 3, 5, 9]

>>> 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
```

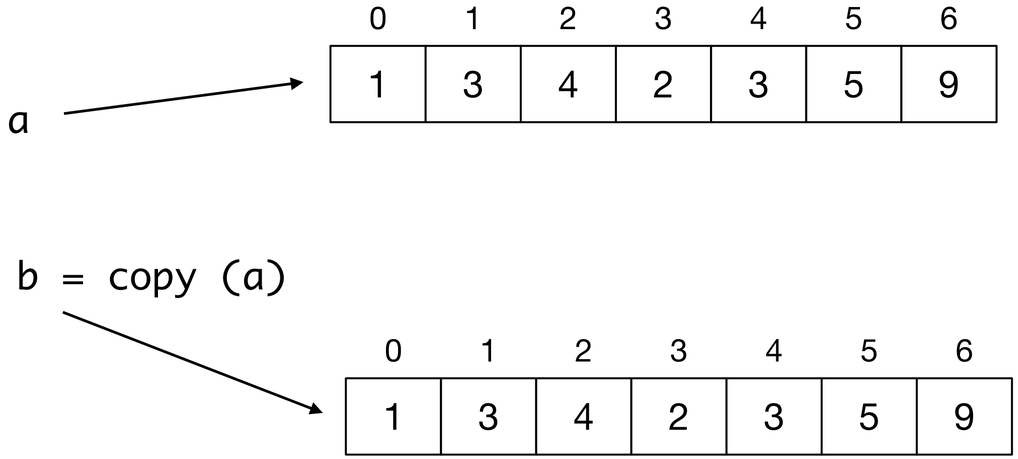
```
>>> c = [1, 3, 4, 2, 3, 5, 9]
>>> id (c)
4342381056

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

Tableau

- copie d'un tableau

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




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

alias et données modifiables
=
DANGER !

Prochain cours

- un bon tutorial Python: `http://www.programiz.com/python-programming`
- les langages procéduraux — suite
- alias
- langage C
- tableaux en C