

Langages de programmation

Cours 2

Jean-Jacques Lévy

`jean-jacques.levy@inria.fr`

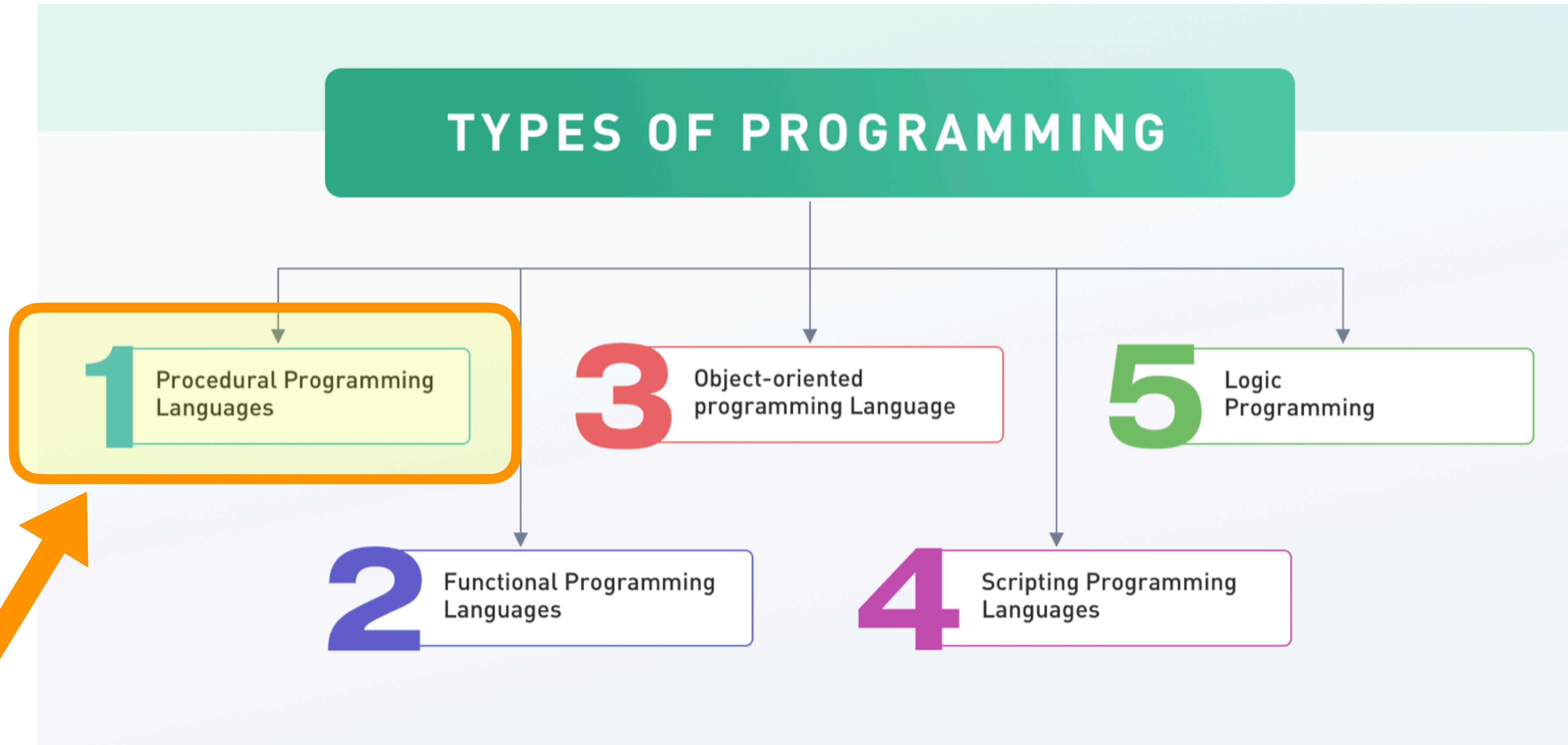
`http://jeanjacqueslevy.net/lp-prog`

Plan

- types de langages de programmation
- langages procéduraux
- tableaux (listes) en Python
- itérations sur les tableaux
- intervalles
- palindrome, carré magique
- alias

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

Types de langages de programmation



Exemples de programmes

- Python (fil conducteur)
 - C
- } **langages impératifs**

- Ocaml
 - Haskell
- } **langages fonctionnels**

- Java
 - C++
- } **langages orienté-objets**

- copie de tableaux
 - tri des éléments d'un tableau
- } **programmation impérative**

- arbre binaire de recherche
 - file d'attente
- } **structures de données dynamiques**

- interface graphique
 - modularité
- } **réutilisabilité**

Ecrire des programmes

- utiliser un système intégré (Visual Studio ou autre)
- ou utiliser une simple fenêtre terminal [le plus rapide pour démarrer], par exemple **Cygwin** avec Windows
- avec un éditeur de texte (**Emacs**, VI, TextEdit, ..)

exemple avec Python

- sur la fenêtre terminal, on tape:

```
mac$ python
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" for more information.
```

- et on peut fonctionner en mode calculette (*toplevel* interactif):

```
>>> 23 + 42
65

>>> 438 * 234
102492
>>> (438 * 234) + 35
102527
>>> ((438 * 234) + 35 / 3)
102503.66666666667
```

Ecrire des programmes

- utiliser un système intégré (Visual Studio ou autre)
- ou utiliser une simple fenêtre terminal [le plus rapide pour démarrer], par exemple **Cygwin** avec Windows
- avec un éditeur de texte (**Emacs**, VI, TextEdit, ..)

exemple avec C

- dans un éditeur de texte, on crée le fichier **bonjour.c**

```
#include <stdio.h>

char s[100] = "bonjour les amis!" ;

int main()
{
    printf ("%s\n", s);
}
```

- sur la fenêtre terminal, on tape:

```
mac$ cc bonjour.c
mac$ a.out
bonjour les amis!
```

Tableaux

- données scalaires: entiers, flottants, booléens, caractères
- données structurées: tableaux

```
>>> a = [3, 2, 7, 8, 1, 12, 30, 4, 2, 12]
>>> a[0]
3
>>> a[4]
1
>>> a[8]
2
>>> a[-1]
12
```

- les tableaux sont modifiables

```
>>> a[7] = 28

>>> a
[3, 2, 7, 8, 1, 12, 30, 28, 2, 12]
```

	0	1	2	3	4	5	6	7	8	9
a	3	2	7	8	1	12	30	4	2	12

- les tableaux de Python peuvent ne pas être homogènes

```
>>> a[9] = 'aaa'

>>> a
[3, 2, 7, 8, 1, 12, 30, 28, 2, 'aaa']
```

Tableaux

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

```
>>> type (a)
<class 'list'>
```

	0	1	2	3	4	5	6	7	8	9
a	3	2	7	8	1	12	30	4	2	12

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

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

	0	1	2	3	4	5	6
s	b	o	n	j	o	u	r

Tableaux

- itération sur une liste (tableau)

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

← itération sur tous les éléments de a

```
>>> s
81
```

- idem avec une fonction

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

← x est l'argument de la fonction

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

Tableaux

- une autre fonction sur les tableaux

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

← x est un tableau de nombres positifs

```
>>> max_of (a)  
30
```

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

- 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

- impression et longueur d'un tableau

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

```
>>> len (a)  
10
```

Intervalles

- intervalles (*range*)

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

← **intervalle semi-ouvert**

0
1
2
3
4
5
6
7
8
9

- abréviation et pas (*step*) des intervalles

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

← **2 est le pas**

0
2
4
6
8

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
```

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

	0	1		
a	1	2	3	4
	0	1	0	1

- 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
```

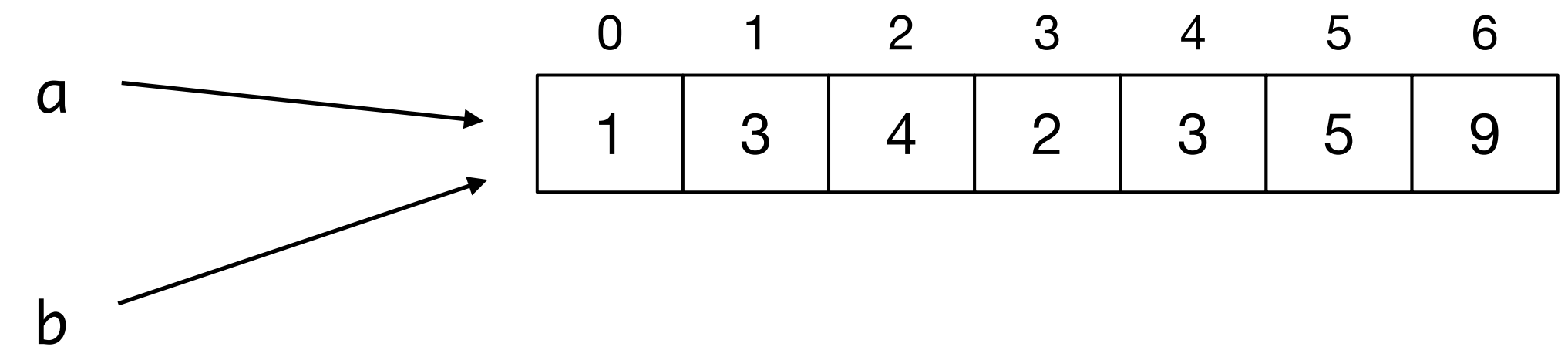
```
>>> b
```

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

```
>>> a
```

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

← a et b sont des alias



- 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
```

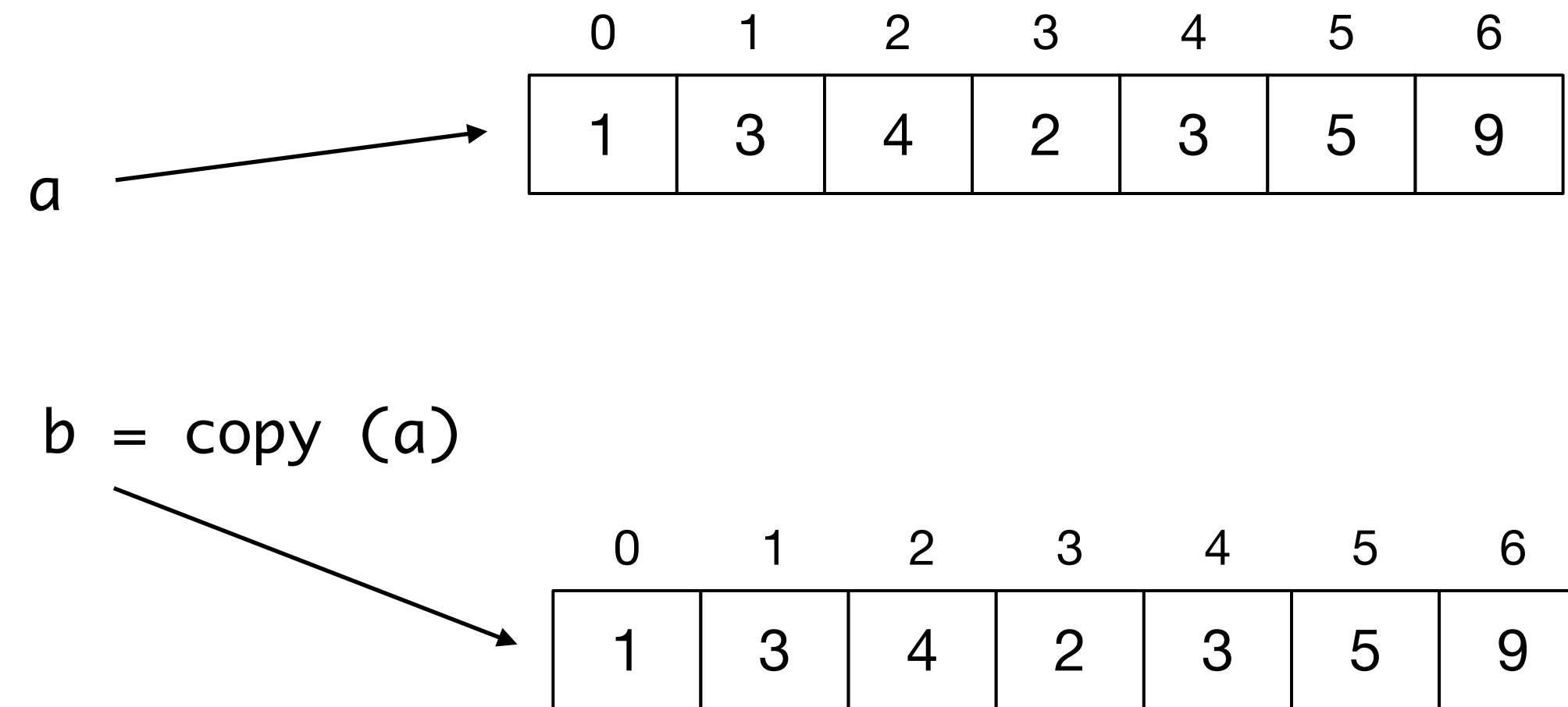
```
False
```

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
```



Prochain cours

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