Informatique et Programmation

Appendice 1

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/prog-py-22

Plan

- fibonnacci itératif
- listes (tableaux), ensembles, dictionnaires
- notation par compréhension
- tris récursifs
- graphique
- courbes fractales

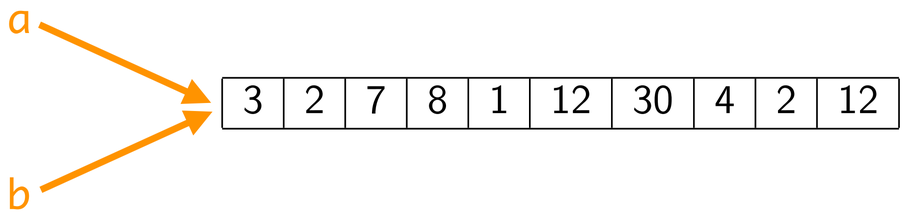
dès maintenant: télécharger Python 3 en `http://www.python.org`

un cours Python en `http://www.w3schools.com/python/default.asp`

Valeur d'un tableau — Alias

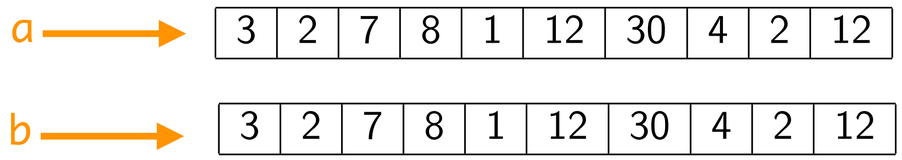
- soient 2 tableaux a et b

```
a = [3, 2, 7, 8, 1, 12, 30, 4. 2. 12]
b = a

a[2] = 888

print (a)
[3, 2, 888, 8, 1, 12, 30, 4. 2. 12]

print (b)
[3, 2, 888, 8, 1, 12, 30, 4. 2. 12]
```



- les variables a et b sont 2 alias d'un même tableau

```
a = [3, 2, 7, 8, 1, 12, 30, 4. 2. 12]
b = [3, 2, 7, 8, 1, 12, 30, 4. 2. 12]

a[2] = 888

print (a)
[3, 2, 888, 8, 1, 12, 30, 4. 2. 12]

print (b)
[3, 2, 7, 8, 1, 12, 30, 4. 2. 12]
```



- les variables a et b sont 2 tableaux distincts

- la valeur de a ou de b est l'adresse mémoire de son premier élément

données modifiables et alias sont sources de bugs

Dictionnaires

- les dictionnaires sont des listes d'associations : clé — valeur

```
adresse = {'JJ': (32, 'boulevard St Michel', Paris, 'F', 75005),
           'Robert': (3, 'rue Chaban-Delmas', Bordeaux, 'F', 33000),
           'maman': (4, 'corniche André de Joly', Nice, 'F', 06300),
           'Santa': (6, 'route du Ciel', Marmande, 'F', 31330),
           'Xi': (2, 'pte de la Paix céleste', Beijing, 'PRC', 0001),
           'Manuel': (3, 'faubourg St Honoré', Paris, 'F', 75001)}
```

- les dictionnaires sont des listes d'associations : clé — valeur

[random.randint (0, 3) for i in range(20)]

```
age = {'JJ':35, "marie": 25, "alice": 38, "bob": 29}
print (age)
→ {'bob': 29, 'marie': 25, 'alice': 38, 'JJ': 35}
print (age['marie'])
→ 25
print (age['JJ'])
→ 35
age['raymond'] = 29
print (age)
→ {'bob': 29, 'marie': 25, 'alice': 38, 'JJ': 35, 'raymond': 29}
age ['JJ'] = 49
print (age)
→ {'bob': 29, 'marie': 25, 'alice': 38, 'JJ': 49, 'raymond': 29}
```

```
print (age['henry'])
→ Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'henry'

'henry' in age
→ False
'alice' in age
→ True
```

Ensembles — Compréhension

- les ensembles sont des listes non ordonnées d'éléments tous distincts

```
print ({10, 2, 3} == {3, 2, 10})
True

print ({10, 2, 3} == {5, 2, 10})
False
```

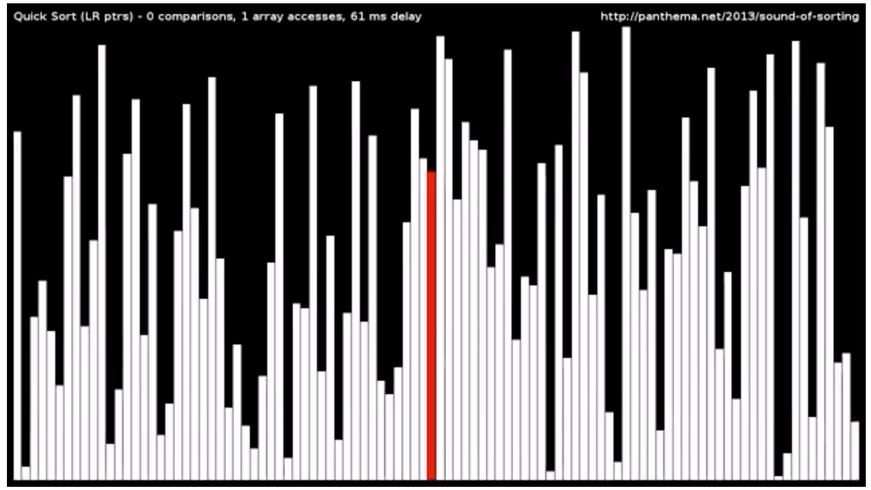
- on peut générer des tableaux, listes, ensembles, dictionnaires avec la notation compréhensive

```
a = [x**2 for x in range(21) if x*2 < 21]
print (a)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

b = {2 * x for x in range (20)}
print (b)
{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38}

c = {x : x**2 for x in range (10)}
print (c)
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

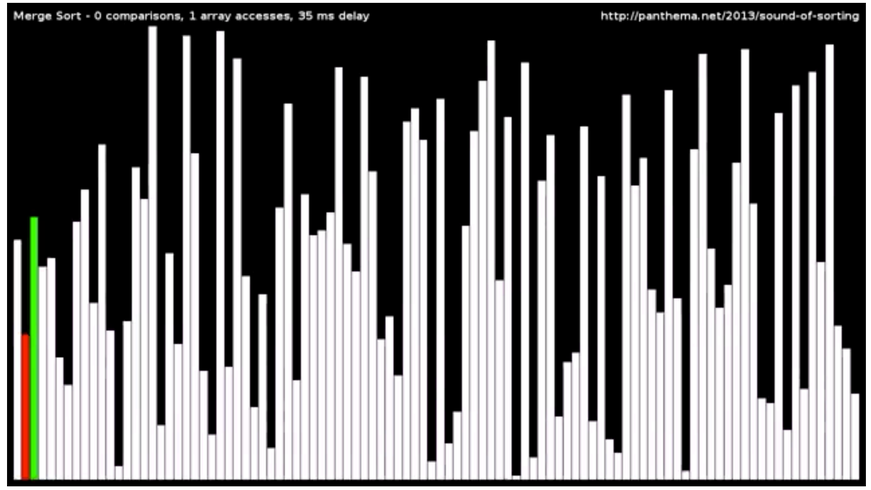
Exercice Que fait l'expression `[random.randint (0, 3) for i in range(20)]` ??

Tri rapide — Tri fusion

- tri et récursivité: quicksort



- tri et récursivité: mergesort

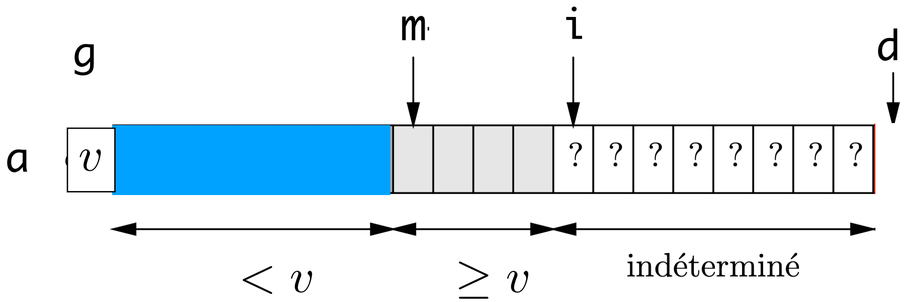


Tri récursif

- tri rapide (*Quicksort*)

```
def tri_rapide (a) :
    tri_rapide1 (a, 0, len(a))


def tri_rapide1 (a, g, d) :
    if g < d - 1 :
        v = a[g]
        m = g + 1
        for i in range (g+1, d):
            if a[i] < v :
                t = a[m]; a[m] = a[i]; a[i] = t
                m = m + 1
        a[g] = a[m-1]; a[m-1] = v
        tri_rapide1 (a, g, m-1)
        tri_rapide1 (a, m, d)
```

- bonne méthode de tri en moyenne



a) on met `a[g]` à sa place dans `a` trié

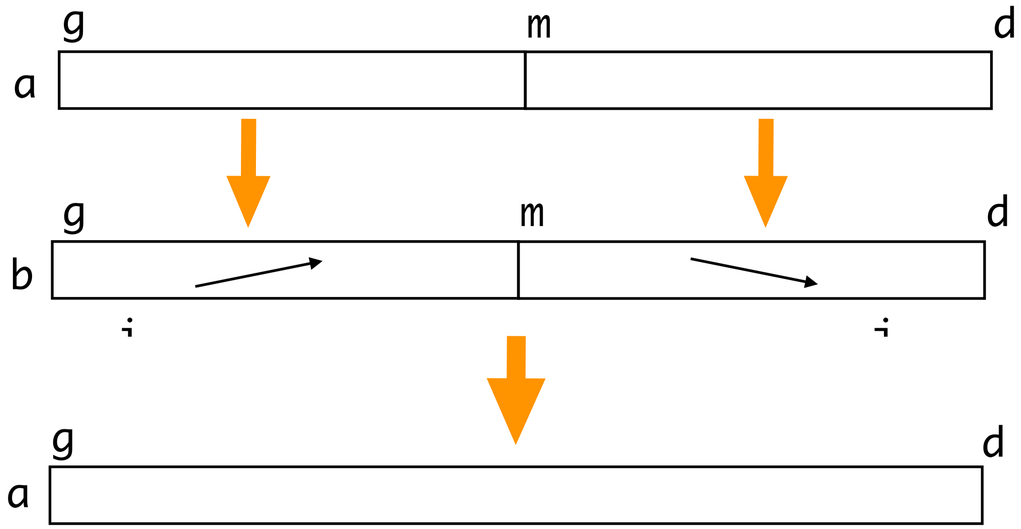
b) et on recommence sur les parties gauche et droite

Tri récursif

- tri fusion (*merge sort*)

```
def tri_fusion (a) :
    n = len (a)
    b = n*[0]  # tableau intermédiaire
    tri_fusion1 (a, 0, n, b)
```

```
def tri_fusion1 (a, g, d, b) :
    if g < d - 1 :
        m = (g + d) // 2
        tri_fusion1 (a, g, m, b)
        tri_fusion1 (a, m, d, b)
        for i in range (g, m) :
            b[i] = a[i]
        for j in range (m, d) :
            b[m + d - 1 - j] = a[j]
        i = g; j = d - 1;
        for k in range (g, d) :
            if b[i] < b[j] :
                a[k] = b[i]; i = i + 1
            else :
                a[k] = b[j]; j = j - 1
```



a) on coupe a en 2

b) on trie les moitiés gauche et droite

c) on copie les résultats dans un tableau annexe b

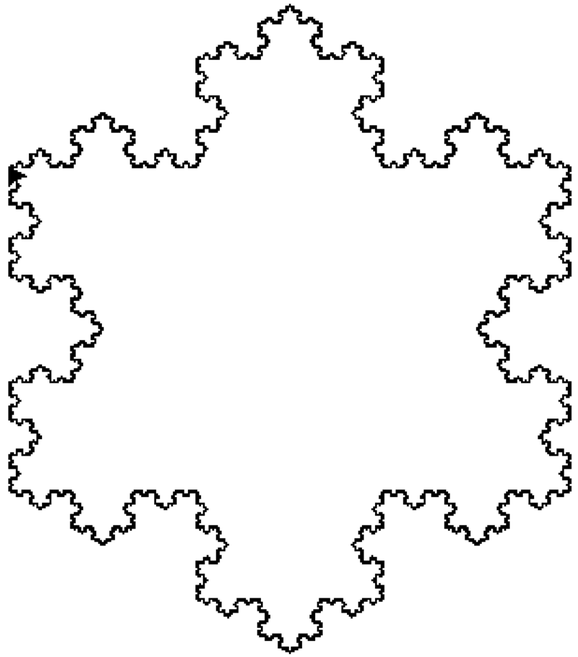
d) on fusionne les 2 moitiés dans le tableau a

- très bonne méthode de tri

Graphique avec la tortue

- un paquetage `turtle.py` simple pour apprendre l'informatique dans les écoles
  (cf. http://fr.wikipedia.org/wiki/Seymour_Papert)

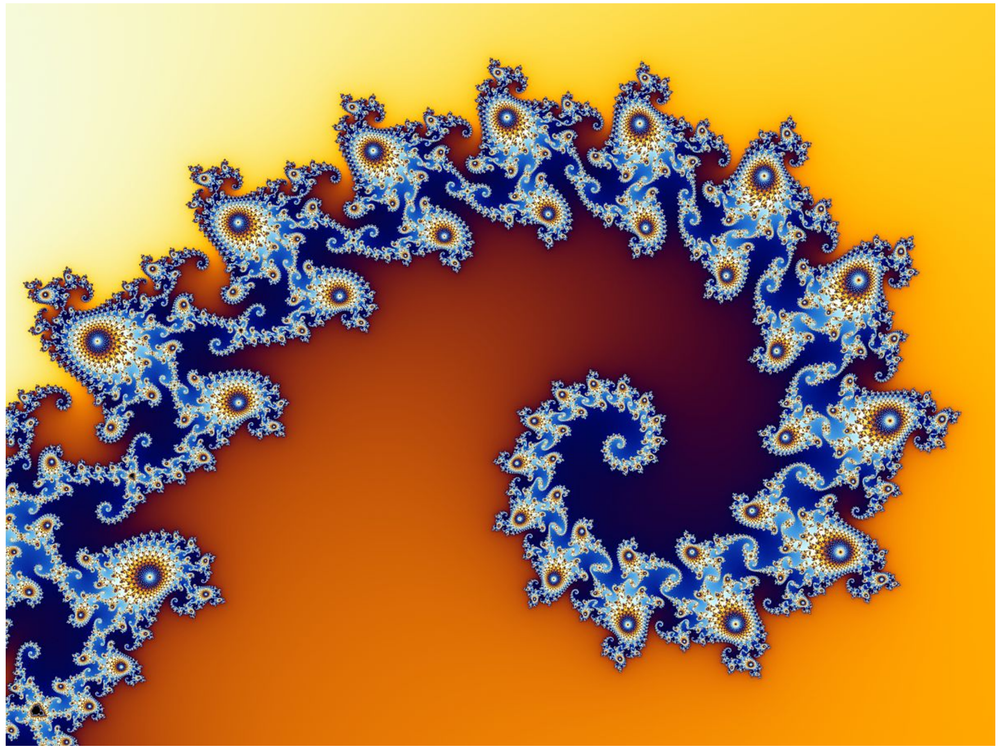
```
from turtle import *

def koch (l, n) :
    if n <= 0 :
        forward (l)
    else:
        koch (l/3, n-1); left(60)
        koch (l/3, n-1); right (120)
        koch (l/3, n-1); left (60)
        koch (l/3, n-1)

def flocon (l, n) :
    koch (l, n); right (120)
    koch (l, n); right (120)
    koch (l, n)
```

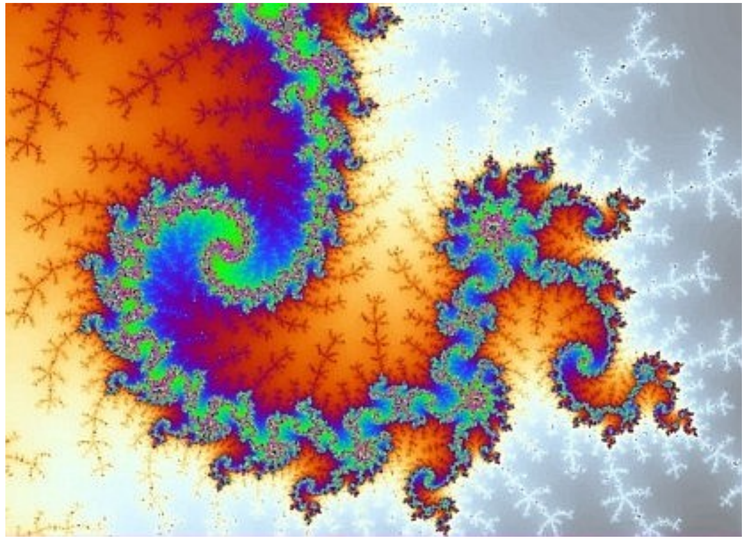


les sources sont en http://jeanjacqueslevy.net/prog-py-22/progs/c5a.py

Fractales

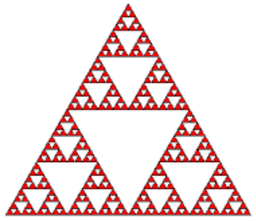
- les fractales de Benoît Mandelbrot

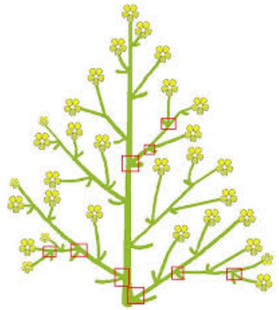


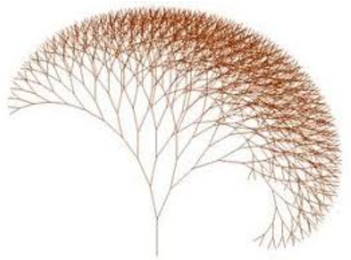


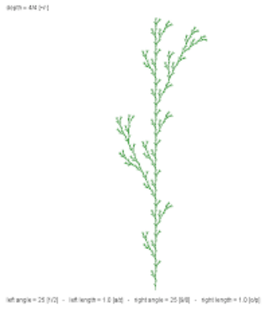
Fractales

- les fractales de Benoît Mandelbrot









Fractales

- la courbe de Hilbert

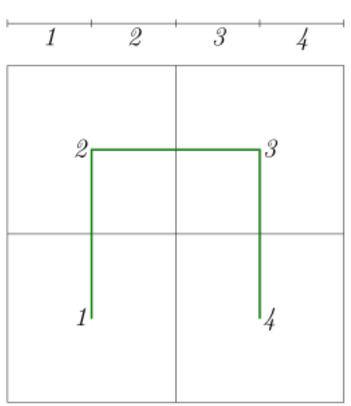


Fig. 1.

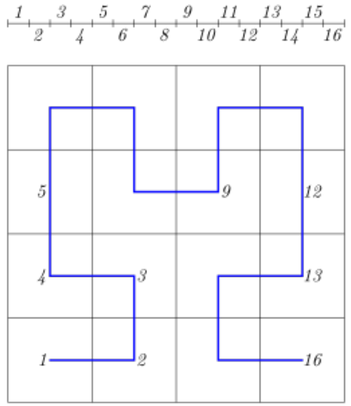


Fig. 2.

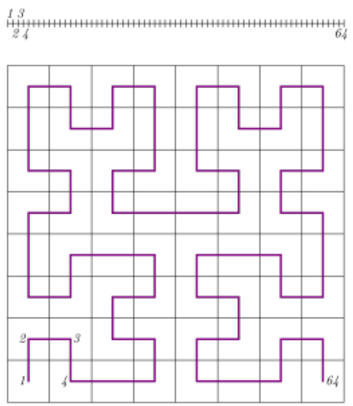
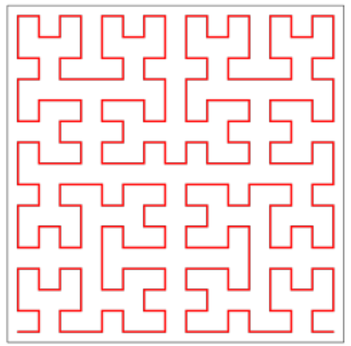
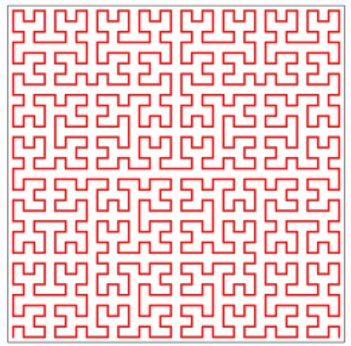
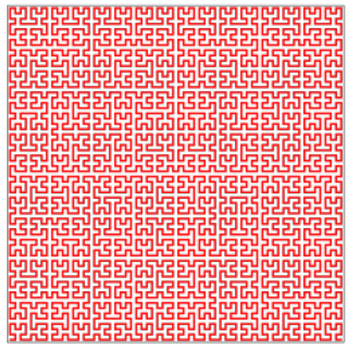


Fig. 3.







Exercice : écrire le programme qui la dessine

Graphique - la tortue de Papert

- classes et objets