Suite de Fibonacci : quel est le meilleur algorithme ?

Comparer les temps avec timeit

La librairie timeit mesure les temps d'exécution en évitant des biais tels que l'usage concomitant d'autres ressources.

fibonacci09_fonctions_comparaison.py3

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
"""
Calculs des premiers éléments de la suite de Fibonacci.
Référence : http://fr.wikipedia.org/wiki/Suite_de_Fibonacci
Comparaison de différentes fonctions avec Timeit
http://docs.python.org/2/library/timeit.html
http://www.diveintopython.net/performance_tuning/timeit.html ; traduit
en français ici :
http://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/
performance_tuning/timeit.php
"""
import timeit
from fibonacci05_fonction import fibonacci_item
from fibonacci06_fonctions import fibonacci_item_from_list
from fibonacci07_fonction_recursive import fibonacci_item_recursive
from fibonacci08_fonction_algo_log import fibonacci_item_logarithmic

if __name__ == '__main__':
    print("Calculs exemples préliminaires...")
    i=int(input("Suite de Fibonacci. Donnez l'indice de l'élément
souhaité ? "))
    print ("Élément de la suite : "),
    if i <= 10: print(fibonacci_item_recursive(i))
    print(fibonacci_item_from_list(i))
    print(fibonacci_item(i))
    print(fibonacci_item_logarithmic(i))
    print("Temps d'exécution")
    #using timeit :
    t1 = timeit.Timer("fibonacci_item_recursive(" + str(i) + ")","from
fibonacci07_fonction_recursive import fibonacci_item_recursive")
    t2 = timeit.Timer("fibonacci_item_from_list(" + str(i) + ")","from
fibonacci06_fonctions import fibonacci_item_from_list")
    t3 = timeit.Timer("fibonacci_item(" + str(i) + ")","from
fibonacci05_fonction import fibonacci_item")
    t4 = timeit.Timer("fibonacci_item_logarithmic(" + str(i) +
")","from fibonacci08_fonction_algo_log import
```

```
fibonacci_item_logarithmic")
    if i <= 10:
        print("fibonacci_item_recursive")
        print("1000 exécutions : ",t1.timeit(1000))
        print("Cinq ensembles de 10000 exécutions : ",t1.repeat(5,
1000))
    print("fibonacci_item_from_list")
    print("1000 exécutions : ",t2.timeit(1000))
    print("Cinq ensembles de 10000 exécutions : ",t2.repeat(5, 1000))
    print("fibonacci_item_")
    print("1000 exécutions : ",t3.timeit(1000))
    print("Cinq ensembles de 10000 exécutions : ",t3.repeat(5, 1000))
    print("fibonacci_item_logarithmic")
    print("1000 exécutions : ",t4.timeit(1000))
    print("Cinq ensembles de 10000 exécutions : ",t4.repeat(5, 1000))
```

Traitements statistiques

Les listes de mesures

- La statistique descriptive permet de décrire un ensemble de mesure par quelques valeurs caractéristiques :
  - la moyenne
  - la médiane
  - le maximum
  - le minimum
  - la variance et sa racine carrée, l'écart type

Exercices :

- créer en Python des fonctions qui extraient ces valeurs pour une liste de nombres.
- rechercher des librairies spécialisées et leurs fonctionnalités fournissant facilement ces paramètres (numpy, scipy, statsmodels, pandas,...)

L'influence d'une variable

Caractériser plusieurs ensembles de mesures pour lesquels on fait varier un paramètre (le numéro de l'élément dans la suite par exemple), et étudier la dépendance des variables mesurées en fonction de ce paramètre.