

Suite de Fibonacci : encore un algorithme

Voici le programme complété pour la technique récursive :

[fibonacci07_fonction_recursive.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
Application de la définition par récursivité.
"""
def fibonacci_item_recursive(n):
    """
    Renvoie l'élément d'indice n de la suite de Fibonacci
    """
    if n == 0:
        return 0
    elif n == 1:
        return 1
    return fibonacci_item_recursive(n-1)+fibonacci_item_recursive(n-2)

if __name__ == '__main__':
    i = input("Suite de Fibonacci. Donnez l'indice de l'élément
souhaité ? ")
    print("Élément de la suite : "),
    print(fibonacci_item_recursive(i))
    print('Premiers éléments de la suite : ')
    for j in range(10):
        print(j, fibonacci_item_recursive(j))
```

La page Wikipedia sur la suite de Fibonacci introduit aussi un [W algorithme logarithmique](#). Même s'il est très intéressant à décortiquer, on peut se contenter de simplement l'appliquer :

[fibonacci08_fonction_algo_log.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
Application de l'algorithme logarithmique
http://fr.wikipedia.org/wiki/Suite_de_Fibonacci#Algorithme_logarithmiqu
e
"""
def fibo2(n):
    """Renvoie F_{n-1}, F_n"""
    if n == 0: # cas de base
```

```
    return 1, 0 # F_{-1}, F_0
else: # récurrence
    f_k_1, f_k = fibo2(n//2) # F_{k-1}, F_k avec k = n/2
    f2_k = f_k**2 # F_k^2
    if n%2 == 0: # n pair
        return f2_k + f_k_1**2, f_k*f_k_1*2 + f2_k #
F_{2k-1}, F_{2k}
    else: # n impair
        return f_k*f_k_1*2 + f2_k, (f_k + f_k_1)**2 + f2_k #
F_{2k}, F_{2k+1}

def fibonacci_item_logarithmic(n):
    """Renvoie F_n"""
    return fibo2(n)[1]

if __name__ == '__main__':
    i = input("Suite de Fibonacci. Donnez l'indice de l'élément
souhaité ? ")
    print("Élément de la suite : "),
    print(fibonacci_item_logarithmic(i))
    print('Premiers éléments de la suite : ')
    for j in range(10):
        print(j, fibonacci_item_logarithmic(j))
```

Sur la même page wikipedia, on trouve une [W expression fonctionnelle](#), de complexité apparente en temps constant, aussi connue sous le nom de [W formule de Binet](#), mais qui passe par le calcul du nombre irrationnel $\sqrt{5}$, ce qui pose un problème pour conserver une précision des chiffres significatifs par rapport à l'arithmétique entière.

Le langage Python permet également de mettre en œuvre des générateurs (generator) : cf. <http://www.koderdojo.com/blog/python-fibonacci-number-generator> et http://www.bogotobogo.com/python/python_generators.php

[fibonacci09_generator.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
"""
def fibonacci_gen(n):
    """
    Renvoie l'élément d'indice n de la suite de Fibonacci
    """
    a, b = 0, 1
    for _ in range(n):
        yield a
        a, b = b, a + b
```

```
if __name__ == '__main__':  
    i = int(input("Suite de Fibonacci. Donnez l'indice max des éléments  
souhaités ? "))  
    print("Le générateur : "),  
    print(fibonacci_gen(i))  
    print('Éléments de la suite : ')  
    for j in fibonacci_gen(i):  
        print(j)
```

Nous disposons à présent de plusieurs méthodes/fonctions pour calculer les éléments de la suite de Fibonacci.

Pour rechercher quel est le meilleur algorithme, [cliquez ici](#) !

From:

<https://dvillers.umons.ac.be/wiki/> - **Didier Villers, UMONS - wiki**

Permanent link:

https://dvillers.umons.ac.be/wiki/teaching:progappchim:suite_de_fibonacci-4

Last update: **2017/02/24 09:09**

