

# Suite de Fibonacci : encore un algorithme


Voici le programme complété pour la technique récursive : <sxh python; title : fibonacci07\_fonction\_recursive.py> `#!/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)
```

</sxh>

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

<sxh python; title : fibonacci08\_fonction\_algo\_log.py> `#!/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\_logarithmique """` 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)
```

</sxh>

Sur la même page wikipedia, on trouve une [expression fonctionnelle](#), de complexité apparente en temps constant, aussi connue sous le nom de [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.

Nous disposons à présent de 5 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?rev=1476794858](https://dvillers.umons.ac.be/wiki/teaching:progappchim:suite_de_fibonacci-4?rev=1476794858)

Last update: **2016/10/18 14:47**

