Generate Fibonacci numbers in Python

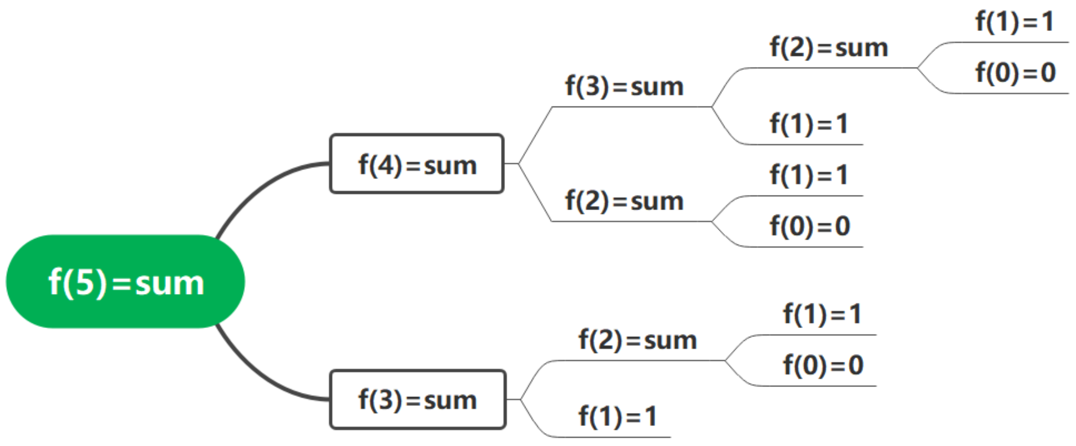
Fibonacci sequence is the series of numbers where each number is the sum of the two preceding numbers. For example, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55… 55 = 34+21; 34 = 21+13; 21 = 13+8 …

To first two elements are 0 and 1, the rest will follow: n = (n-1) + (n-2)

The code to get the nth element of Fibonacci number function f(n):

```
def f(n):
   if n == 0:
      return 0
   elif n == 1:
      return 1
   else:
      return f(n-1) + f(n-2)
```

The basic, straightforward, easy to read, minimal algorithm to generate the Fib sequence is above: a recursive function that calls itself.



Above is the 5th Fibonacci number. As we see, to get f(5), predecessors f(4) and f(3) need to be calculated first. Separately, f(3) & f(2) and f(2)&f(1) need to be calculated, on and on till f(1) and f(0). If we want the 100th, or 300th of the Fibonacci number, we can predict that lots of repeated computation will be done like the f(5) shows, it will take a long time for the computation. Is there a better way to improve the efficiency?

Memorizing the recursive algorithm and avoiding repeated computation

```
d = {0:0,1:1}           #set an initialized dictionary
def fib(n):
   if n in d:
      return d[n]
   d[n] = fib(n-1) + fib(n-2)    # the nth key calculated and assigned the value of f(n-1) + f(n-2)
   return d[n]                   # next when n in the dictionary, no repeating computation needed
```