Correction : La récursivité

■ **Exercice .1.**
Calculer de façon itérative et récursive la fonction de Fibonacci définie par :

— $F(0) = 1$
— $F(1) = 1$
— $\forall n \geq 2\ F(n) = F(n-1) + F(n-2)$

1. Proposer un pseudo-code (ou en python) de l'implémentation itérative et récursive de cette fonction.Dans le cas récursif, souligner la condition d'arrêt, le cas trivial, et l'appel récursif de cette fonction.

```
def Fibo(n):
        if n <= 1:
                return 1
        else:
                return Fibo(n-1) + Fibo(n-2)

```

2. Quel est l'inconvénient de ce calcul récursif? On le démontrera par une visualisation des appels effectués par la fonction récursive et une visualisation du calcul itératif.
   L'inconvénient du calcul récursif est que certains termes seront calculés plusieurs fois. Dans l'exemple suivant l'ordinateur calculera plusieurs fois Fibo(3).

$$Fibo(5) = Fibo(4) + Fibo(3) = F(3) + Fibo(2) + Fibo(3)$$

■

■ **Exercice .2.**
Exercice type Bac :
On souhaite créer une fonction qui donne l'écriture binaire d'un nombre décimal. Écrivez une telle fonction de façon itérative et récursive. Puis dans le cas récursif, souligner la condition d'arrêt, le cas trivial, et l'appel récursif de cette fonction.

```python
def binaire(n):
        if n>0:
                reste = str(n%2)
                return binaire(n//2) + reste
        else:
                return ""

```

■

■ **Exercice .3.**
Exercice classique : Les tours de Hanoï

1. Regarder sur internet, les règles des tours de Hanoï
2. Écrire le pseudo-code de la fonction récursive calculant la solution des déplacements de n anneaux dans les tours de Hanoï, sachant qu'on ne peut pas poser un anneau plus grand sur un anneau plus petit.

```python
def hanoi(n,a=1,b=2,c=3):
        if (n > 0):
                hanoi(n-1,a,c,b)
                print("D\'eplace {} sur {}".format(a
   ,c))
                hanoi(n-1,b,a,c)

```

■

■ **Exercice .4.**
Écrire une fonction récursive qui calcule le quotient de la division entière de deux nombres strictement positifs.

```python
def quotien(a,b):
        if b>a:
                res = 0
        else:
                res = 1 + quotient(a-b,b)
        return res

```

■

■ **Exercice .5.**
Écrire une fonction récursive qui calcule le reste de la division entière de deux nombres strictement positifs.

```
def reste(a,b):
        if b>a:
                res = a
        else:
                res = reste(a-b,b)
        return res
```

■

■ **Exercice .6.**
Un palindrome est une chaîne de caractères qui se lit de gauche à droite ou de droite à gauche, comme « radar », « elle », « été », « esope reste ici et se repose », si l'on fait abstraction des blancs. Écrire un algorithme récursif qui permet de vérifier si une chaîne de caractères est un palindrome ou non.

```
def palindrome(s):
        l = len(s)
        if l <= 1:
                return True
        else:
                return (s[0] == s[-1]) and
   palindrome(s[2:-2])

```

■

■ **Exercice .7.**
La fonction miroir donne l'inverse d'une chaîne de caractères, c'est-à-dire la chaîne formée des mêmes caractères mais dans l'ordre inverse, par exemple miroir("séminaire") = "erianimés". Écrire un algorithme récursif qui permet de réaliser cette fonction, et puis proposer une version itérative équivalente.

```python
def miroir(s):
        l = len(s)
        if l <= 1:
                return s
        else:
                return s[-1] + miroir(s[:-1])

```

■ Exercice .8.

Soit une fonction récursive puissance qui prend en paramètre un réel n et un entier k et renvoie le calcul de n^k

1. Ecrire la fonction récursive puissance qui calcule n^k de façon naïve en considérant $n^k = n \times n^{k-1}$ et $n^0 = 1$.

```python
def puissance(n,k):
        if n==0:
                return 1
        else:
                return n*puissance(n,k-1)

```

2. Il existe une autre méthode de calcul de n^k plus efficace : la méthode par dichotomie. Elle repose sur le fait que si k est pair alors $n^k = (n \times n)^{k/2}$ et $n^k = n \times n^{k-1}$ sinon. Ecrire la fonction récursive puissance-dic qui calcule n^k par dichotomie.

```python
def puissance_dic(n,k):
        if k == 0:
                return 1
        elif k %2 == 0:
                return puissance_dic(n*n,k/2)
        else:
                return n*puissance_dic(n,k-1)
```

3. Transformer la version précédente en récursivité terminale.