# Informatique en CPGE (2018-2019)
# TD 3 : récursivité

**Exercice 1 : suite de Fibonacci**

La suite de Fibonacci est définie par $u_0 = u_1 = 1$ et $u_n = u_{n-1} + u_{n-2}$ pour $n \geq 2$.

1. Ecrire une fonction itérative **fibonacci1(n)** qui à l'aide d'une boucle "while" calcule et renvoie le terme $u_n$ de la suite de Fibonacci.
   Utiliser l'échange de variables (qui permet de passer d'un couple $(u_n, u_{n+1})$ au couple $(u_{n+1}, u_{n+2})$, sans oublier un compteur).
2. Ecrire une fonction itérative **fibonacci2(n)** qui à l'aide d'une boucle "for" calcule et renvoie le terme $u_n$ de la suite de Fibonacci.
3. Ecrire une fonction récursive non terminale **fibonacci_rec(n)** qui calcule et renvoie le terme $u_n$ de la suite de Fibonacci.
   Evaluer le nombre d'opérations nécessaires au calcul de fibo(36) et le temps de calcul sur votre ordinateur avec ce programme. En déduire le temps approximatif qu'il faudrait pour calculer fibo(50).
   Pour mesurer le temps de fonctionnement d'un programme, on peut utiliser les instructions suivantes :

```
from time import time
st=time()
programme ...
print('temps = ',time()-st)
```

4. Ecrire une fonction récursive terminale **fibonacci_recT(n,i,a,b)** qui calcule et renvoie le terme $u_n$ de la suite de Fibonacci. Les paramètres a et b sont deux termes consécutifs de la suite et i est un compteur.
   Pour calculer $u_{350}$, l'appel se fera par : fibonacci_recT(350,0,1,1).

**Exercice 2 : algorithme de Hörner**

On compare deux algorithmes permettant d'évaluer la valeur de $P(x)$ pour une valeur de $x$ donnée avec $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$.
Le polynôme $P$ est défini par la liste des coefficients $[a_0, a_1, \ldots, a_n]$.

1. Le programme correspondant au premier algorithme est le suivant ; déterminer la complexité de cet algorithme.

```
def polynome(coef,x):
    p=0
    for i in range(len(coef)):
        f=1
        for j in range (1,i+1):
            f*=x
        p+=coef[i]*f
    return p
```

   Tester ce programme avec le polynôme défini par $[1.3, -3, 1, 4, 8, -4, 2, 1, 5, -6, 3]$.

2. Le programme correspondant au deuxième algorithme, appelé *algorithme de Hörner*, est basé sur une écriture différente de $P(x)$ : $P(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n)\ldots))$.
   (a) Ecrire une fonction `horner` itérative (avec une boucle "for") et déterminer la complexité de l'algorithme.
   (b) Ecrire une fonction récursive `horner_rec` et déterminer la complexité de l'algorithme.