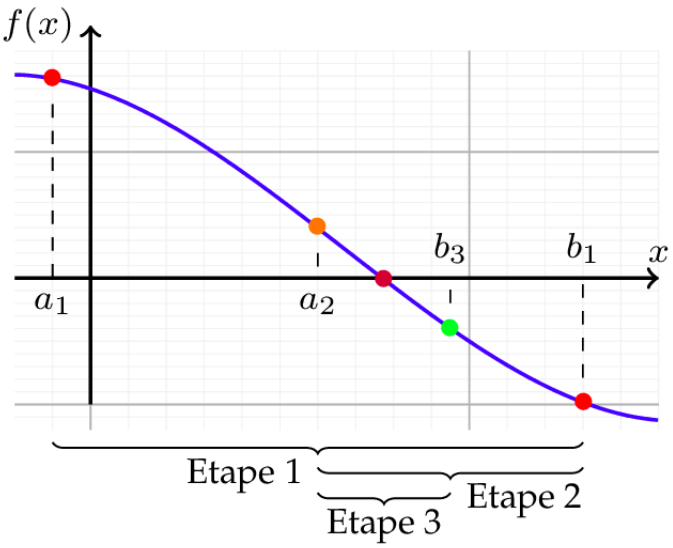
TP3

Rappels de 1ère année : algorithme de dichotomie

Principe de l'algorithme

On cherche une approximation de l'unique solution $\alpha \in [a;b]$ d'une équation du type $f(x) = 0$ où f est une fonction continue réalisant une bijection de $[a;b]$ sur un intervalle.
On rappelle pour cela le principe de l'algorithme de dichotomie.

- On initialise deux variables a et b en leur affectant respectivement les valeurs 0 et 1.
- Tant que $b - a > \varepsilon$, on répète les opérations suivantes.
  On considère le milieu c du segment $[a,b]$. Par monotonie de f sur $]0,1]$, en distinguant les cas $f(c) < 0$ et $f(c) > 0$, on peut déterminer si u_n, appartient à l'intervalle $[a,c]$ ou à l'intervalle $[c,b]$.
  Selon le cas, on met alors à jour la valeur de a ou de b pour se restreindre au sous-intervalle approprié.
- On renvoie finalement la valeur $\dfrac{a+b}{2}$, qui constitue une valeur approchée de α à ε près.



Ce qui donne en Python, en supposant que la fonction admet bien une unique solution dans l'intervalle $[a;b]$:

```python
def dichotomie(f, a, b, epsilon):
    while abs(a - b) > epsilon:
        m = (a + b)/2.
        if f(m) == 0.:
            return m
        elif f(a)*f(m) > 0:
            a = m
        else:
            b = m
    return m
```

REMARQUE **3.1.**
Bien sûr on adaptera le cas où l'on aura à faire à une équation du type $f(x) = n$ en passant par la fonction $g : x \mapsto f(x) - n$.

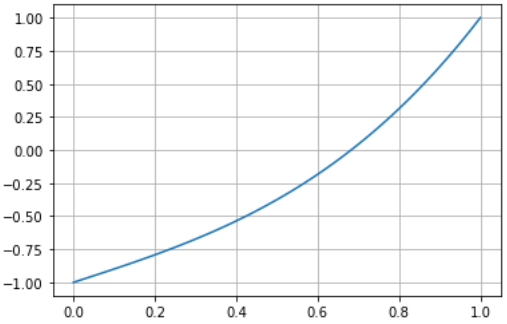
Exemple de type concours

Soit $n \in \mathbb{N}$. On considère la fonction $f_N : x \mapsto x^n + x - 1$ définie.

1. **Etude méthématique :**
   (a) Montrer que l'équation $f_n(x) = 0$ admet une unique solution notée $u_n \in [0;1]$.
   (b) Montrer que $f_{n+1}(u_n) < 0$ puis en déduire que $(u_n)_{n\in\mathbb{N}}$ est croissante.
   (c) Montrer que $(u_n)_{n\in\mathbb{N}}$ converge et tend vers 1. *On pourra utiliser un raisonnement par l'absurde.*
2. **Etude informatique du cas $n = 3$:**
   (a) Ecrire une fonction Python `def f(x,n) :` qui renvoie la valeur $f_n(x)$ lorsqu'on entre la valeur de l'entier n et du réel x.
   (b) On rappelle que la commande `np.linspace(a,b,k)` renvoie un vecteur contenant k nombres allant de a à b espacés uniformément. Compléter la fonction suivante permettant l'affichage graphique ci-contre représentant la courbe de la fonction f_3 ainsi que :

```
import ..... as .....

x= .....
y=[ f(....., .....) for t in x]
plt.plot(x,y)
plt.grid()
plt.show()
```



   (c) Compléter la fonction Python ci-dessous pour qu'elle donne une valeur approchée de u_3 à 10^{-3} près à l'aide de l'algorithme de dichotomie :

```
def approx_u3():
        a=0
        b=1
              while ..................... :
                m = (a + b)/2.
                   if ..................... == 0.:
                        return ..........
                   elif ............................ :
                        a = m
                   else:
                        b = ..............
                return m
```

3. **Cas général:**
   (a) Modifier la fonction `approx_u3()` en une fonction `approx_u(n,eps) :` qui donne une valeur approchée de u_n à `eps` près à l'aide de l'algorithme de dichotomie.
   (b) Expliquer le rôle de la commande suivante : `[approx_u(n,10**(-4)) for n in range(100) ]`
   (c) Ecrire une fonction permettant d'obtenir le graphique ci-dessous représentant graphiquement les valeurs approchées à 10^{-4} près de u_n obtenues à l'aide de la fonction précédente pour n allant de 0 à 100.