Extraction approchée d'une racine cubique

Le problème

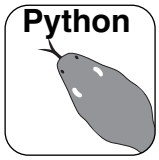
À l'identique de la racine carrée d'un réel positif a (qui est le nombre positif qui élevé au carré donne a), on définit la racine cubique d'un nombre a, qui est le nombre noté $\sqrt[3]{a}$ tel que son cube donne a.
Le but est de trouver une valeur approchée de la racine cubique de a.

On va utiliser une méthode par balayage. On part pour cela d'une valeur initiale x qui s'en approche au mieux par défaut, et l'on calcule x^3. S'il dépasse strictement a, on arrête, sinon on ajoute une quantité à x, jusqu'à dépasser strictement la valeur a.

Complément culturel

L'extraction d'une racine carrée « à la main » a été enseignée aux collégiens jusque dans les années 1950. Dans certains manuels d'arithmétique du XIX^{e} siècle, on trouve également une méthode similaire pour l'extraction d'une racine cubique.

Les programmes



On définit la fonction *racinecubique*() à deux arguments.
Le premier, a, est le nombre dont on veut obtenir une valeur approchée de la racine cubique ;
le second, x_0, est une valeur de départ de l'algorithme.

le programme

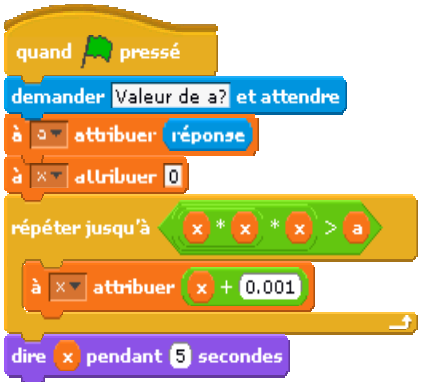
```
>>> def racinecubique(a,x0):
        x=x0
        while x**3<=a:
                x=x+0.00001
        return x
```

le résultat

```
>>> racinecubique(2,1)
1.2599300000017029
```

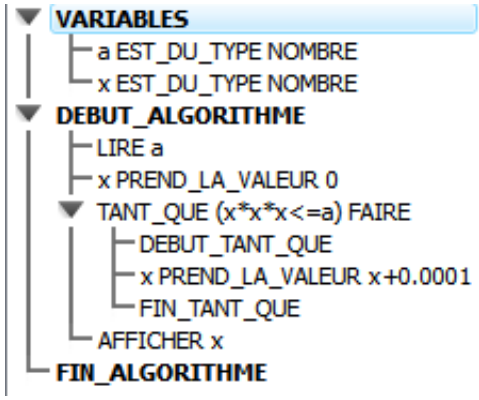
Scratch

Au début du programme Scratch, on ne demande à l'utilisateur d'entrer la valeur de a. On part ensuite de la valeur $x = 0$ (valeur par défaut, on peut la modifier si besoin dans le programme). Le reste est identique au programme Python : on calcule le cube de x et l'on teste si $x^3 > a$. Si c'est le cas, le logiciel affiche la valeur de x, sinon il ajoute 0,001 à x, et recommence.



AlgoBox

On définit les variables a et x au préalable. La variable a est affectée au cours de l'exécution. En revanche on donne la valeur 0 à la variable x. Comme dans les autres programmes, tant que x^3 est inférieur ou égal au réel a, la variable x est incrémentée de 0,0001.



Prolongement*

Isaac Newton (1643–1727) a inventé une méthode générale de résolution numérique des équations que l'on appelle la « méthode de Newton » : pour résoudre l'équation $x^3 = a$, on considère la fonction $f : x \mapsto x^3 - a$. On part d'une approximation de sa racine u_k, qu'on cherche à améliorer de la façon suivante : du point $(u_k ; f(u_k))$ de la courbe représentative de f, on mène la tangente (T) à la courbe, et on assimile la courbe à cette tangente pour en déduire une nouvelle approximation de la racine. Cette nouvelle valeur sera l'abscisse de l'intersection de (T) avec l'axe (Ox). On l'appelle u_{k+1}.
Exécution : la pente de (T) est $f'(u_k) = 3u_k^2$, dérivée de f en u_k.
La suite d'approximations de plus en plus fines sera obtenue à partir de u_0 fixé (si possible relativement proche d'une solution ; dans le programme « newton », on prendra par défaut $u_0 = 1$), et de la relation de récurrence : $u_{n+1} = [u_n - f(u_n)] / f'(u_n)$ qui, pour $f(x) = x^3 - a$, donne $u_{n+1} = (2 / 3)\, u_n + a / 3u_n^2$.

le programme

```
>>> def newton(a,n):
        i=0
        A=1
        while i<n:
                A=1/3*(2*A+a/(A**2))
                i=i+1
        return A
```

le résultat

```
>>> newton(2,20)
1.259921049894873
```