Placement : doubler la mise

Le problème

Lorsque l'on sait qu'une quantité augmente de 3 % par an, il est facile de trouver, quitte à tâtonner, à partir de quelle année la quantité aura doublé.
Comment, en revanche, sans utiliser la fonction logarithme qui ne sera apprise qu'en terminale, savoir quel taux appliquer à une quantité afin qu'elle double par exemple en huit ans ? C'est ce problème que nous vous proposons de résoudre.

La méthode

On part d'une quantité q_0 qui évolue au taux constant de t % (avec $t > 0$).
Après huit ans, la quantité obtenue sera : $q_8 = q_0 \times (1 + t/100)^8$.
On voudrait : $q_8 \geqslant 2\, q_0$.
Avec q_0 positif et non nul, cela revient à résoudre l'inéquation : $(1 + t/100)^8 \geqslant 2$.
La programmation va consister à effectuer un balayage de la droite réelle en partant de $t = 0$, et en l'augmentant à pas constant jusqu'à ce que $(1 + t/100)^8$ soit supérieur ou égal à 2.

Les programmes

Python

Dans le programme « doublement », la condition de sortie de boucle est bien d'avoir t tel que $(1 + t/100)^8 \geqslant 2$.
La syntaxe (1 + t / 100)**8 signifie $(1 + t/100)^8$.

le programme

```
>>> def doublement():
        t=0
        while (1+t/100)**8<2:
                t=t+0.01
        return t

>>> doublement()
9.059999999998513
```

On peut généraliser, et rechercher le taux qui permet de doubler la mise de départ non pas en huit années, mais en N années.
C'est ce que fait le programme « doublementgeneral » ci-dessous.

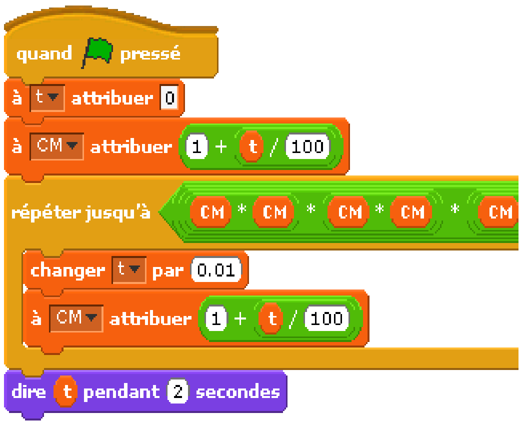
le programme

```
>>> def doublementgeneral(N):
        t=0
        while (1+t/100)**N<2:
                t=t+0.01
        return t

>>> doublementgeneral(10)
7.179999999998914
```

Scratch

La variable annexe CM (pour coefficient multiplicateur) a été créée afin d'améliorer la clarté du programme. Un pas de 0,01 a été choisi pour t ; pour une plus grande précision, il suffit de modifier la ligne de changement de valeur de t à l'intérieur de la boucle. Pour éviter trop de surcharge de syntaxe, la condition « $\geqslant 2$ » a été limitée à « > 2 ».



AlgoBox

Afin d'obtenir plus de lisibilité, le calcul de $(1 + t / 100)^8$ a été décomposé en deux étapes :
- le calcul de CM (CM = $1 + t / 100$) ;
- puis le calcul de CM^8.

La fonction pow est la fonction puissance entière, ainsi pow(CM,8) = CM^8.

```
VARIABLES
  t EST_DU_TYPE NOMBRE
  CM EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  t PREND_LA_VALEUR 0
  CM PREND_LA_VALEUR 1+t/100
  TANT_QUE (pow(CM,8)<=2) FAIRE
    DEBUT_TANT_QUE
    t PREND_LA_VALEUR t+0.001
    CM PREND_LA_VALEUR 1+t/100
    FIN_TANT_QUE
  AFFICHER t
FIN_ALGORITHME
```

Prolongements*

La fonction logarithme népérien, réciproque de l'exponentielle, notée ln, est définie sur $\mathbb{R}^*_+$.
Elle vérifie la relation : pour tous a et b strictement positifs, $\ln(ab) = \ln(a) + \ln(b)$.
Appliquée aux puissances, cela donne : pour tout réel $a > 0$, $\ln(a^n) = n \times \ln(a)$.
En appliquant la fonction ln aux deux membres de l'inéquation $(1 + t / 100)^8 \geqslant 2$, on obtient : $8 \ln(1 + t / 100) \geqslant \ln(2)$, ou encore : $\ln(1 + t / 100) \geqslant \ln(2) / 8$.
En passant à l'exponentielle, cela donne : $1 + t / 100 \geqslant \exp(\ln(2) / 8)$, ou encore : $t \geqslant 100\,[\exp(\ln(2) / 8) - 1]$, valeur appelée « seuil ».
La fonction ln est notée log dans le langage Python.
Voici le calcul du seuil, après importation des fonctions exponentielle et logarithme.

le résultat

```
>>> from math import log
>>> from math import exp
>>> 100*(exp(log(2)/8)-1)
9.050773266525769 9
```