SUITES NUMÉRIQUES PART2 E02

EXERCICE N°5 *Python (n'est pas toujours notre ami…)*

1) Écrire une fonction en Python prenant comme paramètre une liste u , qui retourne True si la liste u contient les premiers termes d'une suite géométrique et False dans le cas contraire.

```
def est_geometrique(L):
    q = L[1]/L[0]
    for i in range(1,len(L)-1):
        if L[i+1]/L[i]!=q:
            return False
    return True
```

2) Voici les premier termes de la suite v .

$\{10000\ ;\ 1000\ ;\ 100\ ;\ 10\ ;\ 1\ ;\ 0{,}1\ ;\ 0{,}01\ ;\ 0{,}001\}$

Vérifier que ce sont bien les premiers termes d'une suite de géométrique dont on précisera la raison.

On doit simplement tester tous les quotients de deux termes successifs.

$$\frac{1000}{10000} = \frac{100}{1000} = \frac{10}{100} = \frac{1}{10} = \frac{0{,}1}{1} = \frac{0{,}01}{0{,}1} = \frac{0{,}001}{0{,}01} = \mathbf{0{,}1}$$

Tous les quotients successifs sont égaux à 0,1.

On en déduit que ce sont bien les premiers termes d'une suite géométrique de raison $q = 0{,}1$

On comprend pourquoi on préfère qu'une fonction fasse tous les calculs à notre place.

3) Que donne cette suite avec la fonction Python ?

```
>>> est_geometrique([10000,1000,100,10,1,0.1,0.01,0.001])
False
>>>
```

Visiblement la fonction ne fonctionne pas toujours…

Mais que se passe t-il ?

On va se servir de la commande « print » pour déboguer...

ici

```
def est_geometrique(L):
    q = L[1]/L[0]
    print("la raison vaudrait : ",q)
    for i in range(1,len(L)-1):
        print("L[",i+1,"] / L[",i,"] = ",L[i+1]," / ",L[i]," = ",L[i+1]/L[i])
        if L[i+1]/L[i]!=q:
            return False
    return True
```

On obtient alors :

```
>>> est_geometrique([10000,1000,100,10,1,0.1,0.01,0.001])
la raison vaudrait :  0.1
L[ 2 ] / L[ 1 ] =  100  /  1000  =  0.1
L[ 3 ] / L[ 2 ] =  10  /  100  =  0.1
L[ 4 ] / L[ 3 ] =  1  /  10  =  0.1
L[ 5 ] / L[ 4 ] =  0.1  /  1  =  0.1
L[ 6 ] / L[ 5 ] =  0.01  /  0.1  =  0.09999999999999999
False
```

Pour faire court, c'est comme si vous faisiez cela :

$$\frac{1}{3}+\frac{1}{3}+\frac{1}{3} = 0{,}33333333+0{,}33333333+0{,}33333333 = 0{,}9999999$$

Le premier « = » est bien sûr faux et vous ne confondez pas un nombre et une de ses approximations. Par contre python le fait et en plus lui travaille en base 2 et pas en base 10 comme nous. Cela fait que pour « lui »