Représentation approximative d'un nombre réel

Les vecteurs suivants $\overrightarrow{u} = \begin{pmatrix} 0,1 \\ 0,3 \end{pmatrix}$ et $\overrightarrow{v} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ sont colinéaires en effet $\overrightarrow{u} = 0,1 \times \overrightarrow{v}$

La fonction suivante devrait retourner la valeur Vraie or ce n'est pas le cas!!!
`print(colineaires(0.1,0.3,1,3))` retourne False

```
def determinant(xU,yU,xV,yV):
    return xU*yV - yU*xV

def colineaires(xU,yU,xV,yV):
    return determinant(xU,yU,xV,yV) == 0

 print(colineaires(0.1,0.3,1,3))
```

Que s'est il passé ?

1. Les nombres ont une représentation **finie** en machine !

Avec les puissances de 10 on a les nombres **décimaux** par exemple $1,75 = 1 + 7 \times 10^{-1} + 5 \times 10^{-2}$

Avec les puissances de 2 on a les nombres **dyadiques**par exemple $1,75 = 1 + 1 \times 2^{-1} + 1 \times 2^{-2} = 1 + \frac{1}{2} + \frac{1}{4}$

On va donc noter $1,75 = (1,11)_2$

Les " premières" puissances de $\frac{1}{2}$ sont dans le tableau suivant :

$(\frac{1}{2})^n = 5^n \times 10^{-n}$

n	1	2	3	4	5	6	7	8
$(\frac{1}{2})^n$	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125	0,00390625

Problème : Tout nombre réel est-il dyadique ?

Théorème 1. *Il existe des nombres réels qui ne sont pas dyadiques, par exemple 0,1*

Preuve Calculons la représentation dyadique de 0,1 :

On cherche n entier naturel tel que $(\frac{1}{2})^n \leqslant 0,1 \leqslant (\frac{1}{2})^{n-1}$

On trouve $n = 4$

Ensuite on calcule 0,1-0,0625 = 0,0375 et on recommence avec 0,0375

Maintenant 0,0375 = 0,03125 + 0,00625 donc pour l'instant $0,1 = (\frac{1}{2})^4 + (\frac{1}{2})^5 + 0,00625$

On continue avec 0,00625, or 0,00625 = 0,00390625 + 0,00234375

On continue et on obtient que 0,1 a **un développement dyadique illimité** et puisqu'une machine est finie la représentation en machine de 0,1 est **tronquée**

$0,1 = (0,0001100110011001100.....)_2$

Retenir

Les calculs sur les nombres flottants ne sont parfois pas exacts

Il ne faut JAMAIS tester une égalité entre deux nombre flottants mais utiliser une marge d'erreur relative.

On corrige donc la fonction `sontColineaires(xU,yU,xV,yV)`

```
>>> def determinant(xU,yU,xV,yV):
        return xU*yV - yU*xV

>>> def sontColineaires(xU,yU,xV,yV):
        return determinant(xU,yU,xV,yV) == 0

>>> print(sontColineaires(0.1,0.3,1,3))
False
>>> print(determinant(0.1,0.3,1,3))
5.551115123125783e-17
```

```
>>> def sontColineaires(xU,yU,xV,yV):
        return abs(determinant(xU,yU,xV,yV)) < 6*10**(-17)

>>> print(sontColineaires(0.1,0.3,1,3))
True
>>> print(sontColineaires(0.1,0.3,1.000000000001,3))
False
```

Exercices

Ex 1

Faire un programme Python qui affiche les 20 premières puissances de $\frac{1}{2}$

Ex 2

Quel est le nombre décimal représenté par $(1010, 1010)_2$

Ex 3

Donner la représentation dyadique du nombre décimal $3,1415$

Ex 4

Existe-t-il des nombres dyadiques qui ne sont pas décimaux ?

Ex 5

Lire le document `https://docs.python.org/fr/3.7/tutorial/floatingpoint.html`