Séquence : *Représentation binaire approximative d'un réels : les flottants.*	**Numérique et Sciences Informatiques**	***Thème :*** *Représentation des données (types de base)*

I – Représentation de la partie à droite de la virgule d'un nombre à virgule

En notation décimale, les chiffres à gauche de la virgule représentent des entiers, des dizaines, des centaines, des milliers, … etc, et ceux à droite de la virgule représentent des dixièmes, centièmes, millièmes, etc.

Ainsi : $3{,}8125_{(10)} = 3\times10^{0} + 8\times10^{-1} + 1\times10^{-2} + 2\times10^{-3} + 5\times10^{-4}$

De la même façon, pour un nombre à virgule en base 2, on utilise les puissances négatives de 2.
Observe cet exemple :

$$\begin{aligned} 11{,}1101_{(2)} &= 1\times2^{1} + 1\times2^{0} + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4} \\ &= 2 + 1 + \frac{1}{2} + \frac{1}{4} + 0 + \frac{1}{16} \\ &= 2 + 1 + 0{,}5 + 0{,}25 + 0 + 0{,}0625 \\ &= 3{,}8125_{(10)} \end{aligned}$$

De la même manière, convertis le nombre suivant en base décimale : $(101{,}010101)_{2,}$

$$(\mathbf{101{,}010101})_{\mathbf{2}} = \mathbf{5{,}328125}.$$

II – Conversion en binaire de la partie à droite de la virgule

1. Observez le schéma ci-contre :

Que permet-il de faire ?

$0{,}375\times10 = \boxed{3}\ ,75$

$0{,}75\ \times10 = \boxed{7}\ ,5$

$0{,}5\ \times10 = \boxed{5}\ ,0$

2. Faites la même chose avec 0,42 et 0,8769. Comment sait-ton que l'on a terminé ?

$0{,}42\ \times10 = \boxed{4}\ ,2$

$0{,}2\ \times10 = \boxed{2}\ ,0$

$0{,}8769\ \times10 = \boxed{8}\ ,769$

$0{,}769\ \times10 = \boxed{7}\ ,69$

$0{,}69\ \times10 = \boxed{6}\ ,9$

$0{,}9\ \times10 = \boxed{9}\ ,0$

On sait qu'il fait s'arrêter lorsqu'il reste 0 à droite de la virgule.

3. Pour obtenir en binaire la partie à droite de la virgule, on procède de la même façon, mais en multipliant par 2 à chaque étape car on travaille alors en base 2.

a. En adaptant le procédé ci-dessus, déterminer la partie à droite de la virgule en base 2 de 0,375.

b. Vérifiez que le résultat obtenu en base 2, lorsqu'il est converti en base 10 redonne bien 0,375.

c. Faire la même chose avec 0,625 puis vérifier le résultat obtenu.

d. Faire de même avec 0,1. Que remarquez-vous ?

$0,375\times 2 = \boxed{0}\ ,75$

$0,75\times 2 = \boxed{1}\ ,5$

$0,5\times 2 = \boxed{1}\ ,0$

a.

$0,625\times 2 = \boxed{1}\ ,25$

$0,25\times 2 = \boxed{0}\ ,5$

$0,5\times 2 = \boxed{1}\ ,0$

$0,375_{(10)} = 0.011_{(2)}$

b. $(0,011)_2 = 2^{-2}+2^{-3} = 0,375$

c. $0,625_{(10)} = 0.101_{(2)}$ $\qquad (0,101)_2 = 2^{-1}+2^{-3} = 0,625.$

d. Développement binaire infini.
On retrouve un reste déjà rencontré. Il y a une période.
On obtient en binaire : 0,00011001100 …

4. ***a.*** On souhaite réaliser un algorithme en langage naturel qui effectue cette conversion.
En observant les conversions faites précédemment, écrire en langage naturel les instructions qui sont répétées.

```
b = a × 2
afficher la partie entière de b
si partie entière de b = 1
    a = b – 1
sinon
    a = b
Fin Si.
```

b. Peut-on prévoir à l'avance combien de fois elles vont être répétées ?
Si non, comment peut-on savoir à quel moment il faudra arrêter de les répéter ?

On ne peut pas savoir. On s'arrête dès que a=0 ou alors quand on atteint un nombre fixé de bits.

c. Rédiger un algorithme en langage naturel pour effectuer la conversion.

5. Voilà un programme Python qui met en œuvre cet algorithme.
Les bits sont stockés dans une chaîne de caractères.

```
from decimal import *

n = Decimal(input("Entrer la partie décimale : "))
en_binaire = ''

i = 0
while (n != 0) and (i < 64):
    n = n*2
    if n >= 1:
        en_binaire = en_binaire + '1'
        n = n - 1
    else:
        en_binaire = en_binaire + '0'
    i = i+1

print(en_binaire)
```

a. Testez le programme avec les valeurs du début de l'activité.
b. Modifiez le programme pour qu'il affiche les calculs intermédiaires comme ci-dessous :

$$0,375 \times 2 = 0\ ,\ 75$$
$$0,75\ \times 2 = 1\ ,\ 5$$
$$0,5\ \times 2 = 1\ ,\ 0$$

Après la ligne « `while (n != 0) and (i < 64):` », écrire « print(« n, "x2=",n*2) ».

6. Reprenez le programme Python précédent et remplacez **Decimal** par **float.**
Reprenez les tests précédents. Que remarquez-vous ?

Problème dans les calculs intermédiaires. Il va falloir se pencher sur l'encodage des nombres à virgule (norme IEEE-754)