REPRÉSENTATION DES NOMBRES (2eme partie)
NOMBRES ENTIERS SIGNES, NOMBRES RÉELS
CORRECTION

1. Représentation binaire d'un entier signé

1.1. Notion de bit de signe

Sur n bits, on code les entiers de -2^{n-1} à $2^{n-1}-1$: On utilise 1 bit pour le signe (il en reste donc n-1) . On peut ainsi coder 2^{n-1} nombres négatifs différents et 2^{n-1} nombres positifs : parmi ces derniers se trouve 0 qui « occupe » un encodage, on va donc jusqu'à $2^{n-1}-1$

2. Représentation approximative des nombres réels

2.1. On teste

>>> a – b = = 0 teste l'égalité des deux nombres a et b (leur différence vaut zéro)

`>>> 4 - 4 = = 0` `>>> 4 - 4.0 = = 0` `>>> 0.2 + 0.8 -1 = = 0` `>>> 0.2 + 0.6 – 0.8 = = 0`

`>>> 0.1 + 0.5 – 0.6 = = 0` renvoient tous *True* mais avec

`>>> 0.1 + 0.2 - 0.3 = = 0` `>>> 0.7 + 0.2 - 0.9 = = 0` le shell renvoie *False* , ce qui n'a pas de sens ! On peut facilement trouver d'autres exemples sans apparente logique....

```
>>>print (abs(0.1+ 0.2 - 0.3)<1E-5)        # test adapté aux flottants
```

renvoie *True* ,ce qui parait plus cohérent. La valeur de 0.1+0.2 est « proche à 10^{-5} près » de 0.3
La fonction Egal (voir fichier Python Egal.py) révèle qu'en fonction des puissances de calcul, l'ordinateur peut encore afficher *False*

2.2. Écriture à virgule flottante

Puissance de 10	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}
Valeur décimale de puissance	1	0,1	0,01	0,001	0,0001
Nombre base 10	3	3	1	2	5

$$3,3125 = 3\times10^{0}+3\times10^{-1}+1\times10^{-2}+2\times10^{-3}+5\times10^{-4}$$

Puissance de 2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Valeur décimale de puissance	2	1	0,5	0,25	0,125	0,0625
Nombre base 2	1	1	0	1	0	1

$$(11,0101)_2 = 1\times2^{1}+1\times2^{0}+0\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4} = 2+1+0+0,25+0+0,0625 = 3,3125$$

2.3. Écriture à virgule flottante en base 2

Un nombre vaut **zéro** si t**ous les bits de son exposant et de sa mantisse sont nuls**

$$0.100000001490116119384765625 + 0.200000002980232238769531 25 = \frac{3}{10} = 0.300000004470348358154296875$$

On peut soustraire à ce résultat la valeur encodée de 0.3 : cela ne donne pas zéro !!

$$0.300000004470348358154296875 - 0.300000011920928955078125 = -0.000000007450580596923828125$$

Le test d'égalité stricte renvoie donc *False*

La représentation de -0.1 n'est pas exacte

1e+100 dépasse la capacité de l'ordinateur : 0 11111111 00000000000000000000000 est l'affichage en binaire (que des 1 sur exposant et des 0 dans la mantisse)

La valeur maximale 2^n-1 de l'exposant est réservée pour les infinis (NaN : not a number en Python)