CORRIGÉ : TP - REPRÉSENTATION DES NOMBRES

Exercice 1 : Représentation des entiers

1. 0100 1100
2.

```
def bin(k,n):
    if n==0:
        return []
    else:
        l = bin(k//2,n-1)
        l.append(k%2)
        return l
```

3. On trouve 211
4.

```
def Horner(L):
    r = 0
    for x in L:
        r = 2*r + x
    return r
```

Exercice 2 : Représentation en complément à 2

1. L'intervalle représentable en complément à 2 sur 8 bits est $[\![-2^7, 2^7-1]\!]$. La représentation de -4 est 11111100 ; pour la trouver on écrit 4 en binaire 00000100, on change les 0 et 1 et vice versa : 11111011 et on ajoute 1, ou on revient à la définition et on écrit $-4+256 = 252$ en binaire.
2.

```
def vers_complement_a_2(k,n):
    if k>0:
        return bin(k,n)
    else:
        return bin(k+2**n,n)

def depuis_complement_a_2(L):
    n = len(L)
    if L[0]==0:
        return Horner(L)
    else:
        return Horner(L) - 2**n
```

3. On trouve 0010 0111.
4.

```
def oppose(L):
    n = len(L)
    for i in range(n-1,-1,-1):
        if L[i] == 1:
            return [1-x for x in L[:i]] + L[i:]
    return L
```

Exercice 3 : Représentation des flottants

1. Voir le cours.
2. Non car 0.1 n'a pas de représentation exacte en python.
3. Il va afficher la plus grande puissance de 10 représentable en Python.
4.

```
a = 1.0
while a/10 != 0.:
    a = a/10
print(a)
```

5. On observe le phénomène d'absorption : pour calculer $a+1$ le processeur aligne les deux nombres avec un exposant commum ce qui repousse le seul chiffre significatif de 1 au delà du 52^e bit, il disparait et $a+1$ et a sont confondus.

Exercice 4 : Entiers multiprécision en Python

1. Ici on ne travaille pas avec des flottants mais avec des entiers multi-précision. Python sait gérer les grands entiers, le programme ne va pas s'arrêter.
2.

```
from time import *
def C(n):
    t1 = time()
    r = 0
    k = 10**n
    for _ in range(1000000):
        r = k+k
    return time() - t1
```

3.

```
Ly = [C(10*n) for n in range(100)]
Lx = [10*n for n in range(100)]
from matplotlib.pyplot import *
plot(Lx,Ly)
```

Exercice 5 : Calculs en complément à 2

1. On trouve :
   (a) 10001100
   (b) 10111001
   (c) 00110011
2. Pour le (b) car on additionne deux nombre positifs et on obtient un nombre négatif.
3.

```
def addition(L1,L2):
    n = len(L1)
    R = [0]*n
    r = 0
    for i in range(n-1,-1,-1):
        m = L1[i]+L2[i]+r
        R[i] = m%2
        r = m//2
    if L1[0] == L2[0] != R[0]:
        print("Attention : depassement arithmetique")
    return R
```

4.

```
def soustraction(L1,L2):
    return addition(L1,oppose(L2))
```

Exercice 6 : Annulation catastrophique

1. On programme :

```
def racines1(a,b,c):
    Delta = b**2-4*a*c
    if Delta > 0:
        s = Delta**(1/2)
        return (-b-s)/(2*a), (-b+s)/(2*a)

for i in range(1,8):
    a= 7*(10**(-i))
    x1, x2 = racines1(a,1/a,-a)
    print('a=', a, ', x1=', x1,', x2=', x2, ', x1*x2=', x1*x2 )
```

et on obtient dans le shell :

```
a= 0.7000000000000001 , , x1*x2= -0.9999999999999999
a= 0.07  , x1*x2= -1.0000000000016107
a= 0.007  , x1*x2= -0.9999999919214783
a= 0.0007  , x1*x2= -0.9999801440413936
a= 7.000000000000001e-05  , x1*x2= -0.0
a= 7e-06 , x1*x2= -0.0
a= 7e-07 , x1*x2= -0.0
```

On remarque qu'à partir de $a = 5 \times 10^{-5}$ le résultat pour le produit des racines est faux.

2. On devrait trouver $\sqrt{2} \times 10^{-14}$ mais on s'aperçoit que le 3^e chiffre significatif est faux.
3. On propose :

```
def racines2(a,b,c):
    b1=b/2
    Delta = b1**2-a*c
    if Delta > 0:
        s = Delta**(1/2)
        if b1 >0:
            x1 = (-b1-s)/(a)
            return x1, c/(a*x1)
        else:
            x2 = (-b1+s)/(a)
            return c/(a*x2), x2
```

et on teste :

```
for i in range(1,8):
    a= 7*(10**(-i))
    x1, x2 = racines2(a,1/a,-a)
    print('a=', a, ', x1=', x1,', x2=', x2, ', x1*x2=', x1*x2 )
```

et on obtient dans le shell :

```
   a= 0.7000000000000001 , x1*x2= -1.0
a= 0.07  , x1*x2= -1.0
a= 0.007  , x1*x2= -0.9999999999999999
a= 0.0007 , x1*x2= -1.0
a= 7.000000000000001e-05 , x1*x2= -1.0
a= 7e-06  , x1*x2= -1.0
a= 7e-07  , x1*x2= -1.0
```

C'est mieux !