Corrigé

Représentation des nombres

Exercice 1 `01101101` débute par un `0` donc représente l'entier positif $(1101101)_2 = 109$.
`10010010` débute par un `1` donc représente l'entier négatif $(10010010)_2 - 2^8 = 146 - 256 = -110$.

Exercice 2

a) $a = 48 = (110000)_2$ est représenté par `00110000` ; $b + 2^8 = 156 = (10011100)_2$ donc b est représenté par `10011100`. On réalise l'addition des représentations :

	0	0	1	1	0	0	0	0
+	1	0	0	1	1	1	0	0
	1	1	0	0	1	1	0	0

Le résultat débute par un `1` donc représente l'entier négatif $(11001100)_2 - 2^8 = 204 - 256 = -52$.

b) $95 = (1011111)_2$ et $42 = (101010)_2$ sont positifs donc représentés par `01011111` et `00101010`. On réalise l'addition :

	0	1	0	1	1	1	1	1
+	0	0	1	0	1	0	1	0
	1	0	0	0	1	0	0	1

Le résultat débute par un `1` donc il représente l'entier négatif $(10001001)_2 - 2^8 = 137 - 256 = -119$; un dépassement de capacité a eu lieu.

Exercice 3 Dans le type `float16` la mantisse est constituée de 10 bits donc le plus petit nombre strictement supérieur à 1 est égal à $(1,0000000001)_2 = 1 + 2^{-10} \approx 1{,}000\,976\,562\,5$. L'epsilon numérique est ici de l'ordre de 10^{-3}.
Pour le type `float32` l'epsilon numérique vaut $2^{-23} \approx 1{,}2.10^{-07}$, et pour le type `float64`, $2^{-52} \approx 2{,}2.10^{-16}$.

Exercice 4 Tout nombre non nul débute par un 1 en base deux, donc pour un tel nombre il suffit de supprimer les 0 de la fin pour qu'ensuite, les espaces binaires soient tous délimités par deux 1 consécutifs.

```
def espacemax(n):
    if n == 0:                # le cas n = 0 est traité séparément
        return 0
    while n % 2 == 0:         # on supprime les 0 de la fin
        n //= 2
    m = 0                     # taille du plus grand espace binaire déjà rencontré
    s = 0                     # taille de l'espace binaire en cours de calcul
    while n > 0:
        n, b = divmod(n, 2)
        if b == 0:            # l'espace binaire en cours se poursuit
            s += 1
        else:                 # l'espace binaire en cours est terminé
            m = max(m, s)
            s = 0
    return m
```

Exercice 5

a) Pour le type `uint16`, `1010101000000000` représente l'entier $2^{15} + 2^{13} + 2^{11} + 2^9 = 43\,520$.

b) Pour le type `int16`, `1010101000000000` représente l'entier relatif $2^{15} + 2^{13} + 2^{11} + 2^9 - 2^{16} = -22\,016$.

c) Pour le type `float16`, `1010101000000000` représente le réel $-(1,1)_2 2^{-5} = -2^{-5} - 2^{-6} = -0{,}046\,875$.

Exercice 6 La variable `r` représente la retenue qui se propage ; en fin d'algorithme elle permet de détecter un éventuel dépassement de capacité.

```
def addition(x, y):
    n = len(x)
    s = [0] * n
    r = 0
    for k in reversed(range(n)):
        (r, s[k]) = divmod(x[k] + y[k] + r, 2)
    if r == 0:
        return s
```