

Corrige_Chapitre16_Flottants

January 21, 2021

1 Exercice 4

```
[35]: def position_virgule(c):  
    """  
    Paramètre : c une chaîne représentant un décimal avec une virgule  
    Valeur renvoyée : position de la virgule  
    """  
    for k in range(len(c)):  
        if c[k] == ',':  
            return k  
  
def ChaineBinDec(c):  
    """  
    Paramètre : c une chaîne de caractères  
    c représentant un nombre à virgule écrit en binaire  
    Valeur renvoyée : nombre flottant correspondant  
    """  
    pos_virgule = position_virgule(c)  
    puissance = 2 ** (pos_virgule - 1)  
    val = 0  
    for bit in c:  
        if bit != ',':  
            val = val + int(bit) * puissance  
            puissance = puissance / 2  
    return val  
  
def ChaineBinDec2(c):  
    """  
    Paramètre : c une chaîne de caractères  
    c représentant un nombre à virgule écrit en binaire  
    Valeur renvoyée : nombre flottant correspondant  
    """  
    partie_entiere = 0  
    i = 0  
    while c[i] != ',':
```

```

    partie_entiere = partie_entiere * 2 + int(c[i])
    i = i + 1
    #print(partie_entiere)
    j = len(c) - 1
    partie_frac = 0
    while c[j] != ',':
        partie_frac = (partie_frac + int(c[j])) / 2
        j = j - 1
    return partie_entiere + partie_frac

```

```

[38]: # Tests
assert ChaineBinDec('1,101') == 1.625
assert ChaineBinDec2('1,101') == 1.625
assert ChaineBinDec('101,001') == 5.125
assert ChaineBinDec2('101,001') == 5.125

```

2 Exercice 6

Convertisseur en ligne : <https://www.h-schmidt.net/FloatConverter/IEEE754.html>

Déterminer les bits de la représentation en double précision du nombre $x = 0,01203125$.

- On détermine d'abord le bit de signe : x est positif donc $s = 0$
- Ensuite on multiplie successivement par 2 pour décaler les bits vers la gauche :

```

[110]: def decimal_vers_IEE754(x, taille_exposant, taille_mantisse):
    #print("détermination du signe")
    if x > 0:
        print("Bit de signe : 0")
    elif x < 0:
        print("Bit de signe : 1")
    else:
        print("0 valeur particulière")
    if x != 0:
        #print("détermination de l'exposant")
        exposant = 0
        while int(x) >= 1:
            x = x / 2
            exposant = exposant + 1
        while int(x) == 0:
            x = x * 2
            exposant = exposant - 1
        decalage = 2 ** (taille_exposant - 1) - 1
        print("Exposant en décimal : ", exposant)
        print(f"Exposant décalé de {decalage} : ", exposant + decalage)
        print(f"Exposant décalé de {decalage} : codage binaire sur 11 bits : ␣
↪", bin(exposant + decalage).lstrip('0b').zfill(taille_exposant))

```

Le développement binaire de 0.01203125 est illimité !

```

Bit de signe : 0
Exposant en décimal : -7
Exposant décalé de + 127 : 120
Exposant décalé de + 127 : codage binaire sur 11 bits : 01111000
Mantisse : 100010100011111010111000

```

[illegible]

```

Bit de signe : 0
Exposant en décimal : -7
Exposant décalé de + 1023 : 1016
Exposant décalé de + 1023 : codage binaire sur 11 bits : 01111111000
Mantisse : 100010100011111010111000010100011

```

```
[103]: # vérification avec le module Decimal
import decimal
decimal.Decimal.from_float(0.01203125)
```

```
[103]: Decimal('0.0120312500000000000277555756156289135105907917022705078125')
```

Un flottant représenté de façon exacte

```
[100]: decimal_vers_IEE754(1/2 + 1/2**2 + 1/2**22, 8,23)
```

```
Bit de signe : 0
Exposant en décimal : -1
Exposant décalé de + 127 : 126
Exposant décalé de + 127 : codage binaire sur 11 bits : 01111110
Mantisse : 10000000000000000000100
```

0,1 n'est pas représenté de façon exacte !

```
[82]: decimal_vers_IEE754(0.1, 11, 52)
```

```
Bit de signe : 0
Exposant en décimal : -4
Exposant décalé de + 1023 : 1019
Exposant décalé de + 1023 : codage binaire sur 11 bits : 0111111011
Mantisse : 10011001100110011001100110011001100110011001100110011010
```

```
[112]: decimal_vers_IEE754(14.5, 8, 23)
```

```
Bit de signe : 0
Exposant en décimal : 3
Exposant décalé de + 127 : 130
Exposant décalé de + 127 : codage binaire sur 11 bits : 10000010
Mantisse : 110100000000000000000000
```

```
[98]: # aujourd'hui 21/01/2021
2021 + 21 / 365
```

```
[98]: 2021.0575342465754
```

```
[99]: decimal_vers_IEE754(2021 + 21 / 365, 8, 23)
```

```
Bit de signe : 0
Exposant en décimal : 10
Exposant décalé de + 127 : 137
Exposant décalé de + 127 : codage binaire sur 11 bits : 10001001
Mantisse : 11111001010000111010111
```

```
[107]: decimal_vers_IEE754(9 + 1/2 + 1/2 ** 2 + 1/2 ** 5, 11, 52)
```

```
[108]: 9 + 1/2 + 1/2 ** 2 + 1/2 ** 5
```

```
[113]: decimal_vers_IEE754(9 + 1/2 + 1/2 ** 2 + 1/2 ** 5, 11, 52)
```

5