Corrige_Chapitre16_Flottants

January 21, 2021

1 Exercice 4

```
[35]: def position_virgule(c):
          Paramètre : c une chaine 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):
          11 11 11
          Paramètre : c une 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 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:</pre>
               print("Bit de signe : 1")
           else:
               print("O 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))
```

```
#print("détermination des bits de mantisse")
x = x - 1
nbits = 0
mantisse = []
while nbits < taille_mantisse:
    x = x * 2
    partie_entiere = int(x)
    mantisse.append(str(partie_entiere))
    if partie_entiere == 1:
        x = x - partie_entiere
    nbits = nbits + 1
print("Mantisse : ", ''.join(mantisse))</pre>
```

Le développement binaire de 0.01203125 est illimité!

```
[111]: # en binary32 : 8 bits d'exposant et 23 bits de mantisse decimal_vers_IEE754(0.01203125, 8, 23)
```

Bit de signe : 0 Exposant en décimal : -7 Exposant décalé de + 127 : 120

Exposant décalé de + 127 : codage binaire sur 11 bits : 01111000

Mantisse: 10001010001111010111000

Vérification avec le convertisseur en ligne https://www.h-schmidt.net/FloatConverter/IEEE754.html

Tools & Thoughts IEEE-754 Floating Point Converter Translations: de This page allows you to convert between the decimal representation of numbers (like "1.02") and the binary format used by all modern CPUs (IEEE 754 floating point).					
IEEE 754 Converter (JavaScript), V0.22					
	Sign	Expor	nent	Mantissa	
Value:	+1	2-		1.5399999618530273	
Encoded as: Binary:	0	12	0 ☑ □ □ □	4529848	
You entered			0.01203125		
Value actually stored in float:			0.012031249701976776123046875		
value actually stored in float.			0.01203124970	15/10/101230406/3	
Error due to conversion:			-2.98023223876953125E-10		
Binary Representation			00111100010001010001111010111000		
Hexadecimal Representation			0x3c451eb8		

```
[106]: # en binary64 : 11 bits d'exposant et 32 bits de mantisse decimal_vers_IEE754(0.01203125, 11, 32)
```

Bit de signe : 0

Exposant en décimal : -7

Exposant décalé de + 1023 : 1016

Exposant décalé de + 1023 : codage binaire sur 11 bits : 011111111000

Mantisse: 10001010001111010111000010100011

```
[103]: # vérification avec le module Decimal
      import decimal
      decimal.Decimal.from_float(0.01203125)
[103]: Decimal('0.012031250000000000277555756156289135105907917022705078125')
     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
     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 : 01111111011
     [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
     [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)
```

Bit de signe : 0

Exposant en décimal : 3

Exposant décalé de + 1023 : 1026

Exposant décalé de + 1023 : codage binaire sur 11 bits : 10000000010

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

[108]: 9.78125

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

Bit de signe : 0

Exposant en décimal : 3

Exposant décalé de + 1023 : 1026

Exposant décalé de + 1023 : codage binaire sur 11 bits : 10000000010