Exercice 1 : Vrai ou Faux

1) En python, il y a plus de nombre de type *int* que de nombres de type $float$.
2) Tous les décimaux appartenant à [1,2] ont une représentation exacte en flottant.
3) Avec la représentation signe, exposant, mantisse, pour effectuer une division par deux en binaire, on retire 1 à l'exposant
4) Avec la représentation signe, exposant, mantisse, si x est une puissance de 2 alors sa mantisse ne contient que des 0.
5) L'expression $1 + 2 ** (-5) == 1$ a la valeur $True$ si le codage des flottants utilise une mantisse codée sur 4 bis
6) Si les flottants sont écrits avec une mantisse codée sur 10 bits alors $1 + 10 ** (-2) == 1$ a la valeur $True$.

1) Oui, le type int utilise des blocs de 64 bits dont le nombre est limité par la capacité de la RAM. Pour une RAM de 16Go, on peut donc atteindre $10^{10^{11}}$. Pour les flottants, les 10bits réservés à aux exposants positifs limitent plus sévèrement la valeur maximale à $2^{2^{10}} \approx 2^{1000} \approx (2^{10})^{100} \approx 10^{300}$
2) Non, on a typiquement une résolution est de l'ordre de $2^{-52} \approx 10^{0.3*52} \approx 10^{-16}$
3) Oui, si $x = (1,m)2^p$ alors $x' = \frac{x}{2} = (1,m)2^{p-1}$ donc on retranche 1 à l'exposant.
   Testons cette affirmations avec 4,4 et 2,2

$$(4{,}4)_{10} = \left(100,\overline{0110}\right)_2 = +1{,}0001\ 1001\ 1001 \ldots 1010 \times 2^2$$

$$(2{,}2)_{10} = \left(10,\overline{0011}\right)_2 = +1{,}0001\ 1001\ 1001 \ldots 1010 \times 2^1$$

4) Oui, cela s'apparente au codage d'un entier
5) Oui car autour de la valeur unité, la résolution est donc de 2**(-4)
6) Faux, car sur 10 bits et autour de 1, la résolution est $2^{-10} \approx 10^{-3}$

Exercice 2 :

On souhaite montrer que l'expression $0{,}2 + 0{,}1 > 0{,}3$ renvoie *True*.

1) Justifier que $(0{,}1)_{10} \approx +1, \overline{1001} \times 2^{-4}$.

En machine, $(0{,}1)_{10}$ est arrondi à la valeur supérieure et la matisse précédente devient (sur 52 bits) :

$$(0{,}1)_{10} \approx (1, \quad 1001 \quad 1001 \quad \dots \quad 1001 \quad 1001 \quad 1010)_2 \times 2^{-4}$$

2) Montrer que $(0{,}2)_{10} \approx +11, \overline{0011} \times 2^{-4}$.

En machine, $(0{,}2)_{10}$ est arrondi et la matisse précédente devient (sur 51 bits) :

$$(0{,}2)_{10} \approx (11, \quad 0011 \quad 0011 \quad \dots \quad 0011 \quad 0011 \ 010)_2 \times 2^{-4}$$

3) Montrer que $(0{,}3)_{10} = +100, \overline{0011} \times 2^{-4}$
4) Calculer 0,1+0,2 et conclure.

Montrons que $0.2 + 0.1! = 0.3$:

- Calculons $x = 0{,}1$

$(0{,}1)_{10} = \left(0{,}00011 \ \overline{0011}\right)_2 = +1, \overline{1001} \times 2^{-4}$

Avec arrondi :

$(0{,}1)_{10} \approx (1, \quad 1001 \quad 1001 \quad \dots \quad 1001 \quad 1001 \quad 1010)_2 \times 2^{-4}$

- Calculons $x = 0.2$:

$(0{,}2)_{10} = \left(0{,}0011 \ \overline{0011}\right)_2 = +1, \overline{1001} \times 2^{-3}$

Avec arrondi :

$(0{,}2)_{10} \approx (1, \quad 1001 \quad 1001 \quad \dots \quad 1001 \quad 1001 \ 1010)_2 \times 2^{-3}$

$(0{,}2)_{10} \approx (11, \quad 0011 \quad 0011 \quad \dots \quad 0011 \quad 0011 \ 010)_2 \times 2^{-4}$

Exprimons 0,3 :

$(0{,}3)_{10} = \left(0{,}01001 \ \overline{1001} \ \right)_2 = +1, \overline{0011} \times 2^{-2}$

$(\ 1, \ 0011 \quad 0011 \quad \dots \quad 0011 \quad 0011 \quad 0011) \times 2^{-2}$

Pas d'arrondi ici car le digit négligé est 0 :

$(\ 1, \ 0011 \quad 0011 \quad \dots \quad 0011 \quad 0011 \quad 0011) \times 2^{-2}$

Calculons :$0.2 + 0.1$:

$$\left|\begin{array}{rrrrrrr} 1, & 1001 & 1001 & \dots & 1001 & 1001 & 1010 \\ 11, & 0011 & 0011 & \dots & 0011 & 0011 & 010 \\ 100, & 1100 & 1100 & \dots & 1100 & 1100 & 1110 \end{array}\right.$$

Soit :

$0.2 + 0.1 = (100, \quad 1100 \quad 1100 \quad \dots \quad 1100 \quad 1100 \quad 1110) \times 2^{-4}$

Avec arrondi et en ne gardant que 52 chiffres significatifs

$0.2 + 0.1 = (1, \quad 0011 \quad 0011 \quad \dots \quad 0011 \quad 0011 \quad 0100)) \times 2^{-4}$

A comparer avec 0,3

Donc $0.2 + 0.1 - 0.3$:

$$\begin{array}{rrrrrrr} 1, & 0011 & 0011 & \dots & 0011 & 0011 & 0100 \\ 1, & 0011 & 0011 & \dots & 0011 & 0011 & 0011 \\ 1, & 0011 & 0011 & \dots & 0011 & 0011 & 0001 \end{array}$$

L'erreur s'explique l'utilisation de plus petit nombre codés avec une meilleure résolution. Soit un écart de l'ordre de $2^{-2-52} = 2^{-54}$

Exercice 3 :

On représente des nombres réels avec une écriture signe, mantisse, exposant. Dans cet exercice, on utilise six bits $b_0b_1b_2b_3b_4b_5$ pour cette écriture.

- Le bit b_0 représente le signe 0 pour + et 1 pour –
- b_1, b_2, b_3 code la mantisse tronquée
- b_4, b_5 code l'exposant uniquement positif (pas d'exposant décalé)

1) Donner les valeurs $b_0b_1b_2b_3b_4b_5$ pour $x = 6{,}0$
2) Quel est le plus petit nombre positif que l'on peut écrire ainsi ?
3) Quel est le plus petit nombre positif que l'on écrire ainsi ?
4) Quel est le successeur de 2,0 ?
5) Combien de nombres appartenant à l'intervalle [1,2[peut-on écrire ?

1) $b_0 = 0$
   Pour la mantisse $(6{,}0)_{10} = (110{,}0)_2 = 1{,}1 \times 2^2$
   $donc\ b_1 = 1, b_2 = 0, b_3 = 0$
   Pour l'exposant : $b_4 = 0, b_5 = 1$
   Donc la représentation flottante associée est 010001
2) Ce plus grand nombre aura pour représentation $011111 \equiv +1{,}111 \times 2^3 \approx 8 + 4 + 2 + 1 = 15$
3) Le plus petit nombre aura pour représentation : $000000 \equiv +1{,}000 \times 2^0 = 1$
4) Il faut trouver le pas pour $x = 2{,}0$
   $(2{,}0)_{10} = (10{,}0)_2 = 1 \times 2^1 \equiv 000001$
   Le dernier digit de la mantisse tronquée est sur 2^{-2} donc la valeur suivante sera $2{,}0 + 2^{-2} = 2{,}25$
5) $(1)_{10} = (1)_2 = (0\ 000\ 00)$ et $(2{,}0)_{10} = (10{,}0)_2 = 1{,}0 \times 2^2 = 0\ 000\ 01$ et donc 8 valeurs possibles :

- 0 000 00
- 0 001 00
- 0 010 00
- 0 011 00
- 0 100 00
- 0 101 00
- 0 110 00
- 0 111 00

la dernière valeur étant 0 111 00 $\equiv +1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 1 + 0{,}5 + 0{,}25 + 0{,}125 = 1{,}875$

Exercice 4 :

Que renvoie le code ci-dessous ?

```
def f():
    x=1.0
    while x!=x+1:
        x=x*2
    return x
```

Nous avons vu que doubler un nombre revient à augmenter d'une unité un exposant donc perdre un ordre de grandeur (en binaire en précision)

$$x = 1{,}0000..\times 2^0 \rightarrow x = 1{,}0000 \times 2^1 \rightarrow \cdots \rightarrow x = 1\ldots.\times 2^p$$

Si on peut écrire $x = 2^p = 1{,}0000\ldots 2^p$ le dernier digit de la mantisse tronqué portera sur $2^{-52+(p-1)}$.

Si $p = 53$ alors le pas vaut 1 et $x == x + 1$ et on trouve $x \approx 10^{16}$

Exercice 5

Ecrire le nombre 3,625 en binaire.

$$(3{,}625)_{10} = (11{,}101)_2 = 1{,}1101 \times 2^1 = 2^1 + 2^0 + 2^{-1} + 2^{-3}$$

Exercice 6 :

On utilise la norme IEEE-754 vue en cours

1) Comment est codé le nombre -4.5 ?
2) Quel est le nombre réel codé par 1011 1111 1110 1000 0000…0000 ?

1) $(-4{,}5)_{10} = (-100{,}1)_2 = -1{,}001 \times 2^2 \equiv 1\ \ 1000000..1\quad 0010\ \overline{0000}$
2) − pour le signe, 0111 1111 1110 correspond à 1022 donc $p = -1$ et donc 1*2^-1+1*2^-2=0,75 et donc le nombre codé est -0,75