TP - Les flottants

Exercice 1 : Vers $+/-\infty$

```
1 def infini_pos():
2     x = 1.
3     n = 0
4     while x != 2*x:
5         n = n + 1
6         x = x*2
7     return n, x
```

Exécuter la fonction ci-dessus.
Expliquer le résultat.
Que se passe-t-il si on remplace `x = 1.` par `x = 1` à la ligne 2 ? Expliquer.
Prévoir ce qui se passe si on remplace `x = 1.` par `x = -1.` à la ligne 2. Vérifier.

Exercice 2 : Vers zéro

```
1 def vers_zero():
2     x = 1.
3     n = 0
4     while x >0:
5         n = n + 1
6         x = x/2
7     return n, x
```

Exécuter la fonction ci-dessus puis expliquer le résultat.
Pourquoi ce résultat n'est pas celui attendu pour la valeur entière obtenu ?
Faire une rapide recherche (wikipédia) sur les nombres dénormalisés.

Exercice 3 : Quelques difficultés usuelles

Reproduire et expliquer :

```
>>> x = 1e200
>>> x*x
inf
>>> (x*x)*0
nan
>>> 0.5+10**400
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    0.5+10**400
OverflowError: int too large to convert to float
```

Exercice 4 : Erreurs de calcul, utilisation de ==

Reproduire et expliquer :

```
>>> 0.1*3
0.30000000000000004
>>> 0.1
0.1
>>> 0.1*2
0.2
```

Prévoir le résultat :

```
>>> 0.1*3 == 0.3
```

Que penser de l'utilisation de == avec les flottants ?

Exercice 5 : Propriétés usuelles de l'addition et de la multiplication

Dans la console, faire les deux calculs suivants :
`(0.1 + 0.9) + 0.7` et `0.1 + (0.9 + 0.7)`
Conclusion ?

Dans la console, faire les deux calculs suivants :
`1.4*3.2 + 1.4*1.6` et `1.4*(3.2 + 1.6)`
Conclusion ?

Exercice 6

On considère la fonction suivante :

```
def div_mult(n):
    x = 1.1
    for k in range(n):
        x = 1 + (x - 1)/3
    for k in range(n):
        x = 1 + (x - 1)*3
    return x
```

1. À l'aide d'un tableau d'état et en supposant que le calculs se font de manière exactes (ce qui n'est pas la cas), déterminer le résultat attendu.
2. Expérimenter. Expliquer.

Exercice 7 : Mauvaise boucle

Pour résoudre un exercice (où il faut compléter le code d'après la docstring) un élève propose le code suivant :

```
def subdivision(a, b, n):
    """ Renvoie une liste de n flottants régulièrement
    répartis entre a (compris) et b (non compris) """

    L = [a]
    x = a
    pas = (b - a)/n

    while x != b:
        x = x + pas
        L.append(x)

    return L
```

Que va-t-il se passer lors de l'exécution de `subdivision(0, 1, 10)` ?
Expliquer
Comment modifier le code ?

Exercice 8 : Un exemple d'amplification d'erreur

Faire afficher le résultat de la fonction pour n allant de 0 à 100.

```
def suite(n):
    (a, b) = (0.2, 0.1)
    for k in range(n):
        (a, b) = (b, 2.5*b -a)
    return a
```

Écouter les explications.