PYTHON POUR LA PHYSIQUE – CHIMIE en 1ère

FICHE n°1 : Réactif limitant, boucle while et fonction

On utilise ici le langage Python pour déterminer l'avancement maximal et identifier le réactif limitant à partir de la donnée des quantités de matière initiales pour une équation de réaction donnée. D'un point de vue chimique, on illustre les notions d'équivalence et de réactif limitant. D'un point de vue algorithmique, c'est l'occasion d'utiliser une boucle **while** et ainsi d'appréhender la gestion d'une condition d'arrêt qui n'est pas connue *a priori*.

Capacité numérique mise en œuvre : Déterminer la composition de l'état final d'un système siège d'une transformation chimique totale à l'aide d'un langage de programmation.

Premier exemple, fonction input, première boucle

La transformation est modélisée par la réaction d'équation :

$$I_2(aq) + 2\ S_2O_3^{2-}(aq) = 2\ I^-(aq) + S_4O_6^{2-}(aq)$$

- Le script débute par la saisie des quantités de matière apportées des deux réactifs (nI2_0 et nS2O3_0). Pour obtenir une activité interactive et dynamique, on fait rentrer ces quantités dans le programme grâce à la fonction **input**. Ainsi, à chaque fois que l'utilisateur lancera le programme, il pourra entrer un nouveau jeu de concentrations initiales. L'instruction **float** convertit simplement les concentrations en flottants pour le traitement.

```
I2_0 = input('quantité initiale en diiode en mol :')
nI2_0 = float(I2_0)
S2O3_0 = input('quantité initiale en thiosulfate en mol :')
nS2O3_0 = float(S2O3_0)
```

- L'avancement est ensuite initialisé à la valeur x = 0 mol, et un incrément est entré (a = 0,001 mol ici).
- Une fonction, 'limitant', initialement vide, aura vocation à lister le nom du (ou des) réactifs limitants selon la composition du mélange initial. Les listes qI2 et qS2O3 à recueillir les quantités de matière successives de ces deux réactifs.

```
limitant =''  # initiallisation de la chaine de caractère correspondant au réactif limitant
x=0  # avancement initial
a=0.001 #pas d'avancement
qI2=[nI2_0]
qS2O3=[nS2O3_0]
```

- Le script opère ensuite par augmentation progressive de l'avancement tant que les deux réactifs sont présents. La boucle conditionnelle **while** permet d'indiquer à Python qu'il doit poursuivre la réaction tant que les quantités de matière des deux réactifs sont positives. La fonction **append** sert à stocker chaque nouvelle valeur de quantité de matière dans les listes créées.

```
while qI2[-1]>0 and qS2O3[-1]>0:
    x=x+a
    qI2.append(nI2_0-x)
    qS2O3.append(nS2O3_0-2*x)
```

- Quand la transformation a été conduite à son terme, le(s) réactif(s) limitant(s) est identifié grâce à la commande liste[-1] qui permet d'appeler la dernière valeur dans une liste. L'avancement final est choisi comme la dernière valeur d'avancement calculée. Enfin, la commande **print** permet d'afficher l'état final. **round** permet d'afficher un arrondi à 2 chiffres significatifs.

```
#résolution du problème et affichage du résultat
if qI2[-1]<=0:
    limitant = 'diiode'
if qS2O3[-1]<=0:
    limitant = 'thiosulfate'
#print(limitant)
print('Le réactif limitant est le ',limitant,'\n Avancement maximum : ',round(x,2),'mol' )
```

Dans les conditions initiales choisies, le script renvoie :

```
quantité initiale en diiode en mol :2
quantité initiale en thiosulfate en mol :3
 Le réactif limitant est le  thiosulfate
  Avancement maximum :  1.5 mol
```

- Il est possible de tester différentes situations initiales avec les élèves pour mettre l'accent sur l'importance des nombres stœchiométriques dans l'identification du réactif limitant (le réactif limitant n'est pas nécessairement celui introduit en plus petite quantité).

Généralisation, première définition, deuxième boucle

On peut maintenant généraliser ce script pour pouvoir gérer tous les coefficients stœchiométriques et toutes les conditions initiales possibles, pour un système siège d'une réaction du type : aA A+ aB B =aC C + aD A

La syntaxe débute par « **def nom_procedure(arguments) :** » et termine par **return**. On reprend les mêmes idées que précédemment.

```
A = input('quantité initiale de A en mol :')
nA = float(A)
B = input('quantité initiale de B en mol :')
nB = float(B)
a = input('a:')
aA = float(a)
b = input('b :')
aB = float(b)
def react_lim(aA,aB,nA,nB) :
    x=0             # Initialisation de l'avancement
    dx=0.00001      # Incrément d'avancement
    qA=[nA]         # Liste stockant les quantités de matière successives de A
    qB=[nB]         # Idem pour B
    RL=[]           # Liste qui stockera le nom du réactif limitant
    while qA[-1]>0 and qB[-1]>0 :
        x=x+dx
        qA.append(nA-aA*x)
        qB.append(nB-aB*x)
    if qA[-1]<=0 :
        RL.append('A')
    if qB[-1]<=0 :
        RL.append('B')
    return(RL,round(x,2))

react_lim(aA,aB,nA,nB)
```

Il suffit alors d'entrer les nombres stœchiométriques et les quantités initiales comme arguments de la procédure et le script renvoie directement le résultat.

```
quantité initiale de A en mol :2
quantité initiale de B en mol :3
a:5
b :2
Out[4]: (['A'], 0.4)
```

Là encore, il peut être formateur de faire constater aux élèves que le réactif limitant n'est pas forcément celui qui est introduit en plus petite quantité.