Calcul d'un tableau d'avancement à l'aide du langage de programmation Python

La réaction étudiée est la combustion du méthane

$$1 \cdot CH_4 + 2 \cdot O_2 \rightarrow 2 \cdot H_2O + CO_2$$

Le tableau d'avancement aura donc l'allure suivante

Équation chimique		$1 \cdot CH_4 + 2 \cdot O_2 \rightarrow 2 \cdot H_2O + CO_2$			
État	Avancement x				
Initial	0	0,54	0,54	0,01	0,0
intermédiaire	x	0,54 - x	0,54 - 2x	0,01 + 2x	0,0 + x
final	$x_{max} = 0{,}27$	0,27	0,00	0,55	0,27

En recopiant et testant les différents codes python suivants, vous découvrirez comment on peut modéliser en informatique ce calcul de quantités de matière et l'évolution du système à l'aide d'un langage de programmation, en utilisant des structures de données listes, des boucles d'instructions, des fonctions et des tests conditionnels.

Script n°1

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce

liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]

print( 'La liste des réactifs : ', liste_reactifs )
print( 'Le premier réactif : ', liste_reactifs[0] )
print( 'Le deuxième réactif : ', liste_reactifs[1] )

print( 'La liste des produits : ', liste_produits )
print( 'Le premier produits : ', liste_produits[0] )
print( 'Le deuxième produits : ', liste_produits[1] )
```

Script n°2

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece

liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]


print(liste_reactifs)
reactif = liste_reactifs[0]
print('nom                           : ', getNom(reactif))
print('coeffficient stoechiometrique : ', getCoef(reactif))
print('Quantité de matière initiale  : ', getQte(reactif))
```

Script n°3

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece
#
# On percoure les listes pour afficher les informatiosn concernants
# toutes les espèces en jeu

liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]


print('Liste des réactifs')

for espece in liste_reactifs:
    print('Nom    : ',getNom(espece))
    print('Coef   : ',getCoef(espece))
    print('Quant. : ',getQte(espece))
    print('      ')

print('Liste des produits')

for espece in liste_produits:
    print('Nom    : ',getNom(espece))
    print('Coef   : ',getCoef(espece))
    print('Quant. : ',getQte(espece))
    print('      ')
```

Script n°4

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece
#
# On écrit une fonction permettant d'afficher l'équation bilan


liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]

def afficherEquation():
    # chaine de caractère qui sera affichée, vide au début
    equation = ""
    # On parcoure la liste des réactifs, pour récupérer le numéro du réactif
    # en commençant à 1, et le réactif (role de enumerate)
    for i , reactif in enumerate(liste_reactifs, 1):
        equation = equation + str(getCoef(reactif)) + ' ' + getNom(reactif)
        # si le réactif n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_reactifs):
            equation = equation + ' + '
    # on dessine la flêche de la réaction
    equation = equation + ' --------> '
    # on ajoute la partie des produits
    for i , produit in enumerate(liste_produits, 1):
        equation = equation + str(getCoef(produit)) + ' ' + getNom(produit)
        # si le produit n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_produits):
            equation = equation + ' + '
    return equation #la chaîne de caractère contient l'équation bilan


# test de la fonction
print( afficherEquation() )
```

Script n°5

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece
#
# On écrit une fonction permettant d'afficher l'équation bilan
#
# On écrit une fonction pour calculer l'avancement maximal x_max


liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]

def afficherEquation():
    # chaine de caractère qui sera affichée, vide au début
    equation = ""
    # On parcoure la liste des réactifs, pour récupérer le numéro du réactif
    # en commençant à 1, et le réactif (role de enumerate)
    for i , reactif in enumerate(liste_reactifs, 1):
        equation = equation + str(getCoef(reactif)) + ' ' + getNom(reactif)
        # si le réactif n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_reactifs):
            equation = equation + ' + '
    # on dessine la flêche de la réaction
    equation = equation + ' --------> '
    # on ajoute la partie des produits
    for i , produit in enumerate(liste_produits, 1):
        equation = equation + str(getCoef(produit)) + ' ' + getNom(produit)
        # si le produit n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_produits):
            equation = equation + ' + '
    return equation #la chaîne de caractère contient l'équation bilan


def calculerXmax():
    # on va calculer pour chaque réactif la valeur de x pour
    # laquelle il disparait et on stocke cette valeur dans une liste
    solutions_possibles = []
    for reactif in liste_reactifs:
```

```
            x = getQte( reactif ) / getCoef( reactif )
            # on ajoute le résultat à la liste des solutions mathématiques
            solutions_possibles.append(x)
        # xmax est la plus petite valeur des solutions disponibles
        xmax = min(solutions_possibles)
        return xmax


print( calculerXmax() )
```

Script n°6

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece
#
# On écrit une fonction permettant d'afficher l'équation bilan
#
# On écrit une fonction pour calculer l'avancement maximal x_max
#
# On écrit une fonction qui calcule l'état du système à l'avancement x


liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]

def afficherEquation():
    # chaine de caractère qui sera affichée, vide au début
    equation = ""
    # On parcoure la liste des réactifs, pour récupérer le numéro du réactif
    # en commençant à 1, et le réactif (role de enumerate)
    for i , reactif in enumerate(liste_reactifs, 1):
        equation = equation + str(getCoef(reactif)) + ' ' + getNom(reactif)
        # si le réactif n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_reactifs):
            equation = equation + ' + '
    # on dessine la flêche de la réaction
    equation = equation + ' --------> '
    # on ajoute la partie des produits
    for i , produit in enumerate(liste_produits, 1):
        equation = equation + str(getCoef(produit)) + ' ' + getNom(produit)
        # si le produit n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_produits):
            equation = equation + ' + '
    return equation #la chaîne de caractère contient l'équation bilan


def calculerXmax():
    # on va calculer pour chaque réactif la valeur de x pour
    # laquelle il disparait et on stocke cette valeur dans une liste
```

```
    solutions_possibles = []
    for reactif in liste_reactifs:
        x = getQte( reactif ) / getCoef( reactif )
        # on ajoute le résultat à la liste des solutions mathématiques
        solutions_possibles.append(x)
    # xmax est la plus petite valeur des solutions disponibles
    xmax = min(solutions_possibles)
    return xmax


def calculerEtat(x):
    # on stocke des valeurs des quantités de matière dans une liste
    liste_qte = []
    for reactif in liste_reactifs:
        qte = getQte( reactif ) - getCoef( reactif ) * x
        liste_qte.append( qte )
    for produit in liste_produits:
        qte = getQte( produit ) + getCoef( produit ) * x
        liste_qte.append( qte )
    return liste_qte


# test de la fonction
x_max =  calculerXmax()
print( x_max)
print(calculerEtat( x_max / 2 ))
print(calculerEtat( x_max ))
```

Script n°7

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece
#
# On écrit une fonction permettant d'afficher l'équation bilan
#
# On écrit une fonction pour calculer l'avancement maximal x_max
#
# On écrit une fonction qui calcule l'état du système à l'avancement x
#
# On ajoute une boucle 'for' pour afficher l'évolution du système
# pour x = 0  à x = x_max pour N valeurs


liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]

def afficherEquation():
    # chaine de caractère qui sera affichée, vide au début
    equation = ""
    # On parcoure la liste des réactifs, pour récupérer le numéro du réactif
    # en commençant à 1, et le réactif (role de enumerate)
    for i , reactif in enumerate(liste_reactifs, 1):
        equation = equation + str(getCoef(reactif)) + ' ' + getNom(reactif)
        # si le réactif n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_reactifs):
            equation = equation + ' + '
    # on dessine la flêche de la réaction
    equation = equation + ' --------> '
    # on ajoute la partie des produits
    for i , produit in enumerate(liste_produits, 1):
        equation = equation + str(getCoef(produit)) + ' ' + getNom(produit)
        # si le produit n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_produits):
            equation = equation + ' + '
    return equation #la chaîne de caractère contient l'équation bilan
```

```
def calculerXmax():
    # on va calculer pour chaque réactif la valeur de x pour
    # laquelle il disparait et on stocke cette valeur dans une liste
    solutions_possibles = []
    for reactif in liste_reactifs:
        x = getQte( reactif ) / getCoef( reactif )
        # on ajoute le résultat à la liste des solutions mathématiques
        solutions_possibles.append(x)
    # xmax est la plus petite valeur des solutions disponibles
    xmax = min(solutions_possibles)
    return xmax


def calculerEtat(x):
    # on stocke des valeurs des quantités de matière dans une liste
    liste_qte = []
    for reactif in liste_reactifs:
        qte = getQte( reactif ) - getCoef( reactif ) * x
        liste_qte.append( qte )
    for produit in liste_produits:
        qte = getQte( produit ) + getCoef( produit ) * x
        liste_qte.append( qte )
    return liste_qte



N = 10
x_max = calculerXmax()
for i in range(0,N):
    x = x_max * i / (N-1)
    print( calculerEtat( x ))
```

Script n°8

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece
#
# On écrit une fonction permettant d'afficher l'équation bilan
#
# On écrit une fonction pour calculer l'avancement maximal x_max
#
# On écrit une fonction qui calcule l'état du système à l'avancement x
#
# On ajoute une boucle 'for' pour afficher l'évolution du système
# pour x = 0  à x = x_max pour N valeurs
#
# On ajoute une fonction de formatage des nombres à virgule  et une
# fonction d'affichage des quantités à l'état x
#


liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]

def afficherEquation():
    # chaine de caractère qui sera affichée, vide au début
    equation = ""
    # On parcoure la liste des réactifs, pour récupérer le numéro du réactif
    # en commençant à 1, et le réactif (role de enumerate)
    for i , reactif in enumerate(liste_reactifs, 1):
        equation = equation + str(getCoef(reactif)) + ' ' + getNom(reactif)
        # si le réactif n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_reactifs):
            equation = equation + ' + '
    # on dessine la flêche de la réaction
    equation = equation + ' --------> '
    # on ajoute la partie des produits
    for i , produit in enumerate(liste_produits, 1):
        equation = equation + str(getCoef(produit)) + ' ' + getNom(produit)
        # si le produit n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_produits):
```

```
            equation = equation + ' + '
    return equation #la chaîne de caractère contient l'équation bilan


def calculerXmax():
    # on va calculer pour chaque réactif la valeur de x pour
    # laquelle il disparait et on stocke cette valeur dans une liste
    solutions_possibles = []
    for reactif in liste_reactifs:
        x = getQte( reactif ) / getCoef( reactif )
        # on ajoute le résultat à la liste des solutions mathématiques
        solutions_possibles.append(x)
    # xmax est la plus petite valeur des solutions disponibles
    xmax = min(solutions_possibles)
    return xmax


def calculerEtat(x):
    # on stocke des valeurs des quantités de matière dans une liste
    liste_qte = []
    for reactif in liste_reactifs:
        qte = getQte( reactif ) - getCoef( reactif ) * x
        liste_qte.append( qte )
    for produit in liste_produits:
        qte = getQte( produit ) + getCoef( produit ) * x
        liste_qte.append( qte )
    return liste_qte


def formater(valeur):
    return "{0:3f}".format(valeur)

def afficherEtat(x):
    listeQte =  calculerEtat(x)
    affichage = ""
    for valeur in listeQte:
        affichage = affichage + formater(valeur) + '       '
    return affichage

N = 10
x_max = calculerXmax()
for i in range(0,N):
    x = x_max * i / (N-1)
    print( afficherEtat( x ))
```

Script n°9

```
# codage de la liste des réactifs et des produits, chaque espèce chimique
# étant une liste comportant le nom, le coefficient stoechiométrique et
# la quantité de matière initiale de l'espèce
#
# des fonctions permettent de récupérer le nom, le coefficient
# stoechimétrique et la quantité de matière pour chaque espece
#
# On écrit une fonction permettant d'afficher l'équation bilan
#
# On écrit une fonction pour calculer l'avancement maximal x_max
#
# On écrit une fonction qui calcule l'état du système à l'avancement x
#
# On ajoute une boucle 'for' pour afficher l'évolution du système
# pour x = 0  à x = x_max pour N valeurs
#
# On ajoute une fonction de formatage des nombres à virgule  et une
# fonction d'affichage des quantités à l'état x
#
# On présente l'ensemble du tableau


liste_reactifs = [
                    [ 'CH4' , 1 , 0.54 ],
                    [ 'O2'  , 2 , 0.54 ]
                ]

liste_produits = [
                    [ 'H2O' , 2 , 0.01 ],
                    [ 'CO2' , 1 , 0.0 ]
                ]


def getNom(reactif):
    return reactif[0]

def getCoef(reactif):
    return reactif[1]

def getQte(reactif):
    return reactif[2]

def afficherEquation():
    # chaine de caractère qui sera affichée, vide au début
    equation = ""
    # On parcoure la liste des réactifs, pour récupérer le numéro du réactif
    # en commençant à 1, et le réactif (role de enumerate)
    for i , reactif in enumerate(liste_reactifs, 1):
        equation = equation + str(getCoef(reactif)) + ' ' + getNom(reactif)
        # si le réactif n'est pas le dernier dans la liste , on ajoute +
        if i < len(liste_reactifs):
            equation = equation + ' + '
    # on dessine la flèche de la réaction
    equation = equation + ' --------> '
    # on ajoute la partie des produits
    for i , produit in enumerate(liste_produits, 1):
        equation = equation + str(getCoef(produit)) + ' ' + getNom(produit)
        # si le produit n'est pas le dernier dans la liste , on ajoute +
```

```
        if i < len(liste_produits):
            equation = equation + ' + '
    return equation #la chaîne de caractère contient l'équation bilan


def calculerXmax():
    # on va calculer pour chaque réactif la valeur de x pour
    # laquelle il disparait et on stocke cette valeur dans une liste
    solutions_possibles = []
    for reactif in liste_reactifs:
        x = getQte( reactif ) / getCoef( reactif )
        # on ajoute le résultat à la liste des solutions mathématiques
        solutions_possibles.append(x)
    # xmax est la plus petite valeur des solutions disponibles
    xmax = min(solutions_possibles)
    return xmax


def calculerEtat(x):
    # on stocke des valeurs des quantités de matière dans une liste
    liste_qte = []
    for reactif in liste_reactifs:
        qte = getQte( reactif ) - getCoef( reactif ) * x
        liste_qte.append( qte )
    for produit in liste_produits:
        qte = getQte( produit ) + getCoef( produit ) * x
        liste_qte.append( qte )
    return liste_qte


def formater(valeur):
    return "{0:3f}".format(valeur)

def afficherEtat(x):
    listeQte =  calculerEtat(x)
    affichage = ""
    for valeur in listeQte:
        affichage = affichage + formater(valeur) + '       '
    return affichage




N = 10
x_max = calculerXmax()
print( afficherEquation() )
for i in range(0,N):
    x = x_max * i / (N-1)
    print( afficherEtat( x ))
print( "l'avancement maximum est "+ formater(x_max) )
```