

DEVOIR SEMESTRE 1**Matière : INFORMATIQUE****Classes : 2^{ème} Année Préparation : BG****Durée : 1 h****Exercice 1**

On souhaite écrire un programme Python qui permet de saisir une chaîne de caractères (sur l'alphabet des 26 lettres de 'a' à 'z') et de prévoir la lettre suivante. Il affiche par la suite la/les lettre(s) la/les plus probable(s) (en termes de fréquence) qui suivra la dernière lettre de la chaîne de caractères saisie. Dans le cas échéant, il renvoie une liste vide.

Exemple d'exécution possible :

```
>>> Saisir une chaîne : ababababab
['a']
>>> Saisir une chaîne : ababacada
['b']
>>> Saisir une chaîne : abbaabbabbacb
['b', 'a']
>>> Saisir une chaîne : a
[]
```

1. Ecrire une fonction **saisir** qui permet de saisir une chaîne de caractères formée que par les alphabets de 'a' à 'z'.
2. Ecrire une fonction **frequence** qui prend en paramètre une chaîne **ch**, extrait **c** le dernier caractère de **ch** et retourne un dictionnaire **D** dont la clé est le caractère qui suit **c** dans **ch** et la valeur est le nombre d'apparition de ce caractère après **c** dans la chaîne **ch**.

Exemple : Pour la chaîne 'abbaabbabbabb', **c**='b' et **D**={'b' : 4, 'a' : 3}

3. Ecrire une fonction **plusProbable** qui prend en paramètre le dictionnaire **D** et retourne une liste formée par la/les clé(s) ayant une valeur maximale dans **D**.

Exemples : Pour **D**={'b' : 3, 'c' : 2, 'a' : 3}, **L**=['b', 'a']

Pour **D**={'b' : 3, 'c' : 2, 'a' : 3, 'd' : 5}, **L**=['d']

Exercice 2

Une molécule est un groupement d'au moins deux atomes qui sont unis par des liens chimiques et elle est représentée par une formule chimique. Une formule chimique est une suite de symboles d'atomes, suivi chacun par un entier qui désigne le nombre d'apparitions (**nbr**) de l'atome dans la molécule. Chaque atome est représenté par la première lettre de son nom en

majuscule, suivie d'une deuxième lettre en minuscule pour différencier des atomes ayant des initiales identiques. Ainsi, le Fluor(**F**) se distingue de Fer(**Fe**) et du Fermium(**Fm**).

Le calcul de la masse molaire d'une molécule, notée $M(\text{Molécule})$, sera comme suit :

- Pour chaque atome de la molécule, calculer le produit ($\text{nbr} * A(\text{atome})$) ou $A(\text{atome})$ est un réel représentant la masse atomique de l'atome ;
- Calculer la somme des produits obtenus.



Exemple :

Pour la molécule dichromate de potassium (**K₂Cr₂O₇**) qui est constituée de 2 atomes de potassium(**K**), 2 atomes de chrome(**Cr**) et 7 atomes d'oxygène(**O**), sa masse molaire moléculaire $M(\text{K}_2\text{Cr}_2\text{O}_7)$ est égale à $2*A(\text{K})+2*A(\text{Cr})+7*A(\text{O})$.

Puisque $A(\text{K})= 39.0983\text{g/mol}$, $A(\text{Cr})= 51.9961\text{ g/mol}$ et $A(\text{O})= 15.9994\text{ g/mol}$, alors $M(\text{K}_2\text{Cr}_2\text{O}_7)= 2*39.0983+2*51.9961+7*15.9994=294.1846\text{g/mol}$

Travail demandé :

Écrire un programme Python contenant les fonctions suivantes :

1. **verifierAtome(atome)** : qui prend en paramètre **atome** de type chaîne de caractères représentant le symbole d'un atome. Cette fonction retourne True si le symbole d'un atome est valide (Première lettre de son nom en majuscule, suivie d'une deuxième lettre en minuscule si c'est le cas) et False sinon.

```
>>> verifierAtome('H')
True
>>> verifierAtome('FE')
False
>>> verifierAtome('Fe')
True
```

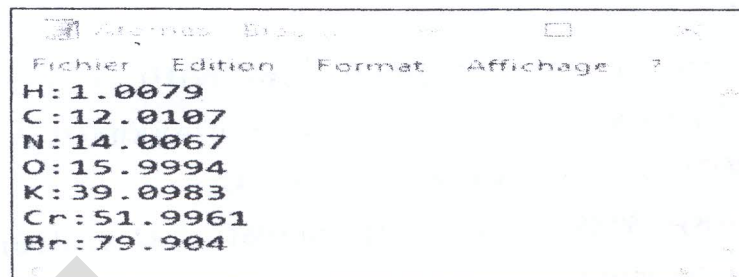
2. **saisirAtome()** : qui permet de saisir par clavier le symbole d'un atome de type chaîne de caractères et sa masse atomique de type réel et les retourne en un tuple (symbole,masse). Cette fonction traite l'exception si le symbole ou la masse atomique est invalide.

```
>>> t=saisirAtome()
Donner le symbole de l'atome : CR
Symbole d'atome invalide !
Donner le symbole de l'atome : Cr
Masse atomique ? er
could not convert string to float: 'er'
Donner le symbole de l'atome : Cr
Donner sa masse atomique : 51.9962
>>> t
('Cr', 51.9962)
```

3. remplirAtomes(N) : qui permet de remplir le fichier 'Atomes.txt' par les données relatives à N atomes saisies par clavier. Chaque ligne du fichier contient le symbole de l'atome et sa masse atomique, séparés par le caractère ':'.

Exemple :

```
>>> remplirAtomes(7) #retourne le fichier 'Atomes.txt' contenant
7 atomes
```



4. construireDic(fich) : qui à partir d'un fichier **fich** stockant les données relatives à des atomes, retourne un dictionnaire dont les clefs sont les symboles des atomes et les valeurs sont leurs masses atomiques correspondantes.

Exemple :

```
>>> atomes=construireDic('Atomes.txt')
>>> atomes
{'H':1.0079, 'C': 12.0107, 'N': 14.0067, 'O': 15.9994, 'K':
39.0983, 'Cr': 51.9961,'Br': 79.904}
```

5. masseMolaire(mol, Dic) : qui à partir du nom d'une molécule **mol** et d'un dictionnaire **Dic** contenant les noms de tous les atomes ainsi que leurs masses atomiques, calcule et affiche la masse molaire moléculaire de la molécule **mol**. Cette fonction doit appeler la fonction **compositionMolecule** donnée en annexe.

Exemple :

```
>>> masseMolaire('K2Cr2O7', atomes)
M(K2Cr2O7) = 294.1846
```

Annexe



On vous donne la fonction **compositionMolecule(mol)** qui prend en paramètre **mol** le nom d'une molécule et retourne un dictionnaire qui possède comme clé le nom de l'atome composant la molécule **mol** et la valeur est le nombre d'apparitions (nbr) de cet atome dans la molécule. On suppose que $1 \leq \text{nbr} \leq 9$.

```
def compositionMolecule(mol):
    D=dict()
    for i in range(len(mol)):
        if 'A'<=mol[i]<='Z':
            if 'a'<=mol[i+1]<='z':
                c=mol[i:i+2]
                if c in D:
                    D[c]+=int(mol[i+2])
                else:
                    D[c]=int(mol[i+2])
            else:
                c=mol[i]
                if c in D:
                    D[c]+=int(mol[i+1])
                else:
                    D[c]=int(mol[i+1])
    return D
```

Exemple d'exécution :

```
>>> compo= compositionMolecule('H2O1')
>>> compo
{'H':2,'O':1}
>>> compo= compositionMolecule('K2Cr2O7')
>>> compo
{'K':2,'O':7,'Cr':2}
>>> compo= compositionMolecule('C1O2H2O1')
>>> compo
{'C':1,'O':3,'H':2}
```