Exercice **Travail sur les lettres, les mots et les textes avec les dictionnaires.**

On veut compter le nombre d'occurrences de chacune des lettres présentes dans un texte. On pourrait faire le compte des "a", puis celui des "b"... mais cela demanderait plusieurs passages... ou alors avoir 26 variables ou une liste à 26 variables... mais c'est lourd!!! sans compter les caractères spéciaux!!!
Avec un dictionnaire on peut le faire en un seul passage de façon assez simple. Pour les tests, nous utiliserons "monTexte" ci-dessous.

```
montexte = """
Il existe un grand nombre de langages de programmation,
chacun avec ses avantages et ses inconvénients. L'idéal serait d'en utiliser plusieurs.
"""

def compteLettres(text):
        """
        entrée : un texte 'tex'
        sortie : un dictionnaire 'dic' avec pour chaque caractère son nombre d'occurrences
        dic = {'a' : ... , "b" :...}
        """
        dic = {}
        ...
        return dic

# application et tests :
lettres = compteLettres(monTexte)
print(lettres)
```

1. **Écrire la fonction** `compteLettres(text)`. Vérifier à la console les valeurs obtenues.

```
>>> lettres['a']
12
>>> lettres['e']
16
```

2. La fonction `compteLettres` prend en compte tous les caractères. Parmi eux on veut pouvoir supprimer du dictionnaire d'occurrence tous ceux qui nous intéressent pas, comme les espaces.

   Écrire une fonction `supprimeCaracteres` qui prend en paramètre un dictionnaire d'occurrences et une liste de caractères à supprimer, et qui supprime du dictionnaire d'occurrence tous les caractères de la liste.

```
def supprimeCaracteres(dic, liste):
        """
        supprime du dictionnaire d'occurrence dic
        les caractères de la liste
        """


        return dic

# tests :
a_supprimer = [' ', '.', ',', ';']
supprimeCaracteres(lettres, a_supprimer )
print(lettres)
```

Vérifier en particulier que les 21 espaces présents dans `montexte` sont bien supprimés...

3. On décide maintenant d'écrire une fonction qui va **simplifier** notre dictionnaire d'occurrences, en particulier en regroupant tous les caractères avec accents, et aussi les majuscules et minuscules... Par exemple, les caractères "e", "é", "è" , "ë" et "E" seront comptés ensemble avec "e".

   Écrire la fonction `simplifieOccurrence(dic, regroup)` qui prend en arguments un dictionnaire d'occurrences `dic`, ainsi qu'un dictionnaire de regroupements `regroup`, et qui renvoie `dicSimple` , un dictionnaire d'occurrences dans lequel on a regroupé les caractères « voisins ».

```
def simplifieOccurrence{dic, regroup}:
        dicSimple = {}
        ...
        return dicSimple

# tests
regroupements = {'e' : ['e', 'é', 'è' , 'ë', 'E']}
lettres_simple = simplifieOccurrence(lettres, regroupements)
print(lettres_simple)
```

   On commencera par recopier `dic` dans `dicSimple` pour éviter de perdre le dictionnaire de départ...

   Vérifier que le test fonctionne, en particulier que tous les caractères autour de « e » sont bien regroupés, puis réessayer en **complétant le dictionnaire de remplacements**.

4. On dispose d'un dictionnaire d'occurrences. On souhaite simplement trouver LE caractère qui a la plus forte occurrence.

   Écrire une fonction `plusForteOccurrence` qui prend comme paramètre ce dictionnaire et renvoie sous forme de couple `(caractere, occurrence)` pour lequel le caractère **présente la plus forte occurrence**.

```
def plusForteOccurrence(dic):
        ...
        return caractere, occurrences
```

   Test : vérifier que `plusForteOccurrence(lettres)` où `lettres` est le dictionnaire obtenu à la question précédente renvoie `('e', 16)`.

5. On souhaite changer la structure des données pour préparer un export au format CVS. Pour l'instant notre dictionnaire d'occurrence à la forme :

```
dict = { 'a' : 16, 'b' : 3, 'c' : 5, ...}
```

   Écrire la fonction `ToCSV(dict)` qui prend comme argument un dictionnaire d'occurrence, et qui renvoie les données sous forme d'une liste de dictionnaires de la forme :

```
data = [
        { 'caractere' : 'a', 'occurrences' : 16} ,
        { 'caractere' : 'b', 'occurrences' : 3} ,
        { 'caractere' : 'c', 'occurrences' : 5} , ...
        ]
```

6. Maintenant mettons un peu d'ordre. En utilisant un algorithme de tri, par exemple le tri par **selection**, écrire une fonction `trierOccurrences` qui prend comme paramètre un dictionnaire d'occurrences et **renvoie les couples** `(caractère, occurrence)` **classé dans un tableau par ordre décroissant d'occurrence**.

   Par exemple `trierOccurrences(data)` doit renvoyer :

```
data = [
        { 'caractere' : 'a', 'occurrences' : 16} ,
        { 'caractere' : 'c', 'occurrences' : 5} , ...
        { 'caractere' : 'b', 'occurrences' : 3} ,
        ]
```

7. On souhaite exporter les données au format CSV, qui aurait la structure suivante :

```
caractere,occurrences
a,18
b,12
...
```

   Écrire le code qui prend ce réaliser cette sauvegarde des données.