1)

a) Écrire la fonction `occurrence(l : list) -> dict` qui prend en argument une liste `l` de valeurs et qui renvoie un dictionnaire donnant, pour chaque valeur apparaissant dans `l`, le nombre de fois qu'elle apparaît dans `l`.

```
print(occurrence([1,3,2,1,4,1,2,1])) # affiche {1: 4, 2: 2, 3: 1, 4: 1}
```

b) Vérifier que la fonction `occurences` codée précédemment fonctionne aussi sur les chaînes de caractères. Que renvoie `occurrences("tagada")` ?

c) Écrire une fonction `plus_frequent(d : dict, k : int) -> str`, prenant en paramètre un dictionnaire `d` contenant des couples (mot, valeur) et qui renvoie le mot de `k` lettres qui est associé à la plus grande valeur. En cas d'égalité, on choisira arbitrairement. S'il n'y a aucun mot de `k` lettres, on renverra une chaîne vide.
Utiliser `occurrence` et `plus_frequent` pour trouver le mot de 5 lettres le plus fréquent dans le fichier `ltdme80j.txt`.

```
f = open("ltdme80j.txt")
texte = f.read().split()
f.close()
```

d) En utilisant la fonction `occurrence`, écrire une fonction `compare_listes(t : list,u : list)` qui détermine si deux listes contiennent les mêmes éléments, avec pour chacun le même nombre d'occurrence.

2) Écrire la fonction `ajoute_dico` prenant en paramètre un dictionnaire `d` et deux valeurs `key` et `value` et ajoute le couple `key-value` au dictionnaire si la clé n'est pas présente dans le dictionnaire. La fonction retourne `True` si le couple a pu être ajouté et `False` sinon.

```
d = {"cool" : 2, 3 : 5}
ajoute_dico(d, "cool", 5)
ajoute_dico(d, 12, 1)
print(d) # affiche {"cool" : 7, 3 : 5, 12 : 1}
```

3) Écrire la fonction `possede_cle` prenant en paramètre un dictionnaire `d` et une valeur `key` qui retourne `True` si `key` est une des clés du dictionnaire et `False` sinon. Il est interdit d'utiliser le mot clé `in`.

```
d = {"cool" : 2, 3 : 5}
print(possede_cle(d, "cool")) # True
print(possede_cle(d, 4)) # False
```

4) Écrire la fonction `minmaxkey` prenant en paramètre un dictionnaire `d` de valeurs numériques et retourne un tuples contenant les clé des plus petites et plus grande valeur.

```
d = {"cool" : 2, 3 : 5, "hot" : -1}
print(minmaxkey(d)) # ("hot", 3)
```

5) Écrire la fonction `merge_dico` prenant en paramètre deux dictionnaires `d1` et `d2` contenant des valeurs numériques qui retourne un nouveau dictionnaire contenant les couples clé-valeur des deux dictionnaires. Si une même clé est présente dans les deux dictionnaires, les valeurs sont additionnées.

```
d1 = {"cool" : 3, 3 : 10}
d2 = {"cool" : 2, 3 : 5, "hot" : -1}
d3 =  merge_dico(d1, d2)
print(d3) # affiche {"cool" : 5, 3 : 15, "hot" : -1} pas forcement dans cet ordre
```

6) Écrire la fonction `sort_by_value_sum` qui prend en paramètre un dictionnaire dont les valeurs sont des listes de nombre et retourne un dictionnaire dont les clés sont triées par la somme des valeurs des listes.

```
d = {'Gfg' : [6, 7, 4], 'best' : [7, 6, 5]}
print(sort_by_value_sum(d)) # affiche {'Gfg': 17, 'best': 18}
```