Etape 2 : Analyse des fréquences des caractères

Pour déchiffrer un message sans en connaitre la clé, la méthode la plus célèbre est **l'analyse de fréquences**. On y considère que la lettre la plus fréquente d'un message est celle la plus fréquente de la langue.

Dans le message que nous avons à déchiffrer, cela signifierait que la lettre 'H' du message correspondrait au 'E'. Toutefois, cette méthode est peu efficace pour des messages aussi courts. Nous allons donc nous appuyer sur le fichier *ducote.txt* qui contient davantage de caractères.

2.1. Ecrire une fonction **frequenceCaracteres** qui :

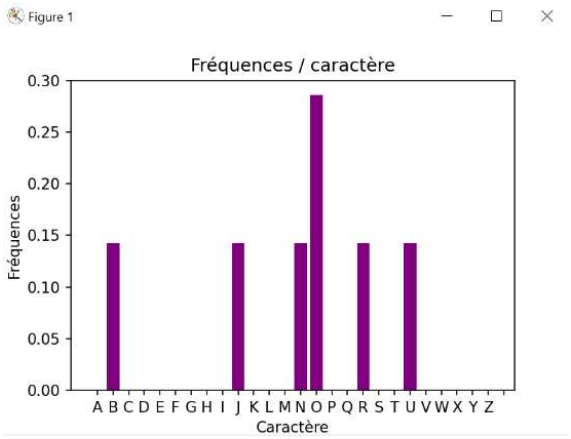
- prend en paramètre une chaine texte,
- et renvoie un tableau de réels comportant la fréquence d'apparition de chaque caractère. A la i^{ème} position de ce tableau, on trouvera la fréquence d'apparition de la i^{ème} lettre de l'alphabet. Vous pouvez utiliser :
  - la fonction count définie sur les listes (https://www.programiz.com/python-programming/methods/list/count)
  - et sum définie sur les tableaux (https://numpy.org/doc/stable/reference/generated/numpy.sum.html)

Exemple sur la chaine de caractères BONJOUR (pas représentative pour la suite) :

```
print ( frequenceCaracteres("BONJOUR") )
"""  affiche :
[0.         0.14285714 0.         0.         0.         0.
 0.         0.         0.         0.14285714 0.         0.
 0.         0.14285714 0.28571429 0.         0.         0.14285714
 0.         0.         0.14285714 0.         0.         0.
 0.         0.        ]
"""
```

Astuce : Pour simplifier le traitement, vous pouvez encoder la chaine en entrée dans la liste d'entiers correspondante et gérer les fréquences des entiers (au lieu de gérer une fréquence de caractères).

Afficher les fréquences des caractères sous forme de diagramme en bâtons.



2.2. Ecrire une fonction **trieFrequencesCaracteres** qui prend en paramètre un tableau de réels représentant les fréquences de caractères et renvoie une liste des lettres les plus présentes vers les moins présentes. Les lettres peuvent être représentées par leur indice (et dans ce cas, on retourne une liste comportant les indices des caractères).

Un tri à bulles ira très bien pour effectuer le tri dans le cadre de cet exercice.

2.3. Pour retrouver le code de chiffrement, il suffit maintenant d'aligner les fréquences des caractères dans le texte complet avec la fréquence des lettres dans le texte chiffré. Compléter le code suivant de la fonction **trouveCle** (zone surlignée) qui à partir des listes triées des fréquences permet de déterminer la clé de chiffrement à l'aide des fréquences :

```
def trouveCle(frequencesTriees_texteRef, frequencesTriees_texte):
    """ 2.3 :
    trouveCle(frequencesTriees_texteRef:list,frequencesTriees_texte:list):list
        entree : 2 listes representant les fréquences triees
                    frequencesTriees_texteRef  pour le texte de reference
                    frequencesTriees_texte pour le texte a dechiffrer
        sortie : liste correspondant à la cle de chiffrage obtenue
    """
    # Calcul des fréquences
    cle = [0 for i in range(26)]
    for i in range(26):
        cle[frequencesTriees_texteRef[i]] = _____ + 1  # +1 pour tenir compte
de l'espace dans l'encodage
    cle = [0, ] + cle  # Ajout de l'espace
    return cle
```

Tester cette clé sur le message chiffré du départ. Qu'obtenez-vous ?