TP9 - Nombre d'occurrences d'un élément dans une liste

Introduction

On considère l'expression ci-dessous qui est une instruction qui demande l'affectation de la valeur **'Bonjour et bienvenue !'** à la variable *chaine*.

- Déterminer à la main sa longueur c'est à dire le nombre de caractères qui la composent.

Saisissez votre réponse ici : La chaine 'Bonjour et bienvenue !' contient ... caractères

- Vérifier votre résultat en utilisant la fonction `len(X)` de Python qui renvoie la longueur d'une chaine ou d'une liste *X*

In [ ]:
```
chaine='Bonjour et bienvenue !'
```

In [ ]:
```
len(chaine)
```

- Compléter le tableau suivant (*en remplaçant les ...*) en comptant (*à la main*) le nombre d'occurrences des lettres et en calculant leurs fréquences (arrondies au millième) c'est-à-dire le nombre de fois que ce caractère apparaît dans la chaine. Attention, tous les caractères comptent !

  Note : Fréquence (car) = Occurences (car) / Longueur (chaine)

	B	b	e	i	j	n	o	r	u	v	!	(espace)
Occurences	1	1	...	...	...	...	...	...	...	...	...	...
Fréquence	0,045	0,045	...	...	...	...	...	...	...	...	...	...

Première méthode : boucle for

Avec Python, il existe une méthode qui permet de parcourir des chaines de caractères. On peut faire décrire à une variable les éléments d'une chaine directement à l'aide d'une boucle de la forme `for x in texte :`

- Compléter la fonction suivante qui utilise cette méthode :

In [ ]:
```
def occurrence_lettre1(lettre,texte):
    '''In : un caractère et une liste de caractères
    Out : la fréquence d'apparition du caractère dans la liste'''
    compteur=0
    for x in texte :
        if ...
            ...
    return ...
```

- Vérifier le bon fonctionnement de votre fonction :

In [ ]:
```
occurrence_lettre1('b',chaine)
```

In [ ]:
```
occurrence_lettre1('o',chaine)
```

Deuxième méthode : indices

On peut accéder à chacune des lettres par leur indice : `chaine[0]` va renvoyer 'B' ... et `chaine[2]` , et `chaine[50]` ?

- Vérifier ce fonctionnement :

In [ ]:
```
chaine[0]
```

In [ ]:
```
chaine[2]
```

In [ ]:
```
chaine[50]
```

- Compléter la fonction suivante qui utilise cette méthode :

In [ ]:
```
def occurrence_lettre2(lettre,texte):
    '''In : un caractère et une liste de caractères
    Out : le nombre d'occurrences du caractère dans la liste'''
    compteur=0
    n=len(texte)  # n est donc la longueur du texte (ou de la liste)
    for i in range(n):
        if ...
            ...
    return ...
```

- Vérifier le bon fonctionnement de votre fonction :

In [ ]:
```
occurrence_lettre2('b',chaine)
```

In [ ]:
```
occurrence_lettre2('o',chaine)
```

In [ ]:
```
occurrence_lettre2('n',chaine)
```

In [ ]:
```
occurrence_lettre2('K',chaine)
```

In [ ]:
```
occurrence_lettre2('n','') # Chaine vide
```

Troixième méthode : count()

Python est vraiment très sympa car il existe une méthode permettant de calculer directement le nombre d'occurrences d'un caractère d'une chaine. C'est la méthode `.count(lettre).` Par exemple `X.count('y')` va renvoyer le nombre de caractères *y* dans *X*.
Noter que les chaînes de caractères doivent être notées entre apostrophes.

- Vérifier le fonctionnement de votre fonction sur des exemples :

In [ ]:
```
chaine.count('e')
```

In [ ]:
```
chaine.count('!')
```

In [ ]:
```
chaine.count('K')
```

Compter le nombre d'occurrences d'un mot dans une chaine de caractères

In [ ]:
```
phrase = "Maître Corbeau, sur un arbre perché, tenait en son bec un fromage."
liste= phrase.split(" ")
```

- Constater le fonctionnement de ce code et le décrire

In [ ]:
```
liste[1]
```

In [ ]:
```
liste[10]
```

Saisissez votre réponse ici : La fonction split ...

- Définir une fonction `compter_mot` qui renverra le nombre de fois que le mot apparaît dans une chaine de caractère :

In [ ]:
```
def compter_mot(mot,texte):
    maListe = ...
    ...
    for ...
        if ...
            ...
    return compteur
```

- Tester votre fonction

In [ ]:
```
compter_mot('sur',phrase)
```

In [ ]:
```
compter_mot('un',phrase)
```

In [ ]:
```
poeme="Maître Corbeau, sur un arbre perché,\
Tenait en son bec un fromage.\
Maître Renard, par l'odeur alléché,\
Lui tint à peu près ce langage :\
Et bonjour, Monsieur du Corbeau.\
Que vous êtes joli ! que vous me semblez beau !\
Sans mentir, si votre ramage\
Se rapporte à votre plumage,\
Vous êtes le Phénix des hôtes de ces bois.\
À ces mots, le Corbeau ne se sent pas de joie ;\
Et pour montrer sa belle voix,\
Il ouvre un large bec, laisse tomber sa proie.\
Le Renard s'en saisit, et dit : Mon bon Monsieur,\
Apprenez que tout flatteur\
Vit aux dépens de celui qui l'écoute.\
Cette leçon vaut bien un fromage, sans doute.\
Le Corbeau honteux et confus\
Jura, mais un peu tard, qu'on ne l'y prendrait plus."
```

In [ ]:
```
compter_mot('et',poeme)
```

In [ ]:
```
compter_mot('Corbeau',poeme)
```

- Quel est le défaut de ce code ?

Saisissez votre réponse ici : La fonction oublie des occurence car ...

En utilisant la fonction replace, on peut retirer de la chaine de caractère toute la ponctuation.
Par exemple pour supprimer les virgules et les remplacer par des espaces d'une variable *texte* il suffit de faire :
`texte.replace(","," ")`

- Reprendre la fonction ci dessus et l'adapter :

In [ ]:
```
def compter_mot2(mot,texte):
    maListe = ...
    ...
    for ...
        if ...
            ...
    return compteur
```

- Vérifier votre modification :

In [ ]:
```
compter_mot2('Corbeau',poeme)
```

Pour aller plus loin

Créez une fonction `frequence(mot,texte)` qui renverra la fréquence (*voir au début du TP*) d'un mot dans un texte.

In [ ]:
```

```