DS n° 02 – Concours blanc

— Faire tous les exercices dans un même fichier `NomPrenom.py` à sauvegarder,
— mettre en commentaire l'exercice traité,
— ne pas oublier pas de commenter ce qui est fait dans votre code,
— il est possible de demander un déblocage pour une question, mais celle-ci sera notée 0.

Découverte du Code César

Le code César est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes. C'est une substitution monoalphabétique, c'est-à-dire qu'une même lettre n'est remplacée que par une seule autre (toujours identique). Le code César a la particularité d'être basé sur un décalage de l'alphabet.

Exemple avec un décalage de (+3) :

Alphabet clair	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alphabet de César (+3)	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Avec cette clef, `CESAR` sera codé `FHPDV`, et `GRULDQ` sera décodé `DORIAN`.
Nous avons reçu le message suivant : `H JXY ZS GTS IJGZY`, chiffré à l'aide d'un code César avec un décalage de (+5). Attention, un espace n'est pas considéré comme un caractère et est laissé à sa place.

1. (a) Proposer un code python permettant de déchiffrer ce message. Attention, il est facile de faire ce décalage « à la main », le code python que vous écrirez sera relu pour vérifier que c'est bien grâce à un code python que le déchiffrement est fait de manière automatique.

   Solution 1.

```
alphabet ='ABCDEFGHIJKLMNOPQRSTUVWXYZ '
message_code='H JXY ZS GTS IJGZY'
n=-5

message_decode=''
for lettre in message_code:
    if lettre!=' ':
        i=alphabet.index(lettre)
        message_decode+=alphabet[(i+n)%26]
    else:
        message_decode+=' '
```

   (b) Afficher le texte déchiffré.

   Solution 2.

```
print(message_decode)
```

Décrypter un message codé

Un message codé a été intercepté. Il a été enregistré dans le fichier `message_code.txt` disponible dans le dossier `Ressources`.

2. (a) Lire ce fichier à l'aide d'une commande python et enregistrer son contenu dans une chaîne de caractères (de type `string`) appelée `message_code`.

   Solution 3.

```
fichier=open('message_code.txt','r')
message_code=fichier.read()
fichier.close()
```

   (b) Afficher les 90 premiers caractères du message. Il devrait apparaître dans votre console, le résultat suivant :

```
GDRG XVG FIIFPKX MX MXBDK
BEFSTXK MX WPG BEFSTXK MX BDKOV
F ZRDP QDS ORPVZRX B XVG XSBDKX
```

Solution 4.

```
print(message_code[:90])
```

On suppose que ce message a été codé à l'aide d'un chiffrage par **substitution**, comme le précédent. Cependant, dans ce cas, une lettre de l'alphabet a été associée à une autre lettre de l'alphabet de manière **aléatoire**. Un exemple de codage utilisant cette technique est présenté ci-dessous.
Exemple : en prenant la clef de substitution suivante :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	Z	E	R	T	Y	U	I	O	P	Q	S	D	F	G	H	J	K	L	M	W	X	C	V	B	N

Le mot `SUBSTITUTION` devient `LWZLMOMWMOGF`.

Il est possible de changer de clef en modifiant l'ordre des lettres de la deuxième ligne. Ainsi, cela permet d'avoir un nombre de 4.10^{26} clefs, ce qui pourrait laisser croire que ce code est difficile à déchiffrer. En effet, le problème de l'exercice est que nous ne connaissons pas la clef qui a réellement servi au chiffrement : c'est une des 4.10^{26} clefs possibles. L'idée est ici d'utiliser une analyse statistique afin de la retrouver. Cette idée vient du constat que dans la plupart des textes écrits en langue française, les lettres les plus utilisées sont souvent les mêmes.

On appellera `clef_a` et `clef_b` les deux lignes du tableau de substitution qui constitue la clef de chiffrement. `clef_a` et `clef_b` devront être écrites dans le script sous un format particulier, à respecter. Pour `clef_a` (l'alphabet en clair), cela donne :

```
clef_a='ABCDEFGHIJKLMNOPQRSTUVWXYZ \n'
```

Il faut donc insérer la ligne précédente dans le code, elle permet de créer une chaîne de caractères (`string`) qui contient toutes les lettres du message codé ou de celui servant pour l'analyse statistique. Attention, il faut qu'un espace apparaisse entre le `Z` et le `\n`.
L'objectif de ce travail est de déterminer `clef_b`, c'est-à-dire la deuxième ligne du tableau de substitution qui a servi à coder le message.
Un poème de Jean Richepin, *La Chanson des gueux* (1881), va servir de base pour déterminer les statistiques d'utilisation des lettres en français. Il est stocké dans le fichier `message_analyse.txt`, présent dans le dossier `Ressources`. S'agissant d'un devoir et non d'un contexte réel de décryptage, des lettres ont été ajoutées à la fin du poème afin de vous permettre de converger plus rapidement vers la solution, il faut les y laisser.

3. (a) Lire ce fichier avec une commande python et enregistrer son contenu entier dans une seule chaîne de caractères (`string`) appelée `message_analyse`.

   Solution 5.

   ```
   fichier=open('message_analyse.txt','r')
   message_analyse=fichier.read()
   fichier.close()
   ```

   (b) Proposer un code python permettant de créer une liste `compteur` qui contient dans l'ordre de la chaîne `clef_a` le nombre d'occurrences de chaque lettre. Le résultat commence comme ceci `[312, 56, 122, 152, 653, 40,...]`, cela signifie que dans le poème, il y a 312 « A », 56 « B », 122 « C », 152 « D », etc. Afficher `compteur`.

   Solution 6.

   ```
   compteur=[0 for i in range(len(clef_a))]
   for lettre in message_analyse:
       i=clef_a.index(lettre)
       compteur[i]+=1
   print(compteur)
   ```

(c) Créer une fonction `analyse_texte(texte)` qui utilise le code de la question précédente et qui renvoie deux listes : une liste contenant les lettres classées par ordre décroissant du nombre d'apparition et une liste contenant les pourcentages d'apparition de chacune des lettres (= 100 x nombre d'occurrences de cette lettre / nombre total de caractères dans le texte). On utilisera ensuite cette fonction de façon à stocker les deux listes renvoyées dans des variables nommées `liste_triee_occurrences` et `liste_stat`

Seules les 15 premières occurrences sont intéressantes (les autres n'étant pas significatives). En faisant un `print(liste_triee_occurrences[:15], liste_stat[:15])` dans le code, on obtient les affichages suivants :

```
[' ','E','S','A','I','L','R','T','N','U','O','\n','M','D','C']
['16.9','12.0','7.0','5.7','5.6','5.6','5.6','5.6','5.6','5.5','5.5','2.9',...]
```

Remarque : pour afficher les flottants (`float`) de façon à obtenir l'exemple ci-dessus, il faut utiliser au moment du remplissage de la liste le code suivant :
`liste_stat.append('%.01f'% valeur_calculee)`
où `valeur_calculee` est la valeur du pourcentage qu'on veut ajouter à la liste.

Solution 7.

```
def analyse_texte(texte):
    compteur=[0 for i in range(len(clef_a))]
    for lettre in texte:
        i=clef_a.index(lettre)
        compteur[i]+=1
    print(compteur)

    liste_triee_occurrences=[]
    liste_stat=[]
    for i in range(len(clef_a)-2):
        m=0
        for i,total in enumerate(compteur):
            if total>m and clef_a[i] not in liste_triee_occurrences:
                m=total
                max=i
        liste_triee_occurrences.append(clef_a[max])
        liste_stat.append('%.01f'% (100*m/len(message_analyse)))
    return liste_triee_occurrences, liste_stat

liste_triee_occurrences, liste_stat=analyse_texte(message_analyse)
print(liste_triee_occurrences[:15], liste_stat[:15])
```

(d) Utiliser la fonction `analyse_texte(texte)` afin de déterminer les lettres qui apparaissent le plus dans le texte codé. On se limitera ici aussi au 15 premières.

```
[' ','X','V','F','P','W','K','G','S','R','D','\n','H','M','B']
['16.3','13.8','7.0','6.9','5.9','5.8','5.2','5.1','5.1','4.9','4.3','3.3',...]
```

Solution 8.

```
liste_triee_occurrences, liste_stat=analyse_texte(message_code)
print(liste_triee_occurrences[:15], liste_stat[:15])
```

On peut donc ici commencer à compléter la `clef_b` en associant à chaque lettre de `clef_a`, la lettre qui apparaît dans le même ordre d'occurrence. Il suffit de faire le lien entre les résultats des questions 3.c et 3.d. On obtient ainsi :

```
clef_a='ABCDEFGHIJKLMNOPQRSTUVWXYZ \n'
clef_b='F-BMX---P--WHSD--KVGR----- \n'
```

Il est alors possible de commencer le décodage du message codé.

4. (a) Créer une fonction `decodage_texte(message)`, qui parcourt le texte lettre par lettre et, si celle-ci se trouve dans la `clef_b`, la remplace par celle correspondant dans la `clef_a`.

Solution 9.

```
def decodage_texte(message,clef_a,clef_b):
    message_decode=''
    for lettre in message_code:
        if lettre in clef_b:
            i=clef_b.index(lettre)
            lettre=clef_a[i]
        message_decode+=lettre
    return message_decode
```

(b) Afficher le texte ainsi obtenu.

Solution 10.

```
print(decodage_texte(message_code,clef_a,clef_b))
```

(c) En parcourant le texte et en analysant le contexte des phrases, proposer de nouvelles correspondances entre la `clef_a` et la `clef_b`. Modifier la `clef_b` et ré-exécuter le code afin de décrypter le message.

Solution 11.

```
clef_b='FQBMXITEPALWHSDOZKVGRCNYJU \n'
```

(d) Afficher à l'aide d'un `print` le nom de l'auteur du message codé.

Solution 12.
print('Louis ARAGON')