Lycée Les 3 Sources **Bourg-Lès-Valence**
NSI 1re Année **2022-23**

Projet Python – César n°2 – correction

Nom et prénom :

Chiffrement de César et force brute

Ce devoir est la suite du TP sur le chiffrement de César. Toutes les fonctions nécessaires se trouvent dans le fichier `cesar2.py` qui est dans `Echange`.
Dans le chiffrement de César, il n'y a que 26 décalages possibles, ce qui est très peu. En pratique, il n'y en a que 25 qui peuvent être utilisés.

Exercice 1 : Expliquer pourquoi un des 26 décalages est inutile, et préciser lequel.
Le décalage de 0 ne change pas le message et est donc inutile.

Pour décoder un message, il est suffisant de tester tous les décalages et de regarder lequel donne un message cohérent. On dit que c'est une attaque par **force brute**.

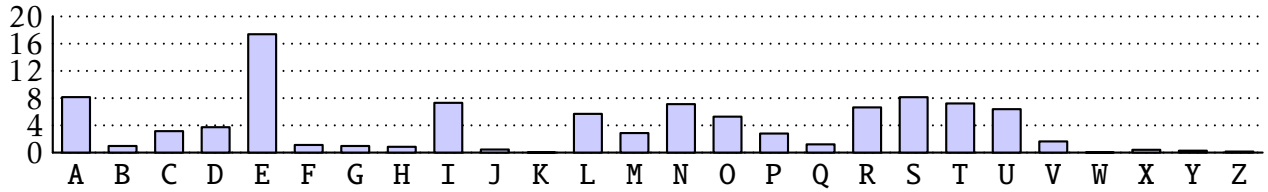
Exercice 2 : Compléter la fonction suivante pour qu'elle teste tous les décalages possibles en affichant la valeur utilisée et le message déchiffré obtenu avec cette valeur. On utilise la fonction `dechiffre(message, clef)` du TP1.

```
def force_brute(message):
    for i in range(26):
        print(i, dechiffre(message, i))
```

Exercice 3 : En utilisant la fonction `force_brute`, déterminer le décalage utilisé pour chiffrer ce message : `"YN SBEPR, VY A'L N DHR PN QR IENV."`. Le décalage est de 13.

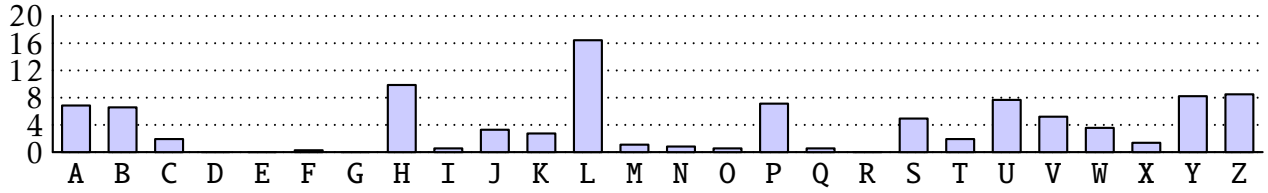
Cryptanalyse du chiffrement de César

La méthode par force brute est facile à utiliser, mais n'est pas très pratique lorsqu'il y a de nombreux messages à décoder. Au IXe siècle, le mathématicien arabe Al-Kindi a eu l'idée d'utiliser les fréquences d'apparition des lettres dans un texte pour décoder un message chiffré. Cette méthode est très adaptée au chiffrement de César. Dans chaque langue, chaque lettre apparaît avec une certaine fréquence. Par exemple, en français, la lettre "e" est la plus courante, avec une fréquence d'environ 17,39 %, alors que le "w" n'apparaît qu'avec une fréquence de 0,03 %.
Le diagramme ci-dessous représente les fréquences des différentes lettres en français.



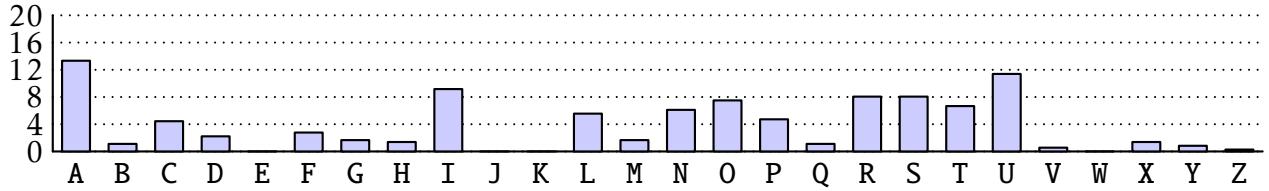
Pour faire une attaque par fréquence, il faut calculer la fréquence d'apparition de chaque lettre dans le message chiffré et essayer d'en déduire le décalage.

Exercice 4 : Le diagramme ci-dessous correspond aux fréquences d'un message codé avec le chiffrement de César. Déterminer le décalage utilisé. Il est de 7.



La méthode des moindres carrés

Déterminer le décalage à l'aide d'histogrammes est facile pour un humain, mais ce n'est pas aussi simple pour un ordinateur.
Une première approche pourrait consister à chercher la lettre la plus fréquente et en déduire que cela correspond au E dans le texte original. Malheureusement, il existe des textes qui n'utilisent pas, volontairement, certaines lettres. On les appelle des **lipogrammes**. L'exemple le plus connu est *La Disparition* de Georges Perec, qui ne contient aucun E. Voici les fréquences des lettres d'un extrait de ce livre :



Comme on peut le voir, même si la fréquence de E est nulle, les autres fréquences sont assez similaires aux fréquences théoriques. Il faut donc une méhode qui permette de prendre en compte les fréquences de toutes les lettres.
La **méthode des moindres carrés** permet de calculer le décalage entre les valeurs théoriques et les valeurs observées. Le principe est le suivant : faire la somme des carrés des différences entre les fréquences observées et les fréquences théoriques. Plus la valeur obtenue est faible, plus les fréquences sont proches des fréquences théoriques.

Exercice 5 : On considère une liste de valeurs `liste = [8, 9, 1, 5, 4]` et deux listes `liste_a = [8, 9, 4, 4, 4]` et `liste_b = [9, 11, 2, 4, 3]`. On souhaite trouver quelle liste est la plus proche de `liste` à l'aide de la méthode des moindres carrés.
On utilise la fonction ci-dessous :

```
def moindres_carres(l1, l2):
    total = 0
    for i in range(len(l1)):
        diff = (l1[i] - l2[i])**2
        total = total + diff
    return total
```

1) À l'aide des tableaux, déterminer la valeur renvoyée pour les deux appels ci-dessous.

`moindres_carres(liste_a, liste)`

i	l1[i]	l2[i]	diff	total
0	8	8	0	0
1	9	9	0	0
2	4	1	9	9
3	4	5	1	10
4	4	4	0	10

Résultat renvoyé : 10

`moindres_carres(liste_b, liste)`

i	l1[i]	l2[i]	diff	total
0	9	8	1	1
1	11	9	4	5
2	2	1	1	6
3	4	5	1	7
4	3	4	1	8

Résultat renvoyé : 8

2) En déduire la liste la plus proche de `liste`. C'est la liste `liste_b`.

Application à la cryptanalyse

Afin d'utiliser une attaque de fréquences par la méthode des moindres carrés, nous allons utiliser un dictionnaire qui associe à chaque lettre sa fréquence dans la langue française :

```
FREQUENCES = {'A': 0.0815, 'B': 0.0097, 'C': 0.0315, 'D': 0.0373, 'E': 0.1739, ...}
```

Ainsi, `FREQUENCES[lettre]` permet d'obtenir la fréquence de `lettre`, si c'est une majuscule.

Exercice 6 : On souhaite compléter la fonction ci-contre qui renvoie un dictionnaire associant à chaque lettre sa fréquence dans `message`. Pour rappel, pour calculer la fréquence, il faut diviser le nombre d'apparitions de la lettre par le nombre d'apparitions de lettres au total dans le message. On ne compte pas les espaces ou les ponctuations. Par exemple, dans `'FSYNSYV PIW KIRW.'`, il y a 14 lettres, dont 2 sont des `'W'`. La fréquence de `'W'` est donc $\frac{2}{14} \approx 0{,}1429$.

1) Calculer la fréquence de la lettre M dans `'MNDG QNDANB.'`.
   Il y a un M sur 10 lettres. Donc la fréquence est 0,1.
2) Compléter la fonction ci-contre.

```
def calcul_frequences(message):
    freq = dict()
    for lettre in ALPHABET:
        freq[lettre] = 0
    compteur = 0
    for symbole in message:
        if symbole in ALPHABET:
            compteur = compteur + 1
            freq[symbole] = freq[symbole] + 1
    for lettre in freq:
        freq[lettre] = freq[lettre] / compteur
    return freq
```

Exercice 7 : Compléter la fonction ci-dessous pour qu'elle renvoie le total des carrés des écarts pour chaque lettre, en comparant la fréquence observée et la fréquence théorique. Vous devez utiliser `FREQUENCES[lettre]`.

```
def moindres_carres_freq(frequences):
    total = 0
    for lettre in frequences:
        total = total + (frequences[lettre] - FREQUENCES[lettre])**2
    return total
```

Exercice 8 : Compléter la fonction ci-dessous qui déchiffre le message avec tous les décalages possibles et qui renvoie le décalage avec le plus petit total avec la méthode des moindres carrés. Avec la méthode des moindres carrés, le total maximal possible est 26.

```
def cryptanalyse(message):
    meilleur_total = 26
    meilleur_decal = 0
    for i in range(26):
        mess = dechiffre(message, i)
        freq = calcul_frequences(mess)
        total = moindres_carres_freq(freq)
        if total < meilleur_total:
            meilleur_total = total
            meilleur_decal = i
    return meilleur_decal
```

```
>>> cryptanalyse('MNDG QNDANB')
9
```

Une application misérable

Le fichier `chapitre_chiffre.txt` contient le premier chapitre des Misérables où chaque paragraphe a été chiffré avec un décalage différent. La fonction `decodage_chapitre()` utilise la fonction `cryptanalyse` pour décoder chaque paragraphe.

```
def decodage_chapitre():
    with open("chapitre_chiffre.txt", "r", encoding='utf8') as f:
        bloc = ""
        liste = []
        for ligne in f:
            bloc = bloc + ligne
            if ligne == '\n':
                dec = cryptanalyse(bloc)
                print(dechiffre(bloc, dec))
                bloc = ""
                liste.append(dec)
        return liste
```

Pour cela, elle lit le fichier ligne par ligne, en construisant un bloc avec toute ces lignes. Les paragraphes sont séparés par une ligne vide, représentée par le symbole `'\n'`.
Vous devez mettre le fichier `chapitre_chiffre.txt` dans le même dossier que `cesar2.py` pour pouvoir utiliser la fonction. Si vos fonctions sont justes, vous devriez obtenir le chapitre décodé.

Exercice 9 : La fonction `decodage_chapitre()` renvoie une liste, qui est vide.

1) Modifier la fonction pour qu'elle renvoie la liste des décalages utilisés pour chiffrer le chapitre. Indiquer dans le code ci-dessus la ligne que vous avez rajouté et à quel endroit.
2) Donner la liste alors obtenue :
   `[20, 1, 17, 7, 20, 6, 15, 13, 23, 19, 1, 4, 5, 5, 4, 15]`