I Echauffement : Cryptage de César

La fonction Python chiffrement ci-dessous prend en paramètre d'entrée une chaine de caractère message constituée uniquement de lettres majuscules sans aucun caractère autre (en particulier sans espaces) et lui applique la méthode de chiffrement dite de César.

```
def chiffrement(message) :
    chiffre = ""
    for c in message : # c comme ...
        x = (num(c) + 3) % 26
        chiffre = chiffre + lettre(x)
    return chiffre
```

`num()` est une fonction qui, à une lettre majuscule, renvoie sa position dans l'alphabet (par exemple A->0, B->1...).
`lettre()` est une fonction qui à tout entier entre 0 et 25 renvoie la lettre correspondante dans l'alphabet en majuscule.

1. Quel traitement effectue cette fonction ?
2. Modifier cette fonction en `cesar(message, decalage)` afin que le décalage soit un paramètre d'entrée.
3. Écrire les fonctions `num()` et `lettre()`
   Pour cela vous pourrez utiliser les fonctions :
   - `ord()` qui prend en paramètre un caractère et renvoie son code ASCII/Unicode
   - `chr()` qui prend en entrée un code ASCII/Unicode et renvoie le caractère correspondant.

   Exemples : `ord('A') -> 65` `chr(70) -> 'F'`
   En Ascii, les majuscules correspondent à des nombres entre 65 (pour 'A') et 90 (pour 'Z'). Les minuscules correspondent à des nombres entre 97 (pour 'a') et 132 (pour 'z'), l'espace correspond à 32.

4. Écrire une fonction `epure(texte)`, où texte est une chaine de caractères, qui renvoie cette chaîne de caractères en majuscule s'ils étaient en minuscules et épurée de tous les autres caractères.

5. Écrire une fonction `dechiffrement_cesar(message,decalage)` qui permet de décoder un message crypté avec la méthode de César en connaissant le décalage.

6. Écrire une fonction `casse_cesar(message)` qui permet de décoder un message crypté avec la méthode de César même sans connaître le décalage.
   Par exemple, décodons le message : `"OHNLTOXSJNTGWFXFXEXIKHYWXFTMAXFTMBJNXLEXIENLVETLLXWNFHGWX"`

Petit aparté culturel :
Au départ, l'encodage des caractères différait d'un pays à l'autre (ou plutôt d'un alphabet à l'autre). Le premier standard fut l'ASCII (nécessitant 7 bits pour encoder tous les caractères nécessaires à un texte en alphabet anglo-saxon). Les ordinateurs travaillant par octet, il restait donc 128 "codes" pour encoder les caractères manquant (La norme Latin-1 en Europe occidentale permettait d'encoder par exemple les lettres accentuées mais pas les caractères cyrilliques ou chinois ...).
Avec la mondialisation des échanges, il y a eu une volonté de créer un standard d'encodage permettant d'encoder tous les caractères du monde : l'Unicode. Deux enjeux se sont imposés. D'abord la nécessité d'une compatibilité avec toutes les données préexistantes. D'autre part, le besoin de ne pas trop augmenter la taille des données. En effet, si l'on veut réserver un nombre fixe de bits par caractères, il en faudrait 4 octets par caractère ce qui ferait exploser inutilement la taille des données pour un texte.
La norme d'encodage Utf-8 est désormais la norme préférentielle pour la plupart des textes courants, parce que, d'une part, elle assure une parfaite compatibilité avec les textes encodés en « pur » ASCII (ce qui est le cas de nombreux codes sources de logiciels), ainsi qu'une compatibilité partielle avec les textes encodés à l'aide de ses variantes « étendues », telles que Latin-1 ; d'autre part, cette nouvelle norme est celle qui est la plus économe en ressources, tout au moins pour les textes écrits dans une langue occidentale. En effet, suivant cette norme, les caractères du jeu ASCII standard sont encodés sur un seul octet. Les autres seront encodés en général sur deux octets, parfois trois ou même quatre octets pour les caractères les plus rares.

II Chiffrement par substitution

La faiblesse de la méthode de cryptage de César réside dans son nombre de décalages possibles faible (25). Même pour un humain, il est facile de tester à la main toutes les possibilités, du fait de ce choix d'une permutation circulaire. Le nombre de possibilités devient plus intéressant si l'on choisit de remplacer une lettre par une autre lettre de l'alphabet sans respecter l'ordre des lettres.
par exemple, en utilisant la chaine `alphabet="QZERTYUIOPASDFGHJKLMWXCVBN"`

1. Écrire une fonction `substitution(message)` qui renvoie une chaîne dans laquelle les lettres A du message ont été remplacées par Q, les B par Z, les C par ...

2. Écrire une fonction `desubstitution(message)` pour décoder un message lorsqu'on connait l'alphabet de substitution.
   indication : On pourra utiliser la méthode pour chaine de caractère `index()` qui renvoie pour un caractère donné sa position dans la chaine.
   par exemple : `alphabet.index('Z') = 1`
   Essayons de déchiffrer le message `"FGFDQOLSTHKGYRTHIBLOJWTOSTLMLBDHQQWLLO"`

   Et quand on ne connait pas l'alphabet de substitution, quel est le nombre de possibilités à tester ?
   Pour autant dès le moyen âge, on savait « casser » la méthode de chiffrement par substitution. En effet, dans chaque langue, certaines lettres ont une fréquence d'apparition caractéristique. Par exemple, en français, la lettre la plus fréquente est le 'E' (environ 16%). Si le caractère 'E' est remplacé toujours par la même lettre ('T' dans l'alphabet de substitution précédent), c'est cette lettre qui présentera l'occurrence la plus forte dans le message chiffré. On arrive ainsi à retrouver les lettres les plus fréquentes et déchiffrer le message. C'est la méthode dite d'analyse des fréquences, d'autant plus efficace que le texte est long.

   Voici les fréquences observées dans un groupement de textes français

Principales lettres	E	A	I	S	T	N	R	U
source Wikipedia, article analyse fréquentielle	15.87	9.42	8.41	7.90	7.26	7.15	6.46	6.24
analyse fréquentielle sur les articles en français Wikipedia	12.10	7.11	6.59	6.51	5.92	6.39	6.07	4.49
analyse fréquentielle observée								

3. Écrire une fonction `frequence(texte)` qui prend en entrée une chaine de caractère texte et renvoie une liste `L` contenant les fréquences d'apparition des différentes lettres de l'alphabet dans texte. `L[0]` correspondra à la fréquence d'apparition du caractère 'A', `L[1]` à celle de 'B' ...

Mémento sur les chaines de caractères

Une chaine de caractère (*string*, abrégé en `str` en Python)s'écrit entre quote simples ou doubles
```
chaine1='Bob Morane'
chaine2="l'aventurier"
```
les fonctions de base (souvent communes aux tableaux) :

- `len(chaine)` donne le nombre de caractères de la chaine
- `print(chaine)` affiche la chaine
- + est l'opérateur de concaténation `print(chaine1 + chaine2)  -> "Bob Moranel'aventurier"`

Elles ne sont pas mutables. Ce sont des itérables (c'est à dire que les caractères qui les contiennent sont indicés en commençant par 0 !) donc

- On peut lire la valeur d'un caractère si l'on connait son indice (à indiquer entre crochets, comme pour les tableaux)
  ```
  chaine1[0]   -> 'B'
  chaine1[5]  -> 'o'
  ```
- On peut en parcourir tous les éléments dans l'ordre de manière simplifiée avec un `for`
  ```
  for e in chaine1:
  print(e)
  ```
- On peut tester simplement si un caractère ou une sous chaine est contenue dans la chaine
  ```
  if 'o' in chaine1:
  print("la chaine contient un o")
  ```

Des méthodes bien pratiques

- la méthode `index` prend en paramètre un caractère (ou une sous-chaine) et renvoie la position de sa première occurrence (et une erreur si elle n'y est pas)
  `chaine1.index('o')   -> 1`
- la méthode `upper()` convertit tous les caractères en majuscule et `lower()` convertit tous les caractères en minuscules
  `chaine1.upper()   -> 'BOB MORANE'`
- la méthode `replace(old,new)` remplace tous les caractère `old` par des caractères `new`
  et tellement d'autres ...

remarque : les chaines de caractères sont munies d'un ordre, l'ordre alphabétique ascii (les majuscules sont toutes plus petites que les minuscules)
```
'Bob'>'Alice' -> True
'Bob'>'alice' -> False
```
La fonction `ord(caractere)` renvoie le code ascii décimal du caractère, la fonction `chr(nombre)` donne le caractère connaissant son code ascii (décimal)

Mémento sur matplotlib.pyplot

D'abord, il faut importer la librairie (en la renommant, ici pl) : `import matplotlib.pyplot as pl`

Il y a différentes commandes :

- `pl.plot(X,Y)` pour créer des points (`X` est le tableau des abscisses et `Y` celui des ordonnées, de même taille). On peut spécifier la valeur des paramètres :
  - `color = 'blue'` ou `'red'` ...
  - `marker = 'o'` ou `'+'` ...
  - `linestyle = 'dashed'`(pointillés) ou `'none'` (ne pas relier les points)...
- `pl.bar(X,Y)` pour faire des diagrammes en bâtons, `pl.scatter(X,Y)` pour faire des nuages de points
- `pyplot.xlim(a, b)` pour graduer les abscisses entre a et b, `pyplot.ylim(a, b)` pour graduer les ordonnées
- `pyplot.title('Titre à indiquer')` pour donner un titre
- `pl.grid()` pour faire afficher un quadrillage
- `pl.savefig(chemin)` pour enregistrer la figure

Il ne faut pas oublier de faire afficher le graphique créé : `pl.show()`

TP

Aller sur le site http ://www.angelique-renaud.com et récupérer

— le fichier code du tp que vous complèterez au fur et à mesure

— les fichiers document.txt et messagecrypte.txt

1. Écrire une fonction `frequence(texte)` qui prend en entrée une chaine de caractère texte (supposée en majuscule) et renvoie une liste `L` contenant les fréquences d'apparition des 26 différentes lettres de l'alphabet dans texte. `L[0]` correspondra à la fréquence d'apparition du caractère 'A', `L[1]` à celle de 'B' ...
indication : On pourra créer une liste contenant 26 zéros au départ : `L = [0]*26`
2. Compléter le code du TP afin de lancer l'analyse fréquentielle du document fourni et faire afficher les résultats sous forme d'un histogramme.

```
monfichier = open ( ..... ,"r")
#mettez l'adresse de votre fichier texte avec des antislashs
contenu = monfichier. .............
#on recupere le contenu du fichier sous forme d'une seule chaine de caractere
monfichier.close()

import matplotlib.pyplot as pl  #import de la librairie graphique
X = range(26)
Y = ..........................................
pl.bar(X,Y)   #commande pour creer des diagrammes en barre
Z=[lettre(i) for i in range(26)]
pl.xticks(X,Z)  #commande pour mettre des etiquettes sur l'axe des abscisses
..................... #commande pour montrer le graphique
```

On peut se servir de la méthode d'analyse des fréquences pour attaquer un code de César de manière plus efficace. Il suffit de rechercher quelle lettre est devenue la plus fréquente. Comme elle correspond à 'E', on obtient le décalage utilisé.

3. Ecrire une fonction decalage(texte) qui renvoie le probable décalage utilisé si le texte a été codé par la méthode de César.

Mémento sur les lectures/écritures de fichiers

D'abord, il faut ouvrir un lien avec le fichier avec la fonction `open(chemin,mode)`, donc connaître le chemin d'accès au fichier !
Il y a différents mode d'accès au contenu d'un fichier :

- `'r'` pour lire (read)
- `'w'` pour écrire (write) : attention le précédent contenu du fichier est écrasé ; si le fichier n'existe pas, il est créé
- `'a'` (pour append/ajouter) pour écrire dans le fichier à la suite du contenu précédent ; si le fichier n'existe pas, il est créé

Il ne faut pas oublier de fermer le lien avec la méthode `close` !
Pour récupérer tout le contenu dans une seule chaine de caractère, on peut utiliser la méthode `read`. Les sauts de lignes sont traduits par le caractère `\n`.
Pour lire ligne par ligne (on s'arrête au caractère `\n`), on peut utiliser la méthode `readlines` qui renvoie une liste contenant chaque ligne. La méthode `readline` (sans le s au bout) renvoie une ligne du fichier sous forme de chaine, et lors de chaque appel suivant, elle renverra la ligne suivante. pour écrire on utilise la méthode `write(chaine)`

III Chiffrement de Vigenère

Inventé au 16ème siècle par un diplomate français en mission en Italie, il faut longtemps considéré comme inviolable. La méthode de chiffrement de Vigenère est une amélioration de la méthode de césar puisqu'on utilise plusieurs alphabets décalés pour chiffrer un message. La table de Vigenère donne tous les alphabets décalés :

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

On choisit une clef de cryptage, par exemple : MATHS
Pour la première lettre du message on utilisera l'alphabet de substitution "M" (où A donne M), pour la seconde on utilisera l'alphabet de substitution "A" (où A donne A)...

Message clair	R	D	V	V	E	N	D	R	E	D	I
Clef	M	A	T	H	S	M	A	T	H	S	M
Message crypté											

La lettre V est donc successivement codée par deux lettres différentes, ce qui rend le chiffrement plus résistant à l'analyse des fréquences.

1. Écrire à partir des fonctions précédentes une fonction `vigenere(message,clef)`
   Néanmoins sur un texte long, certains mots courants (les, une, des, pour, est ...) finissent par "retomber" sur les mêmes morceaux de clefs. On observe donc des groupements de lettres récurrents (qui reviennent). Cela implique que la clef a été répétée un nombre entier de fois entre deux occurrences d'un même groupement de lettre. La taille de la clef est donc un multiple commun aux écarts entre deux mots récurrents.
2. Voici un message crypté par la méthode de Vigenère, cherchez à la main des groupements de lettres récurrents :

   LVQZWTFRGRYQHSWIPXKVQZWXDJGJWHMWNVGBWPUVTMHSWIPXKWGZKGWYTJRTARUUEGK
   PGLVLVIOVYKWWFVTBWVTVRXMJRFYKDIUYSFEIUDSKLINJHXKXKEILSYVIIISWWETHMVNVW
   XAKPVYKLIPVFKWYZJYKKSPJSFTVGKVHFIFRRLDIRRCLVIOFVWGVQJIMWRFVRMDIUFQUJIU
   LRTFRGRYIGYTCILYSWMIKFITKSNKYPRRGWEWGSNJPGJXKGYXVVNFEPEITMTQLVEWWCDI
   GWVVFYLWXFRRLDIUKRUJIUCILDMGIENHEAJHXESTUSKGYUVXXFHGEXEWWQDFKWW

   Observez leurs écartements et en déduire la longueur probable de la clef.

   Il suffit donc maintenant d'effectuer une analyse fréquentielle de toutes les lettres codées par la première lettre de la clé (puis de la seconde...).
3. Écrire une fonction `extrait(texte,d,c)` qui prend en entrée une chaine de caractère chaine et deux entiers d et c (correspondant à la longueur de la clef). Elle renverra les lettres de la chaine codées par la lettre d de la clef.
   Dans l'exemple précédent, la clef MATHS ayant une longueur de 5,
   `extrait("RDVVENDREDI",0,5)` renverra `"RNI"` lettres correspondant à M (position 0) dans la clef
   `extrait("RDVVENDREDI",1,5)` renverra `"DD"` lettre correspondant à A (position 1) dans la clef
4. Écrire un script qui tente de décoder le message, fourni dans le fichier `messagecrypte.txt`.
   Pour aller plus loin Créer un fichier contenant le message décrypté.
5. *Pour aller plus loin* Chercher à écrire une fonction qui cherche des groupements de lettres récurrents et leur espacement.