Codage de Vigenère

Au XVIème sicèle, Blaise de Vigenère a modernisé le codage de César très peu résistant de la manière suivante. Au lieu de décaler toutes les lettres du texte de la même manière, on utilise un texte clé qui donne une suite de décalages.

On cherche à crypter un texte t de longueur n composé de caractères (26 lettres différentes) représentés par des entiers compris entre 0 et 25 :

$$0 \leftrightarrow \texttt{a} \ , \ 1 \leftrightarrow \texttt{b} \ , \ \ldots \ , \ 25 \leftrightarrow \texttt{z}$$

Prenons par exemple la clé `satire`. Pour crypter un texte, on code la première lettre en utilisant le décalage qui envoie le `a` sur l `s` (la première lettre de la clé). Pour la deuxième lettre, on prend le décalage qui envoie le `a` sur le `a` (la seconde lettre de la clé) et ainsi de suite. Pour la sixième lettre, on utilise le décalge de `a` vers `e`, puis, pour la septième, on reprend la clé à partir de sa première lettre. Sur l'exemple `monsieurlafontaine` avec la clé `satire`, on obtient : (la première ligne donne le texte, la seconde le texte crypté et la troisième la lettre de la clé utilisée pour le décalage.

m	o	n	s	i	e	u	r	l	a	f	o	n	t	a	i	n	e
e	o	g	a	z	i	m	r	e	i	w	s	f	t	t	q	e	i
s	a	t	i	r	e	s	a	t	i	r	e	s	a	t	i	r	e

1. Définir une variable globale :

```
1 alphabet = "abcdefghijklmnopqrstuvwxyz"
```

Coder Vigenère

2. Que donne le codage du texte `becunfromage` en utilisant la clé de codage `jean` ?
3. Définir une fonction `En_numero` qui prend en argument une chaine de caractère `texte` et qui renvoie une liste des numéros associés à chaque caractère.
4. Définir une fonction `En_chaine` qui prend en argument une liste d'entiers entre 0 et 25 et qui renvoie la chaine de caractère associée.
5. Définir une fonction `CodageVigenere` qui prend en arguments un tableau t de taille n représentant le texte à crypter et un tableau d'entiers c de longueur k donnant la clé au codage et qui retourne un tableau de taille n contenant le texte crypté t'.
6. Donner le codage du texte long suivant en utilisant la clé de codage `lafontaine`.

```
1 Corbeau_Renard="maitrecorbeausurunarbreperchetenaitensonbecunfromagemaitrerenardpa
2 rlodeurallecheluitintapeuprescelangagehebonjourmonsieurducorbeauquevousetesjolique
3 vousmesemblezbeausansmentirsivotreramageserapportavotreplumagevouseteslephenixdesh
4 otesdecesboisacesmotslecorbeaunesesentpasdejoieetpourmontrersabellevoixilouvreunla
5 rgebeclaissetombersaproielerenardsensaisitetditmonbonmonsieurapprenezquetoutflatte
6 urvitaudepensdeceuxquilecoutecetteleconvautbienunfromagesansdoutlecorbeauhonteuxet
7 confusjuramaisunpeutardquonnelyprendraitplus"
```

Craquer Vigenère

Maintenant, on suppose disposer d'un texte t' assez long crypté par le méthode de Vigenère et on veut retrouver le texte t d'origine. Pour cela, on doit trouver la clé c ayant servi au codage. On procède en deux temps :

- Détermination de la longueur k de la clé c ;
- Détermination des lettres composant c.

Pour craquer le code de Vigenère, nous allons devoir calculer des indices de coïncidence. Pour deux chaînes `s1` et `s2` de même taille, l'indice de coïncidence est le pourcentage de positions où les lettres de `s1` et `s2` sont identiques. Par exemple : pour abracadra et raplapla ont un indice de coïncidence de 12.5% :

```
1 abracadabra
2 raplapla
```

Il y a seulement une lettre sur 8 qui coïncide dans les deux chaînes. Les lettres en trop sont simplement ignorées.

7. Définir une fonction `indice_coincidence` qui prend en argument deux chaînes de caractères `s1` et `s2` et qui renvoie l'indice de coïncidence des deux chaînes.

Sur deux chaînes aléatoires, l'indice de coïncidence est d'environ 3.85% (un vingt-sixième). Pour deux textes écrits en langue naturelle, l'indice de coïncidence est plus élevé. Pour le français, il est supérieur à 7%. Ceci vient du fait que les lettres ne sont pas distribuées aléatoirement. Par exemple, comme il y a de nombreux `e` dans des textes en français, il y a un peu plus de chances que des `e` se retrouvent en même positions dans `s1` et `s2`.

En particulier, si `s` est une chaîne en français, l'indice de coïncidence entre `s` et `s[k:]` doit être supérieur à 7%.

D'autre part, l'indice de coïncidence entre deux chaînes n'est pas modifié si les caractères des deux chaînes ont subit le même décalage. Par contre, l'indice de coïncidence entre deux chaînes où les caractères ont subit deux décalages différents sera de l'ordre de 3.85%.

Pour découvrir la taille de clé, nous allons utiliser le test Friedman. Si `s` est une chaîne codée avec le code de Vigenère, il faut calculer les indices de coïncidence entre `s` et `s[1:]`, `s` et `s[2:]`, `s` et `s[3:]`, ...

Lorsqu'on calcule l'indice de coïncidence entre `s` et `s[k:]` où `k` est la taille de la clé, les lettres alignées ont subit le même décalage. L'indice de coïncidence doit donc être supérieur à 7% ! Dans les autres cas, l'indice de coïncidence sera nettement plus proche de 3.85%...

8. Définir une fonction `Test_friedman` qui prend en arguments une chaîne de caractère `texte` et un entier `decalage_max` et qui renvoie les indices de coïncidences successifs jusqu'à `decalage_max`.
9. Tester la fonction sur des chaînes cryptées avec la fonction `CodageVigenere` pour voir que les indices de coïncidence sont effectivement différents quand le décalage est un multiple de la taille de la clé.

Une fois que l'on connait la taille de la clé, c'est presque gagné : on peut faire une analyse statistique sur les lettres.

Pour ceci, il faut calculer les fréquences d'apparition des lettres par paquets. Si la taille de la clé est de 5 :

- fréquences de toutes les lettres sur les positions 0, 5, 10, 15, ...
- fréquences de toutes les lettres sur les positions 1, 6, 11, 16, ...
- fréquences de toutes les lettres sur les positions 2, 7, 12, 17, ...
- fréquences de toutes les lettres sur les positions 3, 8, 13, 18, ...
- fréquences de toutes les lettres sur les positions 4, 9, 14, 19, ...

Si le texte original est en français, la lettre la plus fréquente sera le `e`, pour tous les paquets de lettres. Ainsi, si sur le premier la lettre la plus courante est `u`, c'est que le décalage était de 16 lettres (`e`+16 = `u`), et donc que la première lettre de la clé était `p` (seizième lettre de l'alphabet).

10. Rédiger une fonction `Frequence` qui prend en argument une chaîne de caractère `texte` et retourne un tableau de 26 cases contenant la fréquence d'apparition de chacune des 26 lettres de l'alphabet dans ce texte.
11. Rédiger une fonction `En_groupes` qui prend en arguments une chaîne de caractère `texte` et entier `k` (représentant la taille de la clé) et qui renvoie une liste de chaînes de caractère des paquets de lettres comme expliqué ci-dessus.
12. Définir une fonction `devine_cle` qui prend en arguments une chaîne de caractère `texte` et un entier `k` (représentant la taille de la clé) et qui renvoie les fréquences des lettres sur les `k` paquets de lettres de `chaine` et calcule la clé la plus probable comme expliqué ci dessus.
13. A partir des différentes fonctions précédentes, définir une fonction principale `Craque_Vigenere` qui prend en argument une chaîne de caractère `texte` et qui renvoie la clef utilisée pour chiffrer ce texte.