Projet Info2

Cryptographie - César et Vigénère

La cryptographie a pour but de cacher le contenu d'un message afin de le rendre incompréhensible par un utilisateur qui n'en connaît pas la clé. On brouille alors le message initial en suivant un protocole mis au point préalablement par l'expéditeur et le destinataire. Le destinataire n'aura qu'à inverser le procédé pour rendre le message lisible.
L'ennemi, s'il ne connaît pas le protocole de brouillage (c'est-à-dire la clé) ne pourra pas naturellement rétablir le texte original.

1. Chiffrement par décalage et code de César

Le chiffre de César doit son nom à Jules César qui, selon Suétone l'utilisait avec un décalage de trois sur la droite pour certaines de ses correspondances secrètes, notamment militaires.

Le chiffre de César consiste à assigner à chaque lettre de l'alphabet une autre lettre, résultant du décalage de l'alphabet d'un certain nombre de lettres. Pour connaître le code utilisé, il suffit de connaître la clé, c'est-à-dire la lettre qui correspond à la lettre `a`.

Par exemple avec une clé égale à 5, on obtient le tableau de correspondance suivant :

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
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

1. Traduire en utilisant le code de César associé à la clé 5 le texte suivant : `La physique c est fantastique`.

.....

.....

1.1. Codage des caractères en Python

2. Rappeler le type d'un caractère ou d'une chaîne de caractère en langage Python.

.....

.....

Dans ce TP nous allons utiliser les fonctions suivantes `ord` et `chr` :

Python
```
ord('a')
    97
```

Python
```
chr(97)
    'a'
```

3. Décrire succinctement le rôle de ces deux fonctions et afficher les lettres dont les codes sont compris entre 97 et 122.

.....

.....

.....

.....

4. Ecrire et documenter la fonction `ordre` qui prend en argument un caractère (de type `string`) et qui renvoie son code simplifié, c'est-à-dire un entier entre 0 et 25 tel que `ordre('a') = 0` et `ordre('z') = 25`.

5. Ecrire et documenter la fonction réciproque `lettre` qui prend en argument un entier compris entre 0 et 25 et qui renvoie la lettre correspondante : `lettre(0)= 'a'` et `lettre(25) = 'z'`

1.2. Codage César et décodage

6. Ecrire une fonction `code_cesar` qui prend en argument un texte (chaîne de caractères) et une clé (nombre entier) et qui renvoie le texte codé.
La chaîne de caractères est constituée de lettre minuscules et ne contient ni caractères spéciaux ni caractères accentués. La chaîne de caractères peut contenir des espaces ' '. La fonction `code_cesar` ne modifie pas ces espaces.

7. Tester la fonction sur un message de votre choix. Puis créer un fichier texte contenant le message codé (d'une dizaine de mots minimum).

8. Ecrire et documenter une fonction `decode_cesar` qui prend en argument un texte crypté et la clé et qui renvoie le texte original. Tester la fonction sur le texte précédent et vérifier que l'on récupère le texte initial.

2. Formatage du texte à coder

La fonction `code_cesar` est adaptée à une chaîne de caractères minuscules non accentués, sans ponctuation ni caractères spéciaux.
Le but de cette partie est de formater une chaîne de caractères quelconque afin de pouvoir lui appliquer la fonction `code_cesar`.

Cahier de charges : *la fonction `formatage` doit renvoyer une chaîne de caractères dans laquelle :*

- *les majuscules sont remplacées par des minuscules,*
- *les caractères accentués sont remplacés par leur équivalent sans accent. (exemple `à` → `a`),*
- *les signes de ponctuation (et tous les autres caractères spéciaux) sont remplacés par des espaces.*

De plus, la fonction `formatage` devra par défaut enlever les espaces ' ' grâce à l'argument `espace = False` mais pourra permettre de les conserver en passant cet argument à `True` :

Python
```
formatage('Bonjour à toi Ô Atom PT')
    'bonjouratoioatompt'
```

Python
```
formatage('Bonjour à toi Ô Atom PT',True)
    'bonjour a toi o atom pt'
```

Pour réaliser la fonction `formatage`, on pourra s'aider des méthodes adaptées aux chaînes de caractères.

9. Ecrire et documenter la fonction `formatage`.

3. Comment craquer le code de César ?

On intercepte un message codé par le chiffrement de César dont on ignore la clé. On veut déterminer par une méthode automatique la clé et le message original.

Nous allons exploiter l'idée que, dans une langue donnée, la fréquence d'apparition de chacune des lettres de l'alphabet n'est pas la même.

10. Ecrire une fonction `frequence` qui prend en argument un texte français (formaté, c'est-à-dire un texte écrit en minuscules sans lettres accentuées ni caractères spéciaux) et qui renvoie une liste contenant la fréquence de chacune des lettres de l'alphabet.
Appliquer cette fonction au texte `reference` disponible dans le dossier de la classe. Vérifier que la liste de fréquences obtenue correspond au tableau ci-dessous. En cas de problème ou de manque de temps, cette liste est disponible dans le fichier `frequence_ref.py`.

a	b	c	d	e	f	g	h	i	j	k	l	m
0.055	0.0081	0.0197	0.0329	0.099	0.0079	0.0099	0.0079	0.053	0.0021	0.0004	0.0373	0.023

n	o	p	q	r	s	t	u	v	w	x	y	z
0.063	0.049	0.0284	0.0054	0.047	0.051	0.060	0.0412	0.0116	0.0014	0.0035	0.0079	0.0004

On note fr la liste des fréquences de référence en français et fc la liste des fréquences des lettres de la séquence codée.
Pour un entier i donné, la quantité $\left|fr_j - fc_{(i+j)\,mod(26)}\right|$ est l'écart entre la fréquence de la lettre d'indice j en français (fr_j), et la fréquence de cette même lettre dans le texte codé en faisant l'hypothèse que la clé est i. La quantité d_i est la somme de tous ces écarts pour la clé i fixée.

$$d_i = \sum_{j=0}^{25} \left|fr_j - fc_{(i+j)\,mod(26)}\right| \qquad i \in \{0, 1, \ldots, 25\}$$

Donc plus d_i est petit, plus le tableau des fréquences avec la clé i est proche du tableau des fréquences de référence en français. i correspond alors à la clé du code de César quand d_i est minimum.

11. Ecrire et documenter une fonction `distance` qui prend en argument deux listes de fréquence (la liste des fréquences de référence en français, et la liste des fréquences du texte codé) et qui renvoie la liste des distances $d = [d_0, d_1, \ldots, d_{25}]$.

12. Ecrire et documenter une fonction `craquer_cesar` qui utilise la fonction `distance` ainsi que d'autres fonctions précédemment programmées et qui renvoie la clé utilisée pour la séquence codée.

13. Ecrire un script permettant de décoder le fichier `texte_secret` et de créer un nouveau fichier contenant le message en clair.

4. Pour aller plus loin : chiffrement de Vigénère

Le chiffre de Vigénère est une amélioration du chiffre de César. Sa force réside dans l'utilisation non pas d'un décalage unique pour toutes les lettres du message mais d'une clé qui va définir un décalage différent pour chaque lettre du message. Ce décalage se répète (on voit apparaître la notion de fréquence qui sera utilisée pour craquer ce type de codage).
Par exemple, on peut établir le tableau de correspondance suivant en cherchant à coder `informatique` avec la clé `piston`.

Message	i	n	f	o	r	m	a	t	i	q	u	e
Clé	p	i	s	t	o	n	p	i	s	t	o	n
Décalage	15	8	18	19	14	13	15	8	18	19	14	13
Message codé	x	v	x	h	f	z	p	b	a	j	i	r

Dans toute la suite, on pourra utiliser les fonctions précédemment définies.

4.1. Codage et décodage

14. Ecrire une fonction `code_vigenere` qui prend en argument un texte (chaîne de caractères) et une clé (chaîne de caractères) et qui renvoie le texte codé.
La chaîne de caractères est constituée de lettre minuscules et ne contient ni caractères spéciaux ni caractères accentués. La chaîne de caractères ne contiendra pas d'espaces.

15. Tester la fonction sur un message de votre choix. Puis créer un fichier texte contenant le message codé (d'une dizaine de mots minimum). Copier ce fichier texte dans le dossier `Documents en partage` de la classe pour qu'il puisse être décodé ou craqué par la suite.

16. Ecrire et documenter une fonction `decode_vigenere` qui prend en argument un texte crypté et la clé et qui renvoie le texte original. Tester la fonction sur le texte précédent et vérifier que l'on récupère le texte initial.

4.2. Comment craquer le code de Vigenere ?

On intercepte un message codé par le chiffrement de Vigénère dont on ignore la clé. Ce code est difficile à casser à la main (surtout si la clé est longue). Nous allons appliquer une méthode en 2 étapes pour trouver la clé et retrouver le texte original.

- ◇ Evaluation de la longueur de la clé
- ◇ Découverte de la clé.

On suppose que la clé est constituée de 5 à 20 caractères minuscules. On suppose que le texte intercepté est suffisamment long.

- **Longueur de la clé**

  Cette méthode de chiffrement est dite polyalphabétique, contrairement au code de César qui est monoalphabétique : avec le chiffre de César, chaque lettre est toujours changée en la même lettre chiffrée. Ainsi l'analyse fréquentielle permet de retrouver simplement la clé d'un code de César, mais n'est plus adaptée au décryptage d'un texte codé par le chiffre de Vigénère.

William Friedman a proposé une méthode reposant à nouveau sur une analyse fréquentielle pour déterminer la longueur de la clé. Il a a défini l'indice de coïncidence, qui mesure la probabilité que, prenant deux lettres au hasard dans un texte, ce soit les mêmes. Cet indice se calcule ainsi :

$$\mathrm{IC} = \sum_{x=a}^{x=z} \frac{n_x\,(n_x-1)}{n\,(n-1)}$$

où n_x représente le nombre de fois qu'apparaît la lettre x dans une chaîne de longueur n.
L'indice de coïncidence est invariant par toute substitution monoalphabétique. En particulier, si on utilise un simple décalage (chiffre de César), les valeurs des n_x sont échangés entre elles, mais tous les termes se retrouvent dans le calcul, avant ou après substitution, et la somme étant commutative, le résultat est inchangé.

- ◇ Indice de coïncidence en français (comparaison de deux textes français) : $IC_F \simeq 0,077$.
- ◇ Indice de coïncidence sur une séquence aléatoire $IC \simeq \frac{1}{26} = 0,038$.

Utiliser le chiffrement de Vigénère avec un clé un peu longue consiste justement à "randomiser" le texte chiffré, puisqu'on perturbe les fréquences d'apparition des lettres.
Par contre, si la longueur de la clé est connue, chaque sous-texte construit en prélevant une lettre sur L retrouvera un indice de coïncidence maximum. On va alors utiliser l'algorithme suivant pour trouver L.

- ◇ on calcule l'indice de coïncidence du texte I_1.
- ◇ on calcule la moyenne des 2 indices correspondant aux 2 sous textes obtenus en ne prenant qu'une lettre sur 2 et en commençant sur la première puis la deuxième lettre I_2.
- ◇ on calcule ensuite la moyenne des 3 indices correspondant aux 3 sous textes obtenus en ne prenant qu'une lettre sur 3 et en commençant sur la première, puis deuxième, puis troisième lettre I_3.
- ◇ et ainsi de suite jusqu'à I_{20}.

Il suffit donc de rechercher parmi les indices de coïncidence calculés (en se limitant par exemple aux clés de longueur inférieure ou égale à 20), le premier qui est supérieur à un certain seuil (on peut fixer arbitrairement ce seuil à 0.068). Une fois cet indice connu, on en déduit la longueur de la clé.

Rq

Pour une clé de longueur L, I_L s'approche de l'indice caractéristique du français.
Si la clé est de longueur 4, alors les indice I_4, I_8, I_{12} seront tous proches de IC_F.

- **Détermination de la clé :**

Une fois la longueur de la clé connue, on peut identifier la clé en appliquant à chaque sous-texte (dont les lettres ont toutes été décalées d'un même indice) la fonction `craquer_cesar`.

17. Ecrire et documenter une fonction `apparition` qui renvoie dans une liste le nombre d'apparitions dans un texte de chaque lettre de l'alphabet (on pourra modifier la fonction `frequence`.

18. Ecrire et documenter une fonction `indice_friedman` qui calcule l'indice de coïncidence d'un texte.

19. Ecrire et documenter une fonction `craquer_longueur` qui renvoie la longueur de la clé.

20. Ecrire et documenter une fonction `craquer_vigenere` qui prend un texte crypté en argument et qui renvoie la clé.

21. Ecrire un script permettant de décoder un texte crypté sans connaissance de la clé.