CRYPTAGE DES DONNEES

Cryptographie

La cryptographie [1], ou *science du secret*, est un art très ancien qui consiste à rendre inintelligible un message à ceux qui ne sont pas habilités à en prendre connaissance.
On appelle *code* le procédé algorithmique qui permet de crypter un message.

Exercice n° 1 : **Codage monoalphabétique : le chiffrement de César**

César, pour ses communications importantes à son armée, cryptait ses messages. Le chiffrement de César consiste à crypter un texte en décalant les lettres de 3 positions dans l'alphabet [2].
Ainsi, pour crypter un message, A devient D, B devient E, C devient F,... W devient Z, X devient A, Y devient B et Z devient C.
Naturellement, le chiffrage de César peut être utilisé avec un décalage autre que 3. Avec un décalage k, pour k entier naturel entre 0 et 25, k est alors appelé **la clé de chiffrement**. La clé permettant de déchiffrer le message est bien sûr $-k$ (ou $26-k$ si l'on veut une clé positive).

Comme il est plus facile de manipuler des nombres que des lettres, nous associerons à chacune des 26 lettres de A à Z un entier de 0 à 25, *via* la bijection :

$$f_{lc} : \{A, B, C, ..., Z\} \longrightarrow \{0, 1, 2, ..., 25\}$$

Notation arithmétique : en maths, on définit l'ensemble $\mathbb{Z}/26\mathbb{Z} = \{0, 1, 2, ..., 25\}$ des *classes d'équivalences modulo 26*. Cet ensemble est muni d'une loi $+$ de sorte à ce que la somme de deux éléments de k et k' de $\mathbb{Z}/26\mathbb{Z}$ soit égale à l'unique représentant de $k + k'$ dans $\{0, 1, 2, ..., 25\}$, modulo 26.

Le chiffrement de César de clé k correspond alors à la *fonction de chiffrement* C_k : $\begin{cases} \mathbb{Z}/26\mathbb{Z} & \to \mathbb{Z}/26\mathbb{Z} \\ x & \longmapsto x + k \end{cases}$.

1. *Fonctions C_k et C_k^{-1}*
   - Coder une fonction `cesar_chiffre_nb` pour la fonction C_k, admettant pour paramètres un entier à coder, ainsi que la clé de chiffrement k, et retournant l'entier codé.
   - Coder une fonction `cesar_dechiffre_nb` pour la fonction réciproque C_k^{-1}, admettant pour paramètres un entier à décoder, ainsi sue la clé de chiffrement, et retourne l'entier décodé.

2. *Fonctions f_{lc} et f_{lc}^{-1}*
   Pour transformer une lettre en un nombre, on utilise le code *ascii* qui à chaque caractère associe un nombre : comme `ord('A')` vaut 65, `ord('B')` vaut 66...
   La transformation réciproque de `ord` est la fonction `chr()`.

   Coder alors la fonction `flc`, ainsi que sa fonction réciproque f_{lc}^{-1}, identifiée par `fcl`.

1. La cryptanalyse est l'art, pour une personne non habilitée, de décrypter, de décoder et de déchiffrer un message.
La cryptologie est l'ensemble formé de la cryptographie et de la cryptanalyse. La cryptologie fait partie d'un ensemble de théories et de techniques liées à la transmission de l'information (théorie des ondes électro-magnétiques, théorie du signal, théorie des codes correcteur d'erreurs, théorie de l'information, théorie de la complexité,...).

2. On trouvera un témoignage de Suétone sur cette pratique dans la *Vie des 12 Césars*. Le chiffrement de César est un cas particulier de chiffrement mono-alphabétique, c'est-à-dire un chiffrement lettre à lettre.

3. *Transformation d'une liste de caractères en mot*

   Coder une fonction `limo` qui, à une liste de lettres (par ex. ['P','T']), associe le mot constitué de ces lettres (ici, 'PT').

4. *Code de l'algorithme de chiffrement, utilisant toutes les fonctions précédentes*

   • Que représente la fonction $f_{lc}^{-1} \circ C_k \circ f_{lc}$?

   • Proposer un code de la fonction `cesar_chiffre()` qui admet en paramètres une phrase et une clé de chiffrement, et qui renvoie cette phrase chiffrée (en pensant à remplacer un espace par un espace).

   IMPORTANT : en vue de l'écrit, donnez des identifiants explicites à vos variables.

5. *Tests pour $k = 3$*

   a) Coder la phrase : "ALEA JACTA EST".

   b) Décoder la mission : "GRQQH WRXW FH TXH WX DV".

6. On prend maintenant une clé k quelconque. Toutefois, le chiffrement de César est très facilement *attaquable*.

   a) Proposer une fonction *attaque_cesar*() qui admet un mot codé et affiche les 26 traductions possibles obtenues par les 26 clés.

   b) *Test* : retrouver le message et la clé de : `LTBLMNEBKXLTGLXLITVX` .

Exercice n° 2 : Comment casser automatiquement un texte en français crypté par César ?

On voit que la méthode précédente n'est pas complètement automatique, car nous devons repérer le texte intelligible parmi les 26 testes proposés. Or, pour pouvoir utiliser ce système dans un cadre un peu plus complexe, il faudrait que la machine trouve le bon décalage automatiquement.
Une façon d'y parvenir est de faire un pari statistique sur les fréquences d'apparition des lettres (méthode de *l'analyse fréquentielle*). Sachant que le texte est en français, nous pouvons supposer que le texte possède le nombre de A moyen d'un texte français, le nombre de B moyen, etc[3]... Cela sera d'autant plus vrai que le texte à déchiffrer sera long.

1. Coder une fonction `freq` dont les paramètres sont une chaîne de caractères et une lettre, et qui retourne la fréquence d'apparition de la lettre correspondante dans la chaîne.

2. *Travail sur les fichiers en mode lecture/écriture (révisions de Sup)*

   Vous trouverez, dans le dossier du TP 7 un fichier texte `phrasecryptee.txt`.

   Il convient de :

   – le déplacer dans le dossier adéquat où Pyzo "pointe" pour ouvrir les fichiers textes : pour cela, créer un fichier texte (vide) en mode écriture par : `open("test.txt",'w')`, puis rechercher dans quel dossier il a été enregistré. Vous copierez alors `phrasecryptee.txt` dans ce dossier.

   – l'ouvrir en mode lecture par : `f =open("phrasecryptee.txt",'r')`

   – afficher l'objet `c=f.readlines()` et identifier un problème.

   – expliquer les commandes ci-derrière et les utiliser pour résoudre le problème identifié :

3. En français, la probabilité d'apparition du "E" est d'environ 16% , celle du "A" d'environ 9,5%.

```
chaine=""
for L in f :
    chaine=chaine+L.replace("\n","")
```

3. **a)** Déterminer la liste des 27 fréquences d'apparition de chaque lettre de "A" à "Z" dans `chaine`.
On rappelle que la fonction `flc()` peut être utile pour appeler facilement une lettre.

b) Représenter alors ces 27 fréquences sur un graphique.
Comment déduire, en français, la clé de décryptage de la phrase. Décrypter enfin la phrase mystère.

Exercice n° 3 : Codage polyalphabétique : le chiffrement de Vigenère

Au 16e siècle, Blaise de Vigénère publie une méthode de chiffrement, basée sur le carré de Trithème. Il l'aurait repris à son compte alors qu'elle aurait été initialement publiée par Giovan Battista Bellaso.
Pour utiliser la méthode de Vigénère, nous devons au préalable choisir un mot-clé. Soit k son nombre de lettres. On considère alors son image par la fonction f_{lc}, qui est de la forme $(n_1,n_2, \dots , n_k) \in [\![0, 26]\!]^k$.
Ce k-uplet est la "clé numérique".
On regroupe les lettres du texte à crypter par blocs de longueur k, les espaces étant supprimés.
Le chiffrement consiste à effectuer un chiffrement de César dont le décalage dépend du rang de la lettre dans le bloc :

- un décalage de n_1 pour la première lettre de chaque bloc
- ...
- un décalage de n_k pour la k-ième et dernière lettre de chaque bloc.

1. On choisit la clé : "PT". Cryptez sur papier la phrase : "AU TOP".

2. Que pensez-vous de l'attaque statistique ?

3. L'élément de base n'est plus une lettre mais un bloc, c'est-à-dire un regroupement de lettres. La fonction de chiffrement associe, à un bloc de longueur k, un autre bloc de longueur k, ce qui donne :

$$C_{n_1,n_1,\cdots,n_k} : \begin{cases} (\mathbb{Z}/26\mathbb{Z})^k & \longrightarrow (\mathbb{Z}/26\mathbb{Z})^k \\ (x_1, x_2, \cdots, x_k) & \longmapsto (x_1 + n_1, x_2 + n_2, \cdots, x_k + n_k) \end{cases}$$

Proposez un algorithme/code `vigenere_chiffre()` qui admet une phrase et une clé de chiffrement sous la forme d'un tuple et renvoie cette phrase chiffrée.

4. Modifier l' algorithme/code précédent afin de programmer le décodage.

5. Tester les algorithmes en codant/décodant la phrase 'AU TOP' avec la clé 'PT'.

Exercice n° 4 : **Comment casser un mono-alphabétique quelconque ?**

Au lieu de faire correspondre circulairement les lettres, on associe maintenant à chaque lettre une autre lettre (sans ordre fixe ou règle générale).
Mathématiquement, le choix d'une clé revient au choix d'une bijection de l'ensemble A,B,...,Z vers le même ensemble A,B,...,Z.
Il y a 26! permutations possibles (26 choix pour la lettre A, 25 choix pour B... et enfin un seul pour Z).
Ce qui fait environ 4.10^{26} clés. Il y a plus de clés différentes que de grains de sable sur Terre ! Si un ordinateur pouvait tester 1 000 000 de clés par seconde, il lui faudrait alors 12 millions d'années pour tout énumérer.
Cependant, la principale faiblesse du chiffrement mono-alphabétique est qu'une même lettre est toujours chiffrée de la même façon. Ainsi dans les textes longs, les lettres n'apparaissent pas avec la même fréquence, et, suivant la langue utilisée, on peut alors décoder les lettres dont la probabilité d'apparition est proche de 1 (ou de 0).
En français par exemple, les lettres les plus rencontrées sont, par ordre décroissant de probabilité d'apparition :

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

1. Coder un algorithme/code `statistiques`, de paramètre une phrase, et retournant dans une liste le nombre d'occurrence de chaque lettre de la phrase.

   On utilisera la structure suivante : la liste aura pour longueur 26, et chaque élément de la liste aura pour structure : `[occurrence("lettre"),"lettre"]`*, où les lettres seront rangées par ordre alphabétique. On utilisera à volonté la bijection f_{nc} pour automatiser la construction de la liste.*

2. Améliorez le résultat produit par l'algorithme/code précédent en sortant une liste de tuples (lettre,fréquence) avec uniquement les lettres codées.

3. Amusez à essayer de décoder de manière statistique cette phrase (un peu courte) :
   LHLZ HFQ BC HFFPZ WH YOUPFH MUPZH