Informatique Tronc Commun
Devoir Maison 1

1 Chiffrement de César

Le chiffrement de César est une méthode de cryptographie élémentaire : pour coder un message, on décale chaque lettre d'un même nombre de rangs dans l'alphabet, nombre qu'on appelle la **clé**. Le message codé ainsi obtenu n'est alors plus compréhensible par quelqu'un ne disposant pas de la clé, et peut être décodé par quelqu'un en disposant. Afin de pouvoir décaler toutes les lettres, on considère que a est le successeur de z. Par exemple, en codant le message "exemple" avec la clé 4, on obtient "ibiqtpi". Dans la suite, on supposera que les chaînes de caractères considérées ne contiennent que les lettres minuscules de a à z, sans accents, ou des espaces.

1. Les fonctions de l'exercice sont à écrire dans le fichier `DM1.py`, qui définit une liste

   ```
   alphabet = ['a','b','c','d', ... , 'y','z']
   ```

   en tant que variable globale, ce qui signifie qu'on peut l'utiliser dans n'importe quelle fonction sans la prendre en argument. Cette liste permet d'associer chaque lettre à son indice, qui est un entier de 0 à 25.

   (a) Écrire une fonction `int_vers_char` prenant en argument un entier `i` supposé entre 0 et 25, et renvoyant la lettre associée.

   Ainsi, `int_vers_char(3)` doit renvoyer `'d'`.

   (b) Écrire une fonction `char_vers_int` prenant en argument un caractère `c`, et renvoyant l'entier associé. La fonction renverra `None` si `c` est une espace.

   Ainsi, `char_vers_int('y')` doit renvoyer 24.

2. Écrire une fonction `chiffrer_char(c, cle)` prenant en argument un caractère `c` et un entier `cle`, et renvoyant la lettre obtenue en décalant `c` d'un nombre de rangs égal à `cle`, ou renvoyant une espace si `c` est une espace.
   Ainsi, `chiffrer_char('y', 5)` doit renvoyer `'d'`,
   `chiffrer_char('e', -3)` doit renvoyer `'b'`,
   `chiffrer_char(' ', 4)` doit renvoyer `' '`.

3. Écrire une fonction `chiffrer_texte` prenant en argument une chaîne de caractères `m` et un entier `cle`, et renvoyant le message obtenu par chiffrement de César en partant de `m` et en utilisant la clé `cle`.
   Ainsi, `chiffrer_texte("top secret", 10)` doit renvoyer `'dyz combod'`.
   On utilisera l'opérateur `+` pour joindre deux chaînes, ou une chaîne et un caractère (par exemple, `'ab' + 'c'` vaut `'abc'`).

4. Si on connaît la clé, il suffit d'utiliser la fonction précédente avec la valeur opposée pour décoder. Pour décoder sans la clé un message suffisamment long, on peut essayer une attaque statistique. Si on suppose que le message est initialement écrit en français, la lettre la plus fréquente a de fortes chances de correspondre à "e" après décodage, ce qui permet de déterminer la clé.

   (a) Écrire une fonction `indice_max` prenant en argument une chaîne `m` et renvoyant l'entier associé à une lettre de nombre d'occurrences maximal dans `m`.

   Ainsi, `indice_max("huuhxu")` doit renvoyer 20 ;
   `indice_max("dyz combod")` peut renvoyer 3 ou 14.

   (b) En déduire une fonction `decrypter_texte` prenant en argument une chaîne de caractères correspondant à un message chiffré, et tentant de le décoder par l'attaque statistique présentée.

   (c) Décrypter le message `mystere_cesar`. Quelle était la clé ?

5. (a) Que renvoie `decrypter_texte("clup cpkp cpjp")` ? Pourquoi l'attaque statistique échoue-t-elle ici ?

   (b) Une autre attaque, dite **par force brute**, consiste à essayer toutes les clés possibles. Écrire une fonction `force_brute` prenant en argument un message chiffré `m` et affichant, pour chacune de ces clés, la clé et le message obtenu en déchiffrant `m` avec cette clé. Retrouver ainsi le message clair correspondant à `"clup cpkp cpjp"`.

2 Chiffrement par substitution

Le chiffrement de César n'a que 26 clés : la force brute le casse immédiatement. Dans un **chiffrement par substitution**, chaque lettre est remplacée par une autre selon une correspondance quelconque, pas forcément un simple décalage. La clé est une chaîne `cle` de 26 lettres toutes distinctes : la lettre `alphabet[i]` est remplacée par `cle[i]`, et les espaces sont conservées.
Dans les exemples, on utilise la clé suivante, qui reprend l'ordre des touches d'un clavier AZERTY :

```
az = "azertyuiopqsdfghjklmwxcvbn"
```

Elle correspond au remplacement suivant :

lettre claire	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
lettre chiffrée	a	z	e	r	t	y	u	i	o	p	q	s	d	f	g	h	j	k	l	m	w	x	c	v	b	n

Le `e` est remplacé par `t`, le `x` par `v`, etc. : le message `"exemple"` est chiffré en `"tvtdhst"`.

1. Écrire une fonction `chiffre_substitution` prenant en arguments une chaîne `m` et une clé `cle`, et renvoyant le message obtenu en chiffrant `m` par substitution avec la clé `cle`.
   Ainsi, `chiffre_substitution("exemple", az)` doit renvoyer `"tvtdhst"`.
2. Le chiffrement de César est un cas particulier de chiffrement par substitution. Écrire une fonction `cle_cesar` prenant en argument un entier `k` et renvoyant la clé de substitution correspondant au chiffrement de César de clé `k`.
   Ainsi, `cle_cesar(3)` doit renvoyer `"defghijklmnopqrstuvwxyzabc"`.
3. Écrire une fonction `est_cle_valide` prenant en argument une chaîne `cle` et renvoyant `True` si c'est une clé valide (26 lettres toutes distinctes), `False` sinon.
   Ainsi, `est_cle_valide(az)` doit renvoyer `True`, et `est_cle_valide("abc")` doit renvoyer `False`.
4. Écrire une fonction `dechiffre_substitution` prenant en arguments une chaîne `m` chiffrée par substitution et la clé `cle` utilisée, et renvoyant le message clair.
   Ainsi, `dechiffre_substitution("sbett zwyygf", az)` doit renvoyer `"lycee buffon"`.
5. (a) Combien existe-t-il de clés de substitution différentes ? On pourra faire calculer ce nombre par Python.
   (b) Un ordinateur capable de tester un milliard de clés par seconde pourrait-il les essayer toutes ?
6. Un chiffrement par substitution ne modifie pas les fréquences des lettres : si la lettre `e` représente environ 17 % des lettres d'un texte français, la lettre qui la remplace représente environ 17 % des lettres du message chiffré. La chaîne suivante donne les 26 lettres de la plus fréquente à la moins fréquente dans les textes français :

   ```
   ORDRE_FR = "easnitruoldcpmvqfgbhjxyzkw"
   ```

   (a) Écrire une fonction `occurrences` prenant en argument une chaîne `m` et renvoyant la liste `occ` de longueur 26 telle que `occ[i]` soit le nombre d'occurrences de la lettre `alphabet[i]` dans `m` (les espaces ne sont pas comptées).
   Ainsi, `occurrences("abba c")` doit renvoyer la liste `[2, 2, 1, 0, 0, ..., 0]` (de longueur 26).
   (b) Écrire une fonction `ordre_frequences` prenant en argument une chaîne `m` et renvoyant la chaîne formée des 26 lettres de l'alphabet, classées de la plus fréquente à la moins fréquente dans `m` ; en cas d'égalité, les lettres sont rangées dans l'ordre alphabétique. On pourra chercher la lettre la plus fréquente, l'ajouter au résultat, remplacer son nombre d'occurrences par -1 pour qu'elle ne soit plus choisie, et recommencer.
   Ainsi, `ordre_frequences("abracadabra")` doit renvoyer `"abrcdefghijklmnopqstuvwxyz"`.
   (c) Écrire une fonction `cle_probable` prenant en argument un message chiffré `m` et renvoyant la clé obtenue en supposant que, pour tout `k`, la lettre de rang `k` dans `ordre_frequences(m)` chiffre la lettre de rang `k` dans `ORDRE_FR`. Autrement dit, si `ORDRE_FR[k]` est la lettre `alphabet[i]`, la clé renvoyée vérifie `cle[i] == ordre_frequences(m)[k]`.
   (d) Déchiffrer `mystere_substitution` avec la clé `cle_probable(mystere_substitution)`. Le texte obtenu est-il lisible ? Comment l'expliquer ?
   (e) Écrire une fonction `echange` prenant en arguments une clé `cle` et deux lettres `a` et `b`, et renvoyant la clé obtenue à partir de `cle` en échangeant les lettres situées aux indices `char_vers_int(a)` et `char_vers_int(b)`. Expliquer pourquoi cela revient à échanger les lettres `a` et `b` dans le texte déchiffré.
   Ainsi, `echange(az, 'a', 'b')` doit renvoyer `"zaertyuiopqsdfghjklmwxcvbn"`.
   (f) En partant de la clé probable et en utilisant dans la console des appels de la forme

   ```
   cle = echange(cle, 'a' , 'b'),
   ```

   terminer le décryptage de `mystere_substitution`. On pourra commencer par les mots courts : les plus fréquents en français sont `de`, `la`, `le`, `les`, `des`, `en`, `et`, `un`, `une`, `que`, `qui`…