1 TD 3 : Cryptage de Vigenère minuté

(correction page ??)

Abordé lors de cette séance	
programmation	fonction, listes, fichiers, dictionnaire
algorithme	histogramme

César en son temps utilisait déjà le cryptage pour transmettre ses messages [1]. Il remplaçait chaque lettre par sa suivante. A devenait B, B devenait C, et ainsi de suite. Même sans connaître le décalage, décrypter un tel code est simple puisqu'il suffit de compter l'occurrence de chaque lettre du message. On peut raisonnablement affirmer que la lettre la plus fréquente est la lettre "e" ou, si l'auteur est Georges Pérec, une voyelle. On supposant que le décalage entre "e" et la lettre la plus fréquence dans le message crypté, il devient facile de deviner la correspondance entre lettres.

Certains cryptographes plus subtiles eurent tôt fait d'agrandir l'alphabet de chiffrement et de faire en sorte qu'une lettre aurait d'autant plus de représentations cryptées qu'elle serait fréquente. Toutefois, les cryptanalystes ripostèrent par l'analyse des fréquences non plus des lettres mais des couples de lettres.

Pour résister à ces méthodes de décryptage, Blaise de Vigenère inventa aux alentours de 1586 (date de la publication de son livre *Le traité des chiffres* [2]) un code qui ne se trahissait plus par l'étude des lettres ou des couples de lettres les plus fréquents. Cette méthode de chiffrement résista jusqu'en 1854, année où Babbage [3] élabora une méthode qui permettait de casser ce code.

Première demi-heure : listes, boucles, fonctions .

Mais tout d'abord voyons en quoi consiste le code de Vigenère et cela commence par la description du **carré de Vigenère** : un alphabet recopié et décalé d'un cran vers la gauche à chaque ligne.

```
ABCDEFGHIJKLMNOPQRSTUVWXZ
BCDEFGHIJKLMNOPQRSTUVWXZA
CDEFGHIJKLMNOPQRSTUVWXZAB
DEFGHIJKLMNOPQRSTUVWXZABC
EFGHIJKLMNOPQRSTUVWXZABCD
FGHIJKLMNOPQRSTUVWXZABCDE
GHIJKLMNOPQRSTUVWXZABCDEF
HIJKLMNOPQRSTUVWXZABCDEFG
IJKLMNOPQRSTUVWXZABCDEFGH
JKLMNOPQRSTUVWXZABCDEFGHI
KLMNOPQRSTUVWXZABCDEFGHIJ
LMNOPQRSTUVWXZABCDEFGHIJK
MNOPQRSTUVWXZABCDEFGHIJKL
NOPQRSTUVWXZABCDEFGHIJKLM
OPQRSTUVWXZABCDEFGHIJKLMN
PQRSTUVWXZABCDEFGHIJKLMNO
QRSTUVWXZABCDEFGHIJKLMNOP
```

1. http://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage
2. http://fr.wikipedia.org/wiki/Chiffre_de_Vigen%C3%A8re
3. http://fr.wikipedia.org/wiki/Charles_Babbage

```
RSTUVWXZABCDEFGHIJKLMNOPQ
STUVWXZABCDEFGHIJKLMNOPQR
TUVWXZABCDEFGHIJKLMNOPQRS
UVWXZABCDEFGHIJKLMNOPQRST
VWXZABCDEFGHIJKLMNOPQRSTU
WXZABCDEFGHIJKLMNOPQRSTUV
XZABCDEFGHIJKLMNOPQRSTUVW
ZABCDEFGHIJKLMNOPQRSTUVWX
ABCDEFGHIJKLMNOPQRSTUVWXZ
```

1) La création de ce carré n'est pas nécessaire, il est possible passer directement à la question suivante sans créer ce carré sous forme d'une liste de 26 chaînes de caractères. On pourrait bien sûr copier/coller tel quel ce carré dans le programme ou le construire. Pour cela, on s'aidera de l'exemple suivant :

```
s = ""
for i in xrange (0,26) :
    s += chr (65+i)
print s
```

Il suffit de vous en inspirer pour créer une liste de 26 chaînes de caractères, toutes décalées les unes par rapport aux autres.

2) Pour chiffrer un message, il faut une clé qui est un mot (existant ou non) en majuscules, par exemple CODE. Il ne manque plus qu'un message à coder : *Les codes secrets ont joué un rôle discret mais important dans l'Histoire.* Tout d'abord, on va juxtaposer le messsage à coder et la clé maintes fois répétées :

A chaque lettre du message correspond ainsi une lettre de la clé et cette lettre va déterminer, à l'aide du carré de Vigenère, son équivalent crypté. En effet, la première lettre du message L est associée à la lettre C de la clé, cette première lettre est remplacée par celle se trouvant à l'intersection de la ligne commençant par C et de la colonne commençant par L, cela donne N. On continue comme cela jusqu'à la fin du message.

```
lescodessecretsontjoueunrolediscretmaisimportantdanslhistoire
CODECODECODECODECODECODECODECODECODECODECODECODECODECODECODEC
NSV...
```

L'objectif de cette question est d'écrire une fonction qui prend une lettre du message, une lettre de la clé et qui retourne la lettre codée. Il est possible que la fonction `ord` vous soit utile par la suite, c'est la fonction réciproque de la fonction `chr`.

```
def lettre_codee (lettre_message, lettre_cle, carre_vigenere) :
    ...
    return lettre_codee
```

Pour ceux qui ont choisi de passer la première question, il faudra utiliser les deux fonctions suivantes :

```
print chr(65)  # retourne la lettre correspondant au code 65 --> A
print ord("A") # retourne le code associé à la lettre "A" --> 65
```

Remarque 0.1 : Lettre minuscules et majuscules
Le language *Python* fait la différence entre les lettres minuscules et majuscules, elles ont des codes différents.

Seconde demi-heure : fonctions .

3) Il suffit de parcourir toutes les lettres du texte à coder pour terminer le cryptage en utilisant la fonction précédente. La fonction `len` permet de retourner la longueur d'à peu près n'importe quoi, d'une liste, et aussi d'une chaîne de caractères.

```
def cryptage (message, cle, carre_vigenere) :
    ...
    return message_code
```

4) Ecrire la fonction réciproque de la précédente pour décrypter le texte.

Troisième demi-heure : fichiers .

On souhaite appliquer le cryptage sur un texte plus grand en téléchargeant un texte depuis le site Gutenberg [4] comme celui-ci de Gide [5] ou un traité de géométrie en latex [6] ou pour les plus courageux une fable de la fontaine [7].

Deux options sont possibles, la première consiste d'abord à télécharger le fichier puis à l'enregistrer dans un fichier à côté du programme *Python* et enfin à le lire depuis le programme *Python* avec les quelques lignes suivantes :

```
f = open ("nom_de_fichier", "r")
text = f.read ()
f.close ()
```

Plus d'explication au chapitre 7 du support de cours [8]. L'autre solution est de télécharger directement le texte depuis le site :

```
import urllib2
f = urllib2.urlopen ("http://www.gutenberg.org/files/30696/30696-h/30696-h.htm")
text = f.read ()
```

Si ce code ne fonctionne pas, il faudra utiliser le suivant :

```
import urllib2
req = urllib2.Request("http://www.gutenberg.org/files/30696/30696-h/30696-h.htm", headers={'User-Agent' : "Magic B
```

4. http://www.gutenberg.org/
5. http://www.gutenberg.org/files/30696/30696-h/30696-h.htm
6. http://www.gutenberg.org/files/26400/26400-t/26400-t.tex
7. http://www.gutenberg.org/files/20971/mp3/20971-02.mp3
8. http://www.xavierdupre.fr/mywiki/InitiationPython

```
f = urllib2.urlopen (req)
text = f.read ()
```

5) Choisir l'une des deux solutions (chargement depuis un fichier ou depuis un url), puis appliquer le cryptage et décryptage. Est-ce le programme fonctionne et est-ce qu'il retourne le texte de départ ?

6) On souhaite supprimer tous les caractères indésirables. Que proposez-vous ?

Quatrième demi-heure : décryptage et dictionnaire

L'objectif de cette dernière partie est de deviner la longueur de la clé à partir d'un message crypté. On suppose seulement qu'on connaît la langue du texte chiffré.

7) Quel est la lettre la plus fréquente en français ? Expliquez pourquoi cette information est utile pour casser le code de César mais pas le code de Vigenère ?

8) On utilise un dictionnaire pour compter la fréquence de chaque lettre du texte chiffré. Dans ce cas, un dictionnaire nous permet d'associer un nombre à une lettre. Chaque nombre est *indicé* par une lettre.

```
h = { }
h ["A"] = 50
h ["B"] = 50
print h ["A"]
print h
```

On cherche donc à construire un histogramme.

```
def histogramme_lettre (texte) :
    h = { }
    ...
    return h
```

On vérifie que cette histogramme ne permet pas de deviner la longueur de la clé.

9) Babbage s'est dit qu'un groupe de trois lettres consécutives avaient toutes les chances, à chaque fois qu'il apparaissait dans le message chiffré, d'être la conséquence du chiffrement des mêmes lettres du message avec les mêmes lettres de la clé. Pour un groupe de quatre lettres, c'est encore plus probable. Par conséquent, l'espacement entre deux mêmes groupes de lettres chiffrées est un multiple de la longueur de la clé. Par exemple, si la répétition d'un groupe est espacée de 30 lettres, puis celle d'un autre de 25, le plus grand diviseur commun de 25 et 30 est 5. La clé possède donc dans ce cas 5 lettres.

Modifier la fonction précédente pour compter la fréquence de tous les triplets de lettres consécutifs dans le texte. Pourquoi un dictionnaire est-il plus indiqué dans ce cas plutôt qu'une liste ?

10) Le triplet le plus fréquent ne donne pas directement d'indication sur la longueur de la clé de chiffrement. Toutefois, si l'explication ci-dessus vous paraît claire, une fois qu'on a déterminé le triplet de lettres le plus fréquent, comment en déduit-on une longueur possible pour la clé ? Il ne s'agit pas d'implémenter l'algorithme mais plus de le décrire.

Pour aller plus loin ou pour ceux qui ont fini plus tôt .

11) Implémenter en une fonction le raisonnement qui vous permet de proposer une longueur possible pour la clé ?

12) Une fois la longueur connue, comment déterminer la clé ?