TD 9 - Fichier crypté - Cryptage de César, cryptage affine

Informatique
MPSI/PCSI - Lycée Thiers

Exercice 1 : chargement du contenu d'un fichier

Exercice 1. Copier le fichier `TexteCrypte.txt` sur votre ordinateur. Le sauvegarder dans le répertoire utilisateur (celui portant votre nom de login). Il s'agit d'un fichier texte contenant un texte célèbre qui a été crypté à l'aide d'un chiffrement de César (dont on ne connait pas la clé). On souhaite le décrypter, le lire, et sauvegarder le texte déchiffré dans un fichier texte.

1. Ouvrir le fichier `TexteCrypte.txt` en mode d'accès lecture (`'r'`).
2. Récupérer dans une variable `texte` son contenu.
3. Refermer le fichier. Afficher le contenu de `texte`.

Exercice 1 : chargement du contenu d'un fichier

Exercice 2. Copier le fichier `TexteCrypte.txt` sur votre ordinateur. Le sauvegarder dans le répertoire utilisateur (celui portant votre nom de login). Il s'agit d'un fichier texte contenant un texte célèbre qui a été crypté à l'aide d'un chiffrement de César (dont on ne connait pas la clé). On souhaite le décrypter, le lire, et sauvegarder le texte déchiffré dans un fichier texte.

1. Ouvrir le fichier `TexteCrypte.txt` en mode d'accès lecture (`'r'`).
2. Récupérer dans une variable `texte` son contenu.
3. Refermer le fichier. Afficher le contenu de `texte`.

```
f = open('TexteCrypte','r')
texte = f.read()
f.close()
```

Exercice 1 : la table ASCII

Les caractères non accentués, chiffres et autres symbôles accessibles au clavier ou non, sont disponibles dans la table ASCII de caractères, numérotés de 0 à 255 (0xFF).
La fonction `chr(n)` retourne le caractère de la table de code ASCII `n`.
La fonction `ord()` retourne le code ASCII d'un caractère dans la table.

Exercice 1 : la table ASCII

Les caractères non accentués, chiffres et autres symbôles accessibles au clavier ou non, sont disponibles dans la table ASCII de caractères, numérotés de 0 à 255 (0xFF).
La fonction `chr(n)` retourne le caractère de la table de code ASCII `n`.
La fonction `ord()` retourne le code ASCII d'un caractère dans la table.

Exercice 4. Exécuter le script suivant qui affiche à l'écran les caractères de la table ASCII de code allant de 33 à 126 (passage à la ligne tous les 16 caractères) :

```
N = 16
for i in range(33,127):
    print(i, ':', chr(i), end = ' ', sep = '')
    if (i-32) % N == 0:
        print('')
```

Explication : dans la fonction `print`, l'option `end = ' '` force l'impression en fin de ligne d'un espace `' '` plutôt que d'un saut de ligne

Le code a pour effet :

```
33:! 34:¨ 35:# 36:$ 37:% 38:& 39:’ 40:( 41:) 42:* 43:+ 44:, 45:- 46:. 47:/ 48:0
49:1 50:2 51:3 52:4 53:5 54:6 55:7 56:8 57:9 58:: 59:; 60:< 61:= 62:> 63:? 64:@
65:A 66:B 67:C 68:D 69:E 70:F 71:G 72:H 73:I 74:J 75:K 76:L 77:M 78:N 79:O 80:P
81:Q 82:R 83:S 84:T 85:U 86:V 87:W 88:X 89:Y 90:Z 91:[ 92:\93:] 94:^ 95:_ 96:‘
97:a 98:b 99:c 100:d 101:e 102:f 103:g 104:h 105:i 106:j 107:k 108:l 109:m 110:n
111:o 112:p 113:q 114:r 115:s 116:t 117:u 118:v 119:w 120:x 121:y 122:z 123:{
124:| 125:} 126:˜
```

Exercice 3 : le chiffrement de César

Le chiffrement de César.

C'est une méthode très simple de chiffrement de messages en un texte crypté pour le rendre "illisible" à qui n'en a pas la clef : dans une message changer chaque lettre par la N-ième lettre suivante dans l'ordre alphabétique (après z on reprend en a), où N est un entier entre 1 et 25, c'est la clé de chiffrement.

1. Ecrire une fonction `Crypt` prenant en paramètre un caractère (chaine d'un seul caractère) et la clé, et retournera le caractère crypté. Par exemple `Crypt('a',3)` renverra le caractère `'d'`, et `Crypt('B',3)` renverra `'E'`.
2. Ecrire une fontion CesarCrypt() qui prend deux paramètres une chaine de caractère (le message à crypter) et la clef de chiffrement N, et renvoie le message crypté (chaine de caractère).
3. Ecrire une fonction CesarDecrypt() qui prend deux paramètres, un texte crypté et la clef, et retourne le texte décrypté.

1)

```
def Crypt(c,n):
    n = n%26
    if 'A' <= c <= 'Z':      # MAJUSCULE
        return chr(ord('A')+(ord(c)-ord('A')+n)%26)
    elif 'a' <= c <= 'z':    # MINUSCULE
        return chr(ord('a')+(ord(c)-ord('a')+n)%26)
    else :
        return c
```

1)

```
def Crypt(c,n):
    n = n%26
    if 'A' <= c <= 'Z':      # MAJUSCULE
        return chr(ord('A')+(ord(c)-ord('A')+n)%26)
    elif 'a' <= c <= 'z':    # MINUSCULE
        return chr(ord('a')+(ord(c)-ord('a')+n)%26)
    else :
        return c
```

2) Fonction de cryptage

```
def CesarCrypt(S,n):
    Scrypt = ''
    for char in S:
        Scrypt += Crypt(char,n)
    return Scrypt
```

1)

```
def Crypt(c,n):
    n = n%26
    if 'A' <= c <= 'Z':      # MAJUSCULE
        return chr(ord('A')+(ord(c)-ord('A')+n)%26)
    elif 'a' <= c <= 'z':    # MINUSCULE
        return chr(ord('a')+(ord(c)-ord('a')+n)%26)
    else :
        return c
```

2) Fonction de cryptage

```
def CesarCrypt(S,n):
    Scrypt = ''
    for char in S:
        Scrypt += Crypt(char,n)
    return Scrypt
```

3) Fonction de décryptage

```
def CesarDeCrypt(S,n):
    return CesarCrypt(S,-n)
```

Exercice 4 : Cryptanalyse

Exercice 5. Cryptanalyse du chiffrement de César. On écrit une fonction `CryptAnalyse` qui prend en paramètre un texte, crypté par chiffrement de César, et qu'on souhaite retourner le texte décrypté. Pour cela, pour un texte suffisament long (en français), on peut quasiment toujours supposer que le caractère le plus fréquent est le `'e'`.

Exercice 4 : Cryptanalyse

Exercice 6. Cryptanalyse du chiffrement de César. On écrit une fonction `CryptAnalyse` qui prend en paramètre un texte, crypté par chiffrement de César, et qu'on souhaite retourner le texte décrypté. Pour cela, pour un texte suffisament long (en français), on peut quasiment toujours supposer que le caractère le plus fréquent est le `'e'`.

1. Ecrire une fonction qui prend en paramètre une chaine de caractère `txt` et qui renvoie une liste de 26 entiers. A l'indice `i`, la liste contiendra le nombre de fois où le i-ème caractère de l'alphabet minuscule apparait dans la chaine ; par exemple à l'indice 0 le nombre de `'a'` dans `txt`.
2. Ecrire une fonction `indMax` prenant en paramètre une liste de nombres et renvoie l'indice où se situe le maximum.
3. Ecrire le code de la fonction `Cryptanalyse` : après avoir récupéré le caractère minuscule le plus fréquent dans la chaine.
4. Déchiffrer le message crypté, et après avoir affiché son contenu, le sauvegarder dans un fichier `texteDecrypte`.

Exercice 4 : Cryptanalyse

1.

```
def compteCaractere(chaine):
    Nbre = [0] * 26
    for x in chaine:
        if 'a' <= x <= 'z':
            indice = ord(x)-ord('a')
            Nbre[indice] += 1
    return Nbre
```

Exercice 4 : Cryptanalyse

2.

```
def indiceMax(L):
    M = L[0]
    m = 0
    for k in range(1,len(L)):
        if L[k] > M:
            M = L[k]
            m = k
    return m
```

Exercice 4 : Cryptanalyse

3.

```
def cryptAnalyse(txt):
    i = indiceMax(compteCaractere(txt))
    TxTcasse = CesarDeCrypt(txt,i+ord('a')-ord('e'))
    return TxTcasse
```

Exercice 4 : Cryptanalyse

3.

```
def cryptAnalyse(txt):
    i = indiceMax(compteCaractere(txt))
    TxTcasse = CesarDeCrypt(txt,i+ord('a')-ord('e'))
    return TxTcasse
```

4.

```
>>> txt = cryptAnalyse(texte)
>>> f = open('texteDecrypte','w')
>>> f.write(txt)
>>> f.close()
>>> print(txt)
```

Exercice 4 : Cryptanalyse

Ce qui fut cause que je pensai qu'il fallait chercher quelque autre methode, [...] j'aurais assez des quatre suivants, pourvu que je prisse une ferme et constante resolution de ne manquer pas une seule fois a les observer.

Le premier etait de ne recevoir jamais aucune chose pour vraie, que je ne la connusse evidemment etre telle : c'est-a-dire, d'eviter soigneusement la precipitation et la prevention; et de ne comprendre rien de plus en mes jugements, que ce qui se presenterait si clairement et si distinctement a mon esprit, que je n'eusse aucune occasion de le mettre en doute.

Le second, de diviser chacune des difficultes que j'examinerais, en autant de parcelles qu'il se pourrait, et qu'il serait requis pour les mieux resoudre.

Le troisieme, de conduire par ordre mes pensees, en commencant par les objets les plus simples et les plus aises a connaitre, pour monter peu a peu, comme par degres, jusques a la connaissance des plus composes; et supposant meme de l'ordre entre ceux qui ne se precedent point naturellement les uns les autres.

Et le dernier, de faire partout des denombrements si entiers, et des revues si generales, que je fusse assure de ne rien omettre.

Rene Descartes, Discours de la Methode.

Exercice 3 : le chiffrement affine

Le chiffrement affine.
Pour le chiffrement affine, chaque lettre $A, B, \ldots, Z$ est remplacée par son rang entre 0 et 25. On choisit deux nombre entiers a et b entre 0 et 25. On note $r(x)$ le reste de la division euclidienne de $y = ax + b$ par 26. La lettre correspondante est la lettre cryptée.

Exemple : avec $a = 11$ et $b = 2$. Pour encrypter D : son rang est 3.
$11 \times 3 + 2 = 35 \equiv 9[26]$. La lettre codée est J.

- **Pour que deux lettres différentes soient cryptées différemment il faut et il suffit que a soit premier avec 26 (*i.e.* $pgcd(a, 26) = 1$).**

1. Le chiffrement de César correspond à quel cas particulier du chiffrement affine ?
2. Ecrire une fonction `affCrypt` qui prend en paramètre une lettre et les clés `a` et `b` et retourne le caractère crypté.
3. Ecrire la fonction `EncryptageAff` qui prend en paramètre le texte à crypter et les clés de cryptage `a` et `b` et retourne le texte crypté.

- **Pour décrypter : appliquer le même principe avec a' et b' tels que :**

$$\begin{cases} aa' \equiv 1[26] \\ b' \text{ est le reste de la division euclidienne de } a'(26 - b) \text{ par } 26 \end{cases}$$

4. Ecrire une fonction `cleInverse` prenant en paramètre `a` et `b` et qui retourne `a'` et `b'`. (On pourra chercher `a'` entre 0 et 25).
5. Ecrire une fonction `DecryptAff` prenant en paramètre un texte crypté (chaine de caractère) et les clé de cryptage `a` et `b`, et qui retourne le texte décrypté.

Exercice 3 : le chiffrement affine

1) Le chiffrement de César correspond au cas où $a = 1$.

Exercice 3 : le chiffrement affine

1) Le chiffrement de César correspond au cas où $a = 1$.

2)

```
def affCrypt(Char,a,b):
    if ord('A') <= ord(Char) <= ord('Z'):
        ecart = ord(Char)-ord('A')
        return chr(ord('A')+(a*ecart+b)%26)
    elif ord('a') <= ord(Char) <= ord('z'):
        ecart = ord(Char)-ord('a')
        return chr(ord('a')+(a*ecart+b)%26)
    else:
        return Char
```

Exercice 3 : le chiffrement affine

1) Le chiffrement de César correspond au cas où $a = 1$.

2)

```
def affCrypt(Char,a,b):
    if ord('A') <= ord(Char) <= ord('Z'):
        ecart = ord(Char)-ord('A')
        return chr(ord('A')+(a*ecart+b)%26)
    elif ord('a') <= ord(Char) <= ord('z'):
        ecart = ord(Char)-ord('a')
        return chr(ord('a')+(a*ecart+b)%26)
    else:
        return Char
```

3) Il faudra au préalable avoir écrit une fonction pour le calcul du pgcd (algorithme d'Euclide), pour tester que le paramètre a soit premier avec 26 :

```
def pgcd(a,b):
```

Exercice 3 : le chiffrement affine

```
def EncryptageAff(S,a,b):
    if pgcd(a,26) != 1:
        return False
    Scrypte = ""
    for char in S:
        Scrypte += affCrypt(char,a,b)
    return Scrypte
```

4)

```
def cleInverse(a,b):
    for A in range(26):
        if a*A%26 == 1:
            break
    return (A,A*(26-b)%26)

def DecryptageAff(S,a,b):
```

Exercice 4 : Chiffrement de Vigenère

Le chiffrement de Vigenère.
Il utilise pour clé privée un mot, par exemple *CLE*. En "répétant" sous le texte la clé, chaque lettre de la clé donne le chiffrement de César à appliquer au caractère au dessus. Par exemple $C : A$ devient $C ; L ; A$ devient $L ; E, A$ devient E. Ainsi :

message à crypter :	M	O	N		M	E	S	S	A	G	E
clé :	C	L	E	C	L	E	C	L	E	C	L
message crypté :	O	Z	R		X	I	U	D	E	I	P

1. Le chiffrement de César correspond à quel cas particulier du chiffrement de Vigénère ?
2. Ecrire une fonction `vigenere` prenant en paramètre le texte à crypter et la clé de cryptage, et qui retourne le texte crypté.
3. Ecrire une fonction `vigenereDecrypt` prenant en paramètre le texte crypté et la clé de cryptage et qui retourne le texte décrypté.

Exercice 4 : Chiffrement de Vigenère

1) Le chiffrement de César correspond au cas d'un chiffrement de Vigenère dont la clé est réduite à un seul caractère.

Exercice 4 : Chiffrement de Vigenère

1) Le chiffrement de César correspond au cas d'un chiffrement de Vigenère dont la clé est réduite à un seul caractère.

2) On emploie la fonction `Crypt` de l'exercice 2 (chiffrement de César) :

```
def Vigenere(msg,cle):
    cle = cle.upper()
    N = len(msg)
    n = len(cle)
    msgCrypte = ''
    for i in range(N):
        decal = (ord(cle[i%n])-ord('A'))%26
        msgCrypte = msgCrypte + Crypt(msg[i],decal)
    return msgCrypte
```

Exercice 4 : Chiffrement de Vigenère

3) IL s'agit encore d'un chiffrement "symétrique" : on applique le même procédé pour le cryptage et le décryptage ; il suffit de prendre la clé "opposée".

```
def VigenereDecrypt(msg,cle):
    cle = cle.upper()
    N = len(msg)
    n = len(cle)
    msgCrypte = ''
    for i in range(N):
        decal = -(ord(cle[i%n])-ord('A'))%26
        msgCrypte = msgCrypte + Crypt(msg[i],decal)
    return msgCrypte
```