Cheminement cryptographique : découverte de quelques algorithmes

Introduction

- Ce DM a pour vocation de vous initier à la *cryptographie*. La *cryptographie* est une des disciplines de la *cryptologie* s'attachant à protéger des messages (assurant confidentialité, authenticité et intégrité) en s'aidant souvent de secrets ou de *clés*. Nous nous intéresserons ici plus précisement au *chiffrement*, c'est-à-dire : la transformation à l'aide d'une clé d'un message en clair en un message incompréhensible (dit texte *chiffré*) pour celui qui ne dispose pas de la *clé de déchiffrement*.
- Par exemple, le chiffrement dit "De César" consiste à substituer une lettre par une autre un peu plus loin dans l'alphabet, c'est-à-dire qu'une lettre est toujours remplacée par la même lettre et que l'on applique le même décalage à toutes les lettres, cela rend très simple le déchiffrement d'un message puisqu'il y a 25 décalages possibles. Ainsi, en appliquant un chiffrement de César décalant les lettres de 3 rangs, le message : "Caius julius surnomme caesar ambitieux leader politique est determiné a devenir dictateur il est accueilli en triomphe dans rome lors de la celebration des lupercales" devient, "mksec tevsec cebxywwo mkockb kwlsdsoeh voknob zyvsdsaeo ocd nodobwsxo k nofoxsb nsmdkdoeb sv ocd kmmeosvvs ox dbsywzro nkxc bywo vybc no vk movolbkdsyx noc vezobmkvoc".
- Au terme de ce devoir, vous serez en mesure de :
  - Comprendre et produire des algorithmes de chiffrement plus ou moins complexes.
  - Chiffrer et déchiffrer un fichier à l'aide d'un algorithme.

[Partie 1] : Avant de commencer...quelques fonctions utiles de base !

Dans cette première partie, il vous est demandé d'écrire quelques fonctions qui vous seront utiles tout au long du sujet.

[1.1] Ecrire une fonction "lire_fichier" qui affichera toutes les lignes d'un fichier. La fonction prendra en paramètre le nom du fichier.

lire_fichier("data.txt") retournera toutes les ligne du fichier "data.txt". **Attention** : la fonction ne tiendra pas compte des espaces dans une ligne. Ouvrez et observez le fichier "coucou_n.txt". Vous testerez votre fonction avec ce fichier. Ainsi **lire_fichier("coucou.txt")** devra exactement retourner l'affichage suivant :

```
coucou1
coucou2
coucou3
coucou4

et non pas :

    coucou1
coucou2
coucou3
coucou4

ou encore :

coucou1

coucou2

coucou3

ou encore :

'coucou1'
'coucou2'
'coucou3'

etc...
```

Entrée [ ]:

```
def lire_fichier(fichier):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
lire_fichier("coucou.txt")
```

[1.2] Ecrire une fonction **"lire_fichier_ligne"** qui lira la n-ième ligne d'un fichier. La fonction prendra en paramètre le nom du fichier et le numéro de ligne.

lire_fichier_ligne("data.txt", 3) : retournera la ligne 3 du fichier data.txt. Vous testerez votre fonction avec le fichier "coucou_n.txt". Ainsi : **lire_fichier_ligne("coucou", 2)** retournera :

```
coucou2
```

Entrée [ ]:

```
def lire_fichier_ligne(fichier,num_ligne):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
lire_fichier_ligne("coucou.txt",2)
```

[1.3] Ecrire une fonction **"ecrire_message_nomfichier"** qui créera un nouveau fichier "nomfichier" (ici, vous nommez votre fichier comme vous l'entendez !) et écrira un message dans ce fichier. La fonction prendra en paramètre le 'message à écrire' et le nom de votre fichier.

Ainsi **ecrire_message_nomfichier('Coucou c'est moi!',test)** écrira 'coucou c'est moi !' à la première ligne du fichier "test.txt". Et **lire_fichier("test.txt")** retournera :

```
Coucou c'est moi !
```

Entrée [ ]:

```
def ecrire_message_nomfichier(message,fichier):
   # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
ecrire_message_nomfichier("coucou c'est moi !","coucou_c_est_moi.txt")
lire_fichier("coucou_c_est_moi.txt")
```

[Partie 2] : Algorithme de César

En *cryptographie*, le *chiffrement* par décalage, aussi appelé *chiffrement* de César est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes. Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Par exemple, un chiffrement de César avec décalage de 1 consiste à remplacer toutes les occurences de 'A' par 'B', toutes les occurences de 'C' par 'D' etc.

Avant de nous lancer dans l'écriture de l'algorithme de chiffrement, introduisons succinctement le principe d'*encodage*. Celui-ci vous servira pour coder l'algorithme de César. Tous les caractères que l'on peut écrire à l'aide d'un ordinateur sont représentés en mémoire par des nombres. On parle d'*encodage*. Le "a" minuscule par exemple est représenté, ou *encodé*, par le nombre 97. Le "A" quant à lui, est codé avec le nombre 65 etc. Les correspondances caractères/représentation en nombre décimal, sont présentées dans le tableau ci-dessous.

Char	Dec	Binary	Char	Dec	Binary	Char	Dec	Binary
!	033	00100001	A	065	01000001	a	097	01100001
"	034	00100010	B	066	01000010	b	098	01100010
#	035	00100011	C	067	01000011	c	099	01100011
$	036	00100100	D	068	01000100	d	100	01100100
%	037	00100101	E	069	01000101	e	101	01100101
&	038	00100110	F	070	01000110	f	102	01100110
'	039	00100111	G	071	01000111	g	103	01100111
(	040	00101000	H	072	01001000	h	104	01101000
)	041	00101001	I	073	01001001	i	105	01101001
*	042	00101010	J	074	01001010	j	106	01101010
+	043	00101011	K	075	01001011	k	107	01101011
,	044	00101100	L	076	01001100	l	108	01101100
-	045	00101101	M	077	01001101	m	109	01101101
.	046	00101110	N	078	01001110	n	110	01101110
/	047	00101111	O	079	01001111	o	111	01101111
0	048	00110000	P	080	01010000	p	112	01110000
1	049	00110001	Q	081	01010001	q	113	01110001
2	050	00110010	R	082	01010010	r	114	01110010
3	051	00110011	S	083	01010011	s	115	01110011
4	052	00110100	T	084	01010100	t	116	01110100
5	053	00110101	U	085	01010101	u	117	01110101
6	054	00110110	V	086	01010110	v	118	01110110
7	055	00110111	W	087	01010111	w	119	01110111

Tester les fonctions ord(), chr() ci-dessous, et dire ce que permettent ces fonctions.

Entrée [ ]:

```
ord('A')
```

Entrée [ ]:

```
ord(' ')
```

Entrée [ ]:

```
ord('b')
```

[2.1] Quelle est la fonction de l'instruction ord() en python ?

```
 VOTRE REPONSE ICI
```

Entrée [ ]:

```
chr(65)
```

Entrée [ ]:

```
chr(98)
```

Entrée [ ]:

```
chr(32)
```

[2.2] Quelle est la fonction de l'instruction chr() en python ?

```
VOTRE REPONSE ICI
```

[2.3] Ecrire la fonction **"minmaj_majmin"** qui convertit un caractère MAJUSCULE en minuscule, et réciproquement. Vous utiliserez les fonctions chr() et ord().

"minmaj_majmin" prend en paramètre un caractère, si le caractère est une majuscule, il le transforme en minuscule, et réciproquement.

Par exemple **minmaj_majmin('A')** retournera :

```
a
```

Pour vous aider : vous noterez que la fonction char.isupper(), qui retourne un booléen (true/false) permet de tester si un caractère est majuscule, ou minuscle.

Entrée [ ]:

```
def minmaj_majmin(caractere):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
minmaj_majmin('b')
```

[2.4] Ecrire la fonction **"minmaj_majmin_inverse"** qui inversera la propriété majuscule ou minuscule de n'importe quelle phrase écrite avec des lettres de l'alphabet. Vous devez utiliser la fonction **minmaj_majmin** précédement définie.

Par exemple **minmaj_majmin_inverse("Ma Phrase")** retournera :

```
'mA pHRASE'
```

Entrée [ ]:

```
def minmaj_majmin_inverse(phrase):
   # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
minmaj_majmin_inverse("Ma Phrase")
```

[2.5] Ecrire la fonction **"minmaj_majmin_phrase"** qui convertira n'importe quelle phrase écrite avec des lettres de l'alphabet en majuscules ou en minuscules.Vous devez utiliser la fonction **minmaj_majmin** précédement définie.

La fonction **minmaj_majmin_phrase** prendra en paramètre une phrase et un indice M ou m. Lorsque l'indice "M" sera indiqué, la phrase sera intégralement convertie en MAJUSCULES, lorsque l'indice "m" sera indiqué, la phrase sera convertie en minuscules.

Par exemple **minmaj_majmin_phrase("ceCi est unE PhrasE Test",'m')** :

```
'ceci est une phrase test'
```

Entrée [ ]:

```
def minmaj_majmin_phrase(phrase,indice):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
minmaj_majmin_phrase("ceCi est unE PhrasE Test",'m')
```

Entrée [ ]:

```
#Tester votre fonction
minmaj_majmin_phrase("ceCi est unE PhrasE Test",'M')
```

[2.6] Ecrire la fonction **'cesar_chiffre'** qui prendra en paramètre un texte non-chiffré et un décalage de taille n et retournera le texte chiffré. Les caractères MAJUSCULES devront être chiffrés en MAJUSCULE, les caractères en minuscule devront être chiffré en minuscule.

Par exemple **cesar_chiffre("aBCd AbcD",1)** retournera :

```
'bCDe BcdE'
```

Entrée [ ]:

```
def cesar_chiffre(text,decalage):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
cesar_chiffre("aBCd AbcD",1)
```

[2.7] Ecrire la fonction **'cesar_dechiffre'** qui prendra en paramètre un texte chiffré et un décalage de taille n et retournera le texte déchiffré. Les caractères MAJUSCULES devront être chiffrés en MAJUSCULES, les caractères en minuscules devront être chiffrés en minuscules.

Par exemple **cesar_chiffre("bCDe BcdE",1)** retournera :

'aBCd AbcD'

Entrée [ ]:

```
def cesar_dechiffre(text,decalage):
    # CODE A COMPLETER
```

[2.8] Ecrire la fonction **'cesar_chiffre_minmaj_inverse'** qui prendra en paramètre un texte non chiffré et un décalage de taille n et retournera le texte chiffré, et où toutes les MAJUSCULES seront transformées en minuscules, et réciproquement. Les caractères MAJUSCULES devront être chiffrés en MAJUSCULES, les caractères en minuscules devront être chiffré en minuscules.

Par exemple **cesar_chiffre("AbcD eFGh",1)** retournera :

'bCDe FghI'

Entrée [ ]:

```
def cesar_chiffre_minmaj_inverse(text,decalage):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
cesar_chiffre_minmaj_inverse("AbcD eFGh",1)
```

[2.9] Ecrire la fonction **'cesar_chiffre_min_ou_maj'** qui prendra en paramètre un texte non chiffré et un décalage de taille n et retournera le texte chiffré, et qui convertira n'importe quelle phrase écrite avec des lettres de l'alphabet en majuscules ou en minuscules. La fonction prendra en paramètre une phrase et un indice M ou m. Lorsque l'indice "M" sera indiqué, la phrase sera convertie en MAJUSCULES, lorsque l'indice "m" sera indiqué, la phrase sera convertie en minuscules.

Par exemple **cesar_chiffre_min_ou_maj("AbcD eFGh",1,"M")** retournera :

'BCDE FGHI'

Entrée [ ]:

```
def cesar_chiffre_min_ou_maj(text,decalage,indice):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
cesar_chiffre_min_ou_maj("AbcD eFGh",1,"M")
```

[2.10] Ecrire la fonction **"chiffre_ligne_fichier_cesar"** qui lira la ligne n d'un fichier, la chiffrera avec l'algorithme de cesar (la fonction **"cesar_chiffre"** devra pour cela être utilisée) et produira un nouveau fichier qui contiendra la ligne chiffrée du premier fichier.

"chiffre_ligne_fichier_cesar" prendra 4 paramètre en entrée : le nom du fichier à chiffrer qui contient la ligne à chiffre, le nom du nouveau fichier à produire, le numéro de la ligne ciblée dans le fichier à chiffer et enfin le décalage souhaité.

Vous testerez votre fonction avec le fichier 'test_cesar.txt'.

Entrée [ ]:

```
def chiffre_ligne_fichier_cesar(fichier_a_chiffrer,fichier_chiffre,num_ligne,decalage):
   # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
chiffre_ligne_fichier_cesar("test_cesar.txt","test_cesar_2.txt",3,2)
```

Entrée [ ]:

```
#Tester votre fonction
lire_fichier("test_cesar_2.txt")
```

[2.11] Ecrire la fonction **"chiffre_fichier_cesar"** qui lira chaque ligne d'un fichier, chiffrera chacune d'elle avec l'algorithme de cesar (la fonction **"cesar_chiffre"** devra pour cela être utilisée) et produire un nouveau fichier qui contiendra les lignes du premier fichier chiffré.

Vous testerez votre fonction avec le fichier 'test_cesar.txt'.

Entrée [ ]:

```
def chiffre_fichier_cesar(fichier_a_chiffrer,fichier_chiffre,decalage):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
chiffre_fichier_cesar("test_cesar.txt","test_cesar_3.txt",1)
```

Entrée [ ]:

```
#Tester votre fonction
lire_fichier("test_cesar_3.txt")
```

[Partie 3] : Chiffrement de Vigenere

Le chiffrement de Vigenère est un système de *chiffrement* par substitution polyalphabétique dans lequel une même lettre du message clair peut, suivant sa position dans celui-ci, être remplacée par des lettres différentes, contrairement à un système de chiffrement mono-alphabétique (comme le chiffre de César). Ceci signifie que la *clé de chiffrement* est une chaîne de caractères, et c'est là-dessus que repose la sécurité de l'algorithme : car une même lettre ne sera alors pas forcément chiffrée de la même façon !

Par exemple : prenons comme message secret le mot "annulationcosmique", et comme *clé* de chiffrement "python". La première étape consiste à rendre notre *clé* aussi longue que notre message à chiffrer, pour cela, on répète simplement autant de fois la *clé* que nécessaire. Ceci donne :

```
a n n u l a t i o n c o s m i q u e
p y t h o n p y t h o n p y t h o n
```

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

On a ainsi :

a+p = position de a + position de p = 0+15 = 15 = p

n+y = position de n + position de y = 13+24= 37=25+12, or à la 12ème position on trouve l, donc n+y=L

Explication pour n+y : comme il n'y a pas de lettre à la position 37, on va jusqu'à la position 25 (Z), puis on revient au début de l'alphabet et on le parcourt de gauche à droite de 12 cases (car 25 + 12 = 37). On tombe ainsi sur M à la douzième position. (Attention, la première position est A ! L'aphabet est indicé à partir de 0 !)

n+t = 13+19= 32 = 25 + 7 = G (preuve que l'on peut coder le n de deux manières différentes avec le chiffrement de Vigenere !)

Et ainsi de suite.

[3.1] Ecrire la fonction **"encodage_unitere_vigenere"** qui retourne la lettre correspondant à la somme de deux autres lettres en fonction de leurs positions dans l'alphabet.

encodage_unitere_vigenere prendra 2 paramètres en entrée : une lettre à chiffrer, et une lettre clé de chiffrement. Elle retournera la lettre produite par la somme des deux.

Attention, la fonction vérifiera préalablement que la clé et la lettre à chiffrer sont toutes les deux en MAJUSCULES ou en minuscule. Si l'une des deux est en MAJUSCULE et l'autre en minuscules, alors elle mettra par défaut les deux en minuscules. Vous vérifierez le bon fonctionnement de votre fonction sur les 4 combinaisons possibles (2 lettres majuscules, 2 lettres minuscules, une lettre minuscule au premier paramètre et une majuscule au second, et inversement).

Par exemple, **encodage_unitere_vigenere('n','Y')** retournera :

```
'l'
```

Entrée [ ]:

```
def encodage_unitere_vigenere(lettre,cle):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
encodage_unitere_vigenere('N','Y')
```

Entrée [ ]:

```
#Tester votre fonction
encodage_unitere_vigenere('n','y')
```

Entrée [ ]:

```
#Tester votre fonction
encodage_unitere_vigenere('N','y')
```

Entrée [ ]:

```
#Tester votre fonction
encodage_unitere_vigenere('n','Y')
```

[3.2] Ecrire la fonction **"vigenere_chiffre"** qui retourne un message chiffré avec l'algorithme de Vigenere.

La fonction **vigenere_chiffre** prendra 2 paramètres en entrée : un message à chiffrer, et une clé de chiffrement. Vous devrez obligatoirement utiliser la fonction **"encodage_unitere_vigenere"** précédemment implémentée. Tester votre fonction sur les exemples proposés.

Entrée [ ]:

```
def vigenere_chiffre(message, cle):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
vigenere_chiffre("annulationCosmique","PythOn")
```

Entrée [ ]:

```
#Tester votre fonction
vigenere_chiffre('annulationcosmique', 'python')
```

Entrée [ ]:

```
#Tester votre fonction
vigenere_chiffre('ANNULATIONCOSMIQUE', 'PYTHON')
```

[3.3] Ecrire la fonction **"chiffre_ligne_fichier_vigenere"** qui lira la ligne n d'un fichier, la chiffrera avec l'algorithme de vigenere (la fonction "vigenere" sera pour cela utilisée) et produira un nouveau fichier qui contiendra la ligne chiffrée du premier fichier.

"chiffre_ligne_fichier_vigenere" prendra 4 paramètres en entrée : le nom du fichier à chiffrer qui contient la ligne à chiffrer, le nom du nouveau fichier à produire, le numéro de la ligne ciblée dans le fichier à chiffer et enfin la clée de chiffrement souhaitée.

Entrée [ ]:

```
def chiffre_ligne_fichier_cesar(fichier_a_chiffrer,fichier_chiffre,num_ligne,cle):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
chiffre_ligne_fichier_cesar("test_vigenere.txt","test_vigenere_1.txt",1,"PythOn")
```

Entrée [ ]:

```
#Tester votre fonction
lire_fichier("test_vigenere_1.txt")
```

[3.4] Ecrire la fonction **"chiffre_fichier_vigenere"** qui lira chaque ligne d'un fichier, chiffrera chacune d'elle avec l'algorithme de vigenere (la fonction **"vigenere_chiffre"** sera pour cela utilisée) et produira un nouveau fichier qui contiendra les lignes du premier fichier chiffrées.

La fonction **"chiffre_fichier_vigenere"** prendra en entrée 3 paramètres : le nom du fichier à chiffrer, le nom du nouveau fichier, et la clé de chiffrement souhaitée.

Entrée [ ]:

```
def chiffre_fichier_vigenere(fichier_a_chiffrer,fichier_chiffre,cle):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
chiffre_fichier_vigenere("test_vigenere.txt","test_vigenere_2.txt","python")
```

Entrée [ ]:

```
#Tester votre fonction
lire_fichier("test_vigenere_2.txt")
```

[Partie 4] : Déchiffrer un fichier pour réveler une image cachée...

Soit la fonction **afficher_fichier_crypte** implémentée ci-dessous. Exécutez cette fonction avec les fichiers "fichier_chiffre_1.txt", "fichier_chiffre_2.txt" et "fichier_chiffre_3.txt". Observez attentivement le résultat de l'exécution ainsi que le contenu du fichier affiché.

Entrée [ ]:

```
def afficher_fichier_crypte(fichier):
    with open(fichier,"r") as fichier:
        line=fichier.read()
        print(line)
    fichier.close()
```

Entrée [ ]:

```
afficher_fichier_crypte("fichier_chiffre_1.txt")
```

Entrée [ ]:

```
afficher_fichier_crypte("fichier_chiffre_2.txt")
```

Entrée [ ]:

```
afficher_fichier_crypte("fichier_chiffre_3.txt")
```

[4.1] Ecrire la fonction **"dechiffre_fichier"** qui déchiffrera le contenu des trois fichiers "fichier_chiffre_1.txt","fichier_chiffre_2.txt" et "fichier_chiffre_3.txt" et écrire le contenu déchiffré dans un nouveau fichier.

```
Pour déchiffrer l'un des fichiers, il vous faudra remplacer chaque occurence :
- d'un '*' par un espace ' '
- d'un 'A' par un '.'
- d'un 'B' par un '_'
- d'un 'C' par un '-'
- d'un 'D' par un '|'
```

La fonction **"dechiffre_fichier"** prendra deux paramètres en entrée : le nom du fichier à déchiffrer, le nom du nouveau fichier chiffré. Vous utiliserez ensuite la fonction **afficher_fichier_crypte** pour lire le contenu du nouveau fichier et vérifier le déchiffrement du fichier initial.

Entrée [ ]:

```
def dechiffre_fichier(fichier_a_dechiffrer,fichier_dechiffre):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
dechiffre_fichier("fichier_chiffre_1.txt","fichier_dechiffre_1.txt")
```

Entrée [ ]:

```
#Tester votre fonction
dechiffre_fichier("fichier_chiffre_2.txt","fichier_dechiffre_2.txt")
```

Entrée [ ]:

```
#Tester votre fonction
dechiffre_fichier("fichier_chiffre_3.txt","fichier_dechiffre_3.txt")
```

Entrée [ ]:

```
#Tester votre fonction
afficher_fichier_crypte("fichier_dechiffre_1.txt")
```

Entrée [ ]:

```
#Tester votre fonction
afficher_fichier_crypte("fichier_dechiffre_2.txt")
```

Entrée [ ]:

```
#Tester votre fonction
afficher_fichier_crypte("fichier_dechiffre_3.txt")
```

[4.2] Ecrire la fonction **"chiffre_fichier"** qui chiffrera le contenu des trois fichiers précédemment déchiffré "fichier_chiffre_1.txt","fichier_chiffre_2.txt" et "fichier_chiffre_3.txt" et écrire le contenu chiffré dans un nouveau fichier.

```
    Pour chiffrer l'un des fichiers, il vous faudra remplacer chaque occurence :
    - d'un espace ' ' par un '*'
    - d'un '.' par un 'A'
    - d'un '_' par un 'B'
    - d'un '-' par un 'C'
    - d'un '|'par un 'D'
```

La fonction **"chiffre_fichier"** prendra deux paramètres en entrée : le nom du fichier à chiffrer, le nom du nouveau fichier déchiffré. Vous utiliserez ensuite la fonction **afficher_fichier_crypte** pour lire le contenu du nouveau fichier et vérifier que le fichier obtenu est le même que le fichier initial fichier_chiffre_n.

Entrée [ ]:

```
def chiffre_fichier(fichier_a_chiffrer,fichier_chiffre):
    # CODE A COMPLETER
```

Entrée [ ]:

```
#Tester votre fonction
chiffre_fichier("fichier_dechiffre_1.txt","fichier_rechiffre_1.txt")
```

Entrée [ ]:

```
#Tester votre fonction
chiffre_fichier("fichier_dechiffre_2.txt","fichier_rechiffre_2.txt")
```

Entrée [ ]:

```
#Tester votre fonction
chiffre_fichier("fichier_dechiffre_3.txt","fichier_rechiffre_3.txt")
```

Entrée [ ]:

```
#Tester votre fonction
afficher_fichier_crypte("fichier_rechiffre_1.txt")
```

Entrée [ ]:

```
#Tester votre fonction
afficher_fichier_crypte("fichier_rechiffre_2.txt")
```

Entrée [ ]:

```
#Tester votre fonction
afficher_fichier_crypte("fichier_rechiffre_3.txt")
```