TP8 – CRYPTAGE

OBJECTIFS :

Mettre en œuvre les algorithmes de cryptage de César et Vigénère. Revoir l'utilisation des chaines de caractères.

1. CODAGE DE CESAR :AVE CESAR (BWF DFTBS)

On cherche à crypter un texte t de longueur n composé de caractères en minuscules (soit 26 lettres différentes) représentés par des entiers compris entre 0 et 25 (0 ↔ a, 1 ↔ b, …, 25 ↔ z). Nous ne tenons pas compte des éventuels espaces.

Ainsi, le texte **ecolepolytechnique** est représenté par le tableau suivant où la première ligne représente le texte, la seconde les entiers correspondants, et la troisième les indices dans le tableau t.

e	c	o	l	e	p	o	l	y	t	e	c	h	n	i	q	u	e
4	2	14	11	4	15	14	11	24	19	4	2	7	13	8	16	20	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Ce codage est le plus rudimentaire que l'on puisse imaginer. Le principe est de décaler les lettres de l'alphabet vers la gauche de 1 ou plusieurs positions. Par exemple, en décalant les lettres de 1 position, le caractère **a** en **b**, ... le **z** en **a**. Le texte **avecesar** devient donc **fajhjxfw** pour un décalage de 5.

On crée la fonction ***decalage_lettre(t, d)*** qui prend en arguments le caractère t et un entier d ; et qui retourne le caractère t décalé de d positions.

```
def decalage_lettre(t,d):
    i = ord(t)-97
    i = (i+d) % 26
return chr(i+97)
```

*Q1. Recopier ces instructions et expliquer ce que font les fonctions : **ord() , chr()***

*Q2. Ecrire la fonction **codageCesar(txt,d)** qui prend en entrée une chaine de caractères txt et un entier d et qui renvoie la chaine de caractère décalée de d positions.*

*Q3. Que donne le codage du texte **lyceebellevue** en utilisant un décalage de 5 ?Tester votre fonction pour **avecesar** et **ecolepolytechnique**.*

*Q4. Vous pouvez améliorer la fonction en prenant en compte les espaces **codageCesar2(txt,d)** et tester avec **lycee bellevue** ou **ecole polytechnique***

*Q5. Écrire de même la fonction **decodageCesar**(txt, d) prenant les mêmes arguments mais qui réalise le décalage dans l'autre sens.*

Pour réaliser ce décodage, il faut connaître la valeur du décalage. Une manière de la déterminer automatiquement est d'essayer de deviner cette valeur. L'approche la plus couramment employée est de regarder la fréquence d'apparition de chaque lettre dans le texte crypté. En effet, la lettre la plus fréquente dans un texte suffisamment long en français est la lettre e.

*Q6. Écrire la fonction **frequences**(t') qui prend en argument la chaine de caractères t' représentant le texte crypté (sans espaces) ; et qui retourne un tableau de taille 26 dont la case d'indice i contient le nombre d'apparitions du caractère i dans t ($0 \leq i < 26$).*

*Q7. Écrire la fonction **decodageAuto**(t') qui prend en argument la chaine de caractères t' représentant le texte crypté ; et qui retourne le texte t d'origine (en calculant la clé pour que la lettre e soit la plus fréquente dans le texte décrypté).*

2. CODAGE DE VIGENERE

Au XVIème siècle, Blaise de Vigenère a modernisé le codage de César très peu résistant de la manière suivante.

Rappel de la méthode :

Au lieu de décaler toutes les lettres du texte de la même manière, on utilise un texte clé qui donne une suite de décalages. Prenons par exemple la clé **concours**. Pour crypter un texte, on code la première lettre en utilisant le décalage qui envoie le a sur le c (la première lettre de la clé). Pour la deuxième lettre, on prend le décalage qui envoie le a sur le o (la seconde lettre de la clé) et ainsi de suite. Pour la huitième lettre, on utilise le décalage a vers s, puis, pour la neuvième, on reprend la clé à partir de sa première lettre. Sur l'exemple **ecolepolytechnique** avec la clé **concours**, on obtient : (la première ligne donne le texte, la seconde le texte crypté et la troisième la lettre de la clé utilisée pour le décalage)

e	c	o	l	e	p	o	l	y	t	e	c	h	n	i	q	u	e
g	q	b	n	s	j	f	d	a	h	r	e	v	h	z	i	w	s
c	o	n	c	o	u	r	s	c	o	n	c	o	u	r	s	c	o

*Q8. Écrire la fonction **codageVigenere**(t, c) qui prend comme arguments une chaine de caractères t représentant le texte à crypter, et une chaine de caractères c donnant la clé servant au codage ; et qui retourne une chaine de caractères crypté t'.*

Vous pourrez utiliser une fonction auxiliaire decalage_lettre(t,d) qui convertit la lettre t en entier, qui calcule sont numéro de lettre crypté et qui renvoie la conversion de cet entier en lettre.

*Q9. Vous pouvez facilement écrire la fonction **decodageVigenère(t,c)** en utilisant la distance opposée. Eyt vous pouvez aussi écrire la fonction c**odageVignere2(t,c)** qui prend en compte les espaces dans la chaine de caractères t.*

*Tester votre fonction sur le code maitre corbeau **codageVignere2(text,'jean')** qui se trouve dans le fichier vigenère vide.py*

Maintenant, on suppose disposer d'un texte *t'* assez long, crypté par la méthode de Vigenère, et on veut retrouver le texte *t* d'origine. Pour cela, on doit trouver la clé *c* ayant servi au codage. On procède en deux temps :

1) détermination de la longueur *k* de la clé *c*,

2) détermination des lettres composant *c*.

La première étape est la plus difficile. On remarque que deux lettres identiques dans *t* espacées de $l \times k$ caractères (où *l* est un entier et *k* la taille de la clé) sont codées par la même lettre dans *t'*. Mais cette condition n'est pas suffisante pour déterminer la longueur *k* de la clé *c* puisque des répétitions peuvent apparaître dans *t'* sans qu'elles existent dans t. Par exemple, les lettres *t* et *n* sont toutes deux codées par la lettre *h* dans le texte crypté à partir de **ecolepolytechnique** avec **concours** comme clé. Pour éviter ce problème, on recherche les répétitions non pas d'une lettre mais de séquences de lettres dans *t'* puisque deux séquences de lettres répétées dans *t*, dont les premières lettres sont espacées par $l \times k$ caractères, sont aussi cryptées par deux mêmes séquences dans *t'*.

Dans la suite de l'énoncé, on ne considère que des séquences de taille 3 en supposant que toute répétition d'une séquence de 3 lettres dans *t'* provient *exclusivement* d'une séquence de 3 lettres répétée dans *t*. Ainsi, la distance séparant ces répétitions donne des multiples de *k*.

La valeur de *k* est obtenue en prenant le PGCD de tous ces multiples. Si le nombre de répétitions est suffisant, on a *de bonnes chances* d'obtenir la valeur de *k*. On suppose donc que cette assertion est vraie.

*Q10. Écrire la fonction **pgcd**(a, b) qui calcule le PGCD des deux entiers strictement positifs a et b par soustractions successives de ses arguments ou par division euclidienne.*

Par la suite, on va utiliser la fonction ***pgcdDesDistancesEntreRepetitions**(t', i)* qui prend en argument le texte crypté t' et un entier i ($0 \leq i \leq n-2$) qui est l'indice d'une lettre dans t' ; et qui retourne le pgcd de toutes les distances entre les répétitions de la séquence de 3 lettres $(t[i], t[i+1], t[i+2])$ dans la suite du texte $t[i+3], t[i+4], \ldots t[n-1]$. Cette fonction retourne 0 s'il n'y a pas de répétition.

```
def pgcdDesDistancesEntreRepetitions(t,i):
    assert i >=0 and i < len(t)-1-3
    taille = 0
    emprunte = t[i:i+3]
    for j in range(i+3,len(t)-3):
        if t[j:j+3] == emprunte:
            if taille==0:
                taille = -(j-i)
            else:
                taille = pgcd(abs(taille),j-i)
    if taille>0:
        return taille
    else:
        return 0
```

*Q11. Utiliser la fonction **pgcdDesDistancesEntreRepetitions**(t', i) pour écrire la fonction **longueurDeLaCle**(t') qui prend en argument le texte crypté t' ; et qui retourne la longueur k de la clé de codage.*

*Tester votre fonction sur le code maitre corbeau **text_sans_espaces***
print(longueurDeLaCle(codageVignere(text_sans_espaces,'jean')))

*Q12. Donner le nombre d'opérations réalisées par la fonction **longueurDeLaCle** en fonction de la longueur de t' ? (On ne comptera que le nombre d'appels à la fonction PGCD).*

*Q13. Une fois la longueur de la clé connue, écrire la fonction **trouver_la_cle(text2))** permettant de retrouver chacune des lettres de la clé à partir du texte crypté **text2**.*

*Tester avec **text2=codageVignere(text_sans_espaces,'jean')***
print(trouver_la_cle(text2))

*Q14. Écrire la fonction **decodageVigenereAuto**(t') qui prend en argument le tableau t' représentant le texte crypté ; et qui retourne le texte t d'origine. (On n'hésitera pas à recopier des parties de texte dans des tableaux intermédiaires).*

Table ASCII :

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□