Grand Devoir Maison 1

Pour le mardi 3 novembre 2026

Vous devrez rendre l'ensemble de votre travail sur **copie manuscrite**, y compris les programmes.

Le thème du sujet est la cryptographie : codage, décodage et cryptanalyse.

Certaines manipulations demandent d'aller sur le site `https://cahier-de-prepa.fr/tsi2x-monge/`, rubrique "Informatique 2.1" puis "Grands Devoirs", notamment pour valider certaines parties de votre GDM.

Exercice 1 (Préliminaires)**.**

Dans tout le sujet, nous travaillerons avec un alphabet étendu à 100 symboles. **Les deux fonctions que nous allons créer ici nous servirons dans toute la suite du devoir.** Pour des raisons évidentes de simplification de manipulation, nous souhaitons transformer notre alphabet en nombres. Ainsi chaque lettre correspondra à un nombre entre 0 et 99 d'après le tableau suivant :

code	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
chaîne	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
code	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
chaîne	u	v	w	x	y	z	(	)	[	]	{	}	<	>	/	=	+	-	*	%
code	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
chaîne	~	^	&	|	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
code	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
chaîne	Q	R	S	T	U	V	W	X	Y	Z	.	,	;	:	!	?	’	"	0	1
code	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
chaîne	2	3	4	5	6	7	8	9	_		\n	$	@	#	\\	à	ç	é	è	ê

Le code 43 est un caractère pipe (altgr+6), le 76 est un guillement simple, le 77 un guillement double, le 88 un underscode ("tiret du 8"), le 89 un espace, le 90 un saut de ligne et le 94 un anti-slash (altgr+8).

1. Créer une fonction `text2code(message)` qui prend en paramètre une chaîne de caractères et retourne la liste des codes correspondants.
   Par exemple, `text2code("73 Monge")` donne la liste `[85, 81, 89, 56, 14, 13, 6, 4]`.
2. Créer la fonction inverse `code2text(code)`, qui retourne la chaîne de caractères codée par la liste `code`.
   Par exemple, l'appel de `code2text([63, 18, 8, 80, 37, 79])` donne la chaîne `"Tsi2-1"`.

Exercice 2 (chiffre de César)**.**

Pour sécuriser ses échanges avec ses généraux, César chiffrait ses messages. Il commençait par supprimer tous les espaces de son texte puis décalait la position des lettres de 3 vers la droite : le A était remplacé par D, le B par E, etc, le Z par C.

Nous utiliserons une généralisation de cette technique : notre alphabet comportera les 100 symboles du préliminaire et le décalage sera paramétrable.

1. Expliquer pourquoi le décalage de César (de 3 positions) correspond à effectuer l'opération suivante pour notre alphabet :

$$\text{code}_{\text{chiffré}} = \text{code}_{\text{clair}} + 3 \qquad (\%\ 100\ )$$

2. Construire une fonction `code_cesar(message, decalage)` qui prend en argument une chaîne de caractères `message` et un entier `decalage` et qui retourne la chaîne de caractères chiffrée du `message` selon la méthode de César avec le `decalage`.

On pourra tester la fonction sur la chaîne `'Bonjour le monde !'` qui donne, avec un décalage de 58, la chaîne `'d;,X;O?DZSD.;,RSD<'`.

3. Construire une fonction `decode_cesar(chiffre, decalage)` qui décode la chaîne chiffrée `chiffre` à l'aide du `decalage`.
(indication : on pourra se servir de la fonction de codage pour décoder...)

La chaîne `'gOR4?<GT-V12GS?!'` a été chiffrée avec un décalage de 39.

4. Nous allons maintenant cracker le chiffre de César, c'est-à-dire retrouver le texte original sans pour autant connaître le décalage.

La faille dans le chiffre se situe dans le fait que le nombre de décalages n'est pas très grand. On peut donc tester toutes les possibilités.

La fonction suivante permet d'ouvrir le fichier texte `fichier`, de lire son contenu sous forme de chaîne de caractères, de refermer le fichier et de renvoyer le contenu lu :

```
def lecture(nom_fichier):
    fichier_id = open(nom_fichier, 'r', encoding='utf8')
    contenu = fichier_id.read()
    fichier_id.close()
    return contenu
```

Ainsi, avec la ligne `texte = lecture('fichier.txt')`, la variable `texte` contient le texte contenu dans le fichier nommé `fichier.txt`. À noter que le fichier doit se trouver dans le même répertoire que le script Python qui le lit.

Sur le site compagnon, se trouve le fichier `cesar.txt`. Il contient un message chiffré.

Créer une fonction `essai_cesar(crypte)` qui tente de décoder le message `crypte` avec tous les décalages possibles et affiche (avec `print`), pour chaque décalage, le décalage et les 20 premiers caractères de la chaîne "décodée".

Trouver le bon décalage, décoder complètement le message et suivre les instructions.

Exercice 3 (Méthode de Vigenère)**.**

Le code de César est faible car il a très peu de clés. Il a aussi une autre faiblesse : le décalage est constant tout du long du texte. Pour éviter ce problème, Vigenère (au XVI^{ème} siècle) a eu l'idée de changer le décalage pour chaque lettre. Voici comment fonctionne son chiffre : il part d'un mot secret. Ce mot secret est une clé qu'il faut avoir pour (dé)coder. Ce mot est répété autant de fois qu'il le faut pour avoir la même longueur que le texte à chiffrer. Ensuite, chaque lettre du message est codée en utilisant un décalage égale à la lettre du mot secret en rapport avec celle-ci.

Voici un exemple pour mieux comprendre : le mot secret est 'Tsi2.1' et le texte à chiffré est 'Lycée - Monge.'.

texte clair	L 55	y 24	c 2	é 97	e 4	89	- 37	89	M 56	o 14	n 13	g 6	e 4	. 70
mot secret	T 63	s 18	i 8	2 80	. 70	1 79	T 63	s 18	i 8	2 80	. 70	1 79	T 63	s 18
texte chiffré	18 s	42 &	10 k	77 "	74 !	68 Y	0 a	7 h	64 U	94 \	83 5	85 7	67 X	88 _

On additionne (modulo 100, bien sûr) les codes pour avoir la lettre codée. La version chiffrée de `"Lycée - Monge"` est donc `'s&k"!YahU\57X_'`. Comme on le voit, la lettre 'e' est codée '!' puis 'X'... Cela donne un nombre de combinaisons très grand et rend l'analyse par fréquences inapplicable. Ce code a été très efficace jusqu'au XIX$^{\text{ème}}$ siècle.

1. Construire une fonction `code_Vigenere(message, cle)` qui prend en argument deux chaînes de caractères `message` et `cle` et qui retourne la chaîne de caractères du message codé par la méthode de Vigenère.
2. Décrire la méthode de décodage du chiffre de Vigenère.
3. Construire la fonction de décodage `decode_Vigenere(chiffre, cle)`.
4. Décoder le fichier `vigenere.txt` à l'aide de la clé `"Tsi2.1, lycée Monge"` et suivre les instructions.

Exercice 4.

Le code de Vigenère ne résiste pas à une attaque poussée de cryptanalyse. Sa faiblesse vient du fait que le mot secret est déroulé de façon cyclique le long du texte à déchiffrer. On peut certes augmenter la taille de la clé, mais il faut qu'elle soit raisonnable à mémoriser...

Je vous propose une autre façon de chiffrer, qui mélange un peu toutes les lettres. D'abord, on peut supposer que le message a un nombre de lettres pair (sinon, on rajoute un espace à la fin du texte). Ensuite, on effectue un chiffrement par la méthode de Vigenère. Puis, pour masquer le chiffre, on mélange les lettres : pour chaque paire de lettres, on échange dizaines et unités, d'une part et on échange la dizaine d'une lettre avec la dizaine de l'autre, d'autre part. Enfin, on procède à un deuxième chiffrement de Vigenère avec la même clé. Il ne reste plus qu'à convertir le tableau obtenu en lettres pour avoir notre message chiffré.

texte clair	B	i	e	n	v	e	n	u	e	
	45	8	4	13	21	4	13	20	4	89
mot secret	T	s	i	2	.	1	T	s	i	2
	63	18	8	80	70	79	63	18	8	80
1$^{\text{er}}$ chiffre de Vigenère	8	26	12	93	91	83	76	38	12	69
inversion 1	80	62	21	39	19	38	67	83	21	96
inversion 2	60	82	31	29	39	18	87	63	91	26
2$^{\text{ème}}$ chiffre de Vigenère	23	0	39	9	9	97	50	81	99	6
	x	a	%	j	j	é	G	3	ê	g

La version chiffrée de `"Bienvenue"` est donc `"xa\%jjéG3êg"` avec le mot clé `"Tsi2.1"`.

1. Écrire une fonction `code_vigperm(message, cle)` qui implémente ce chiffre.
2. Décrire la méthode de décodage à utiliser ici.
3. Écrire une fonction `decode_vigperm(chiffre, cle)` qui déchiffre le message `chiffre` codé par la fonction précédente.
4. Décoder le fichier `vigperm.txt` à l'aide de la clé `"Tsi2.1, lycée Monge"` et suivre les instructions.

Pour de l'aide éventuelle :
mail : `vincent.deveaux@ac-grenoble.fr`
site compagnon : `https://cahier-de-prepa.fr/tsi2x-monge/`, rubrique "Informatique 2.1" puis "Grands Devoirs"