Partie 1

Exercice 1 : Substitution monoalphabétique : Code de César

Le chiffre de César fonctionne par décalage des lettres de l'alphabet.

Cet algorithme de chiffrement utilise une fonction périodique pour transformer les rangs de chaque lettre. On utilise l'alphabet suivant :

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

- **Question 1** : Compléter l'algorithme du chiffrement de César. On suppose qu'il existe une clé `c` correspondant au décalage utilisé. Le message clair est `m` et le message chiffré est `m_chiffre`. On dispose de 2 fonctions, `rang` qui retourne le rang d'un caractere `c` dans l'alphabet, et `lettre` qui retourne le caractère correspondant au rang `i`.

```
Pour chaque lettre l du message m:
  l_chiffre = ...
  m_chiffre = ...
```

- **Question 2** : Quelle clé a été utilisée pour chiffer ce texte (avec l'algorithme de César) ?

$$jyfwavnyhwoplhwwspxbll$$

- **Question 3** : Décrypter ce message.
- **Question 4** : Proposer un programme en python qui chiffre un message clair `m` en un message codé selon la clé numérique `cle`. Programmer les fonctions `rang`, `lettre`, ainsi que la fonction de chiffrement `chiffrer`.

Aide :

- La fonction `ord('x')` retourne un entier correspondant à la position d'un caractère dans la table ascii. Par exemple, `ord('A')` retourne 65, `ord('B')` retourne 66,...
- La fonction `chr(N)` retourne le caractère de rang N : `chr(65)` retourne 'A'.

Partie 2

Substitution polyalphabétique : Chiffrement de Playfair

A partir des explications données dans la video :

- **Question 1** : Construisez la matrice pour l'algorithme de Playfair avec la clé *estienne*.
- **Question 2** : Chiffrer le message : *ACTIONREACTION*
- **Question 3** : S'agit-il d'une méthode utilisant la substitution monoalphabetique, ou polyalphabetique ?
- **Question 4** : Est-ce qu'avec cette méthode, le decryptage peut être facilité par l'analyse fréquentielle ?

Partie 3

Frequences des lettres dans un texte

La fonction suivante retourne une liste de 26 valeurs de type *float*, donnant dans l'ordre de l'alphabet, la frequence pour chaque lettre dans un texte :

```
def freq(m):
    frequences = [0]*26
    n = len(m)
    for c in m:
    ...
```

- **Question 1** : Compléter le script de la fonction `freq`.

Soient les deux textes chiffrés suivants :

a. DGFHTMOMEIAMTLMFGOKFGOKEGDDTRWEIAKZGFGFSTEKGOMLASTTIFG FOSTLM FTFGOK MGWMFG OKRTSA JWTWTA WDTFMG FDAOLT WMOSSA FGOKET WKRWFD TEIAFM ROAZSG MOFKOT FFTXAW MLARGW ETWKJW AFROSD OAWSTA WDAMOF HGWKDT STEITK SADAOF

b. YOHGMV MFCBRB GWFNIZ ZPSURW FUOIPU WYITFA NIETGG DOCKAC PQE- BEW PMXEMK VGRAIL KWWXZO CILGPM QOVCGE GMYEBQ RYACJM WX- ULFR VQMDCY LZFYZM YTPCRF DCRJNS FIHIQG RZEPRC VWMDIL KGYDQO RVFMXM CRCNIM UGRBKR BOOIUG PQCBVZ NEYACE

- **Question 2** : L'un des 2 a été chiffré avec un algorithme de substitution mon-alphabetique, et l'autre, poly-alphabétique. Lequel est mono-alphabétique ?

Partie 4

Chiffrement symétrique par la fonction XOR

- **Question 1** : La fonction XOR est la fonction du OU EXCLUSIF.

Donner la table de vérité de la fonction XOR

- **Question 2** : A partir des lignes suivantes, vérifier que l'opérateur ^ en python réalise la fonction XOR sur 2 nombres écrits en valeur décimale :

```
> 3 ^ 4
7
> 4 ^ 5
1
```

- **Question 3** : Programmer la fonction `xor` en python qui retourne le resultat de XOR appliqué aux 2 paramètres a et b.

Exemple :

```
>>> xor(53,23)
34
>>> xor(34,23)
53
```

- **Question 4** : Peut-on utiliser la fonction XOR pour chiffrer PUIS dechiffrer un message, avec la MEME clé de chiffrement ? Expliquer à partir de l'exemple ci-dessus.
- **Question 5** : On s'interresse maintenant à la programmation de la fonction XOR pour chiffrer un message à partir d'un masque :

$$message = "CETTEPHRASEESTVRAIMENTTRESTRESLONGUEMAISCESTFAITEXPRES"$$

Le masque (correspond à la clé) aura une longueur inférieur au message. On utilise le masque selon la méthode de chiffrement polyalphabetique vue en cours.

Le chiffrement demande de faire correspondre les caractères avec des nombres entiers. On utilisera pour cela un alphabet de 32 caractères (2^5). Chaque lettre de l'alphabet correspondra alors à un nombre compris entre 0 et 31.

- **Question** : Lorsque l'on place 2 entiers inférieurs à 32 dans la fonction XOR, celle-ci retourne un entier également inférieur à 32. Pourquoi ? *(Representer ces nombres en binaire, sur un octet, et verifier cette affirmation).*
- **Question 6** : Compléter la fonction `chiffre(message, masque)` qui chiffre message en le XORant avec masque.

Cette fonction doit pouvoir aussi servir à déchiffrer le message chiffré.

```
alphabet=[chr(i+ord('A')) for i in range(26)] + ['*','/','@','&','!','à
    ']
def chiffre(message, masque):
    assert len(message)>=len(masque)
    message_chiffre = ""
    for i in range(len(message)):
        indice_lettre = alphabet.index(message[i])
        indice_masque = ...
        val = xor(...,...)
        message_chiffre+=alphabet[val]
    return ...
```

- **Question 7** : Utiliser la fonction `chiffre` pour effectuer le chiffrement puis le dechiffrement du *message*.
- **Question 8** : Pensez vous que cette méthode de chiffrement est assez resistante face à une attaque de cryptoanalyse ? Si oui, pourquoi ?
- **Question 9** : On souhaite maintenant transfomer ce programme en Programmation Objet. Voici la manière avec laquelle on souhaite l'utiliser :

```
>>> message = "CETTEPHRASEESTVRAIMENT..."
>>> masque = "WAPITI"
>>> com1 = Chiffre_XOR(message,masque)
>>> message_chiffre = com1.chiffre
>>> com2 = Chiffre_XOR(message_chiffre,masque)
>>> message_clair = com2.chiffre
```

La classe `Chiffre_XOR` possedera les attributs suivants :

- `alphabet`
- `message`
- `masque`

Et la méthode de classe `chiffre`, sur le modèle de celle vue plus haut.

La fonction `xor` ne fait pas partie des mathodes de la classe `Chiffre_XOR`, mais peut être utilisée à l'interieur de celle-ci.

Ecrire le script de la classe `Chiffre_XOR`

Partie 5

Exercice tiré d'un sujet de bac : chiffrements symétriques/ asymétriques

1) Après un chiffrement symétrique on obtient la message suivant : ri. Sachant que la clé de chiffrement est : 00001010 (la clé est directement donnée en binaire), déterminez le message d'origine.

On donne l'extrait de la table ASCII suivant :

lettre	code binaire	lettre	code binaire
a	01100001	t	01110100
b	01100010	v	01110110
c	01100011	w	01110111
d	01100100	x	01111000
e	01100101	y	01111001
f	01100110	z	01111010
i	01101001	| (vertical bar)	01111100
r	01110010	{ (left opening brace)	01111101
s	01110011	~ (tilde)	01111110

2) Un utilisateur B souhaite échanger un message chiffré avec un utilisateur A en utilisant un chiffrement asymétrique. A possède une clé publique (AKpub) et une clé privée (AKpriv). B possède une clé publique (BKpub) et une clé privée (BKpriv). B souhaite chiffrer un message m afin de pouvoir l'envoyer à A :

a) Quelle clé va être utilisée par B pour chiffrer le message m ?

Qb) uelle clé va être utilisée par A pour déchiffrer le message m ?

3) Expliquez en quelques lignes le principe du protocole HTTPS (on s'intéressera uniquement à l'aspect Sécurité du protocole)

Partie 6

Exercice tiré du bac 2021

Pour chiffrer un message, une méthode, dite du masque jetable, consiste à le combiner avec une chaîne de caractères de longueur comparable. Une implémentation possible utilise l'opérateur XOR (ou exclusif).

Dans la suite, les nombres écrits en binaire seront précédés du préfixe 0b.

1) Pour chiffrer un message, on convertit chacun de ses caractères en binaire (à l'aide du format Unicode), et on réalise l'opération XOR bit à bit avec la clé.

Après conversion en binaire, et avant que l'opération XOR bit à bit avec la clé n'ait été effectuée, Alice obtient le message suivant :

```
m = 0b 0110 0011 0100 0110
```

a) Le message m correspond à deux caractères codés chacun sur 8 bits : déterminer quels sont ces caractères. On fournit pour cela la table ci-dessous qui associe à l'écriture hexadécimale d'un octet le caractère correspondant (figure 2). Exemple de lecture : le caractère correspondant à l'octet codé 4A en hexadécimal est la lettre J.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

b) Pour chiffrer le message d'Alice, on réalise l'opération XOR bit à bit avec la clé suivante :

```
k = 0b 1110 1110 1111 0000
```

Donner l'écriture binaire du message obtenu.

2)

a) Dresser la table de vérité de l'expression booléenne suivante :

```
(a XOR b) XOR b
```

b) Bob connaît la chaîne de caractères utilisée par Alice pour chiffrer le message. Quelle opération doit-il réaliser pour déchiffrer son message ?

Partie 7

Corrections

7.1 Exercice 1

```
1  def cesar(mot,c):
2      m=""
3      for l in mot:
4          l_chiffre = chr((ord(l)-ord('a')+c)%26+ord('a'))
5          m+=l_chiffre
6      return m
7  >>> cesar('aaa',25)
8  'zzz'
9  # essais
10 texte="jyfwavnyhwoplhwwspxbll"
```

```
for c in range(25):
    print('cle:{} message:{}'.format(c,cesar(texte,c)))
...
cle:19 message:cryptographieappliquee
```

7.2 Exercice 3

```
def freq(m):
    frequences = [0]*26
    n = len(m)
    for c in m:
        frequences[ord(c)-ord('A')]+=1/n
    return frequences

message1 = '
   DGFHTMOMEIAMTLMFGOKFGOKEGDDTRWEIAKZGFGFSTEKGOMLASTTIFGFOSTLMFTFGOKMGWMFGOKRTSA
   '
L1 = freq(list(message1))

import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])

etiquettes = [chr(97+i) for i in range(26)]
valeurs = L1

ax.bar(etiquettes, valeurs)

plt.title("Histogramme")  # Titre du graphique
plt.ylabel('Nombre')  # Titre de l'axe y
plt.xlabel('alphabet')
plt.show()  # Affichage d'une courbe
```

7.3 Exercice 4

```
alphabet=[chr(i+ord('A')) for i in range(26)] + ['*','/','@','&','!','à
   ']
def chiffre(message, masque):
    assert len(message)>len(masque)
    message_chiffre = ""
    for i in range(len(message)):
        indice_lettre = alphabet.index(message[i])
        indice_masque = alphabet.index(masque[i%len(masque)])
        val = xor(indice_lettre,indice_masque)
        message_chiffre+=alphabet[val]
    return message_chiffre

message = "CETTEPHRASEESTVRAIMENTTRESTRESLONGUEMAISCESTFAITEXPRES"
masque = "WAPITI"
secret = chiffre(message,masque)
print(secret)
```