Exercice 1

Un système est protégé par un mot de passe. Après un essai infructueux le système attend 1 seconde avant de redemander le mot de passe. Combien de temps faudra-t-il pour pénétrer le système dans les cas suivants :

1. le mot de passe est un prénom ;
2. c'est un mot du dictionnaire ;
3. il est composé de 4 chiffres ;
4. il fait 8 caractères alphanumériques (y compris les 15 signes de ponctuations)

Exercice 2

Lors de la transmission d'un message , on souhaite s'assurer de son intégrité .
Pour cela on utilise une fonction de **hachage**.
La valeur donnée par cette fonction est appelée **empreinte** ou **haché** du message.
On porra consulter le lien ci-dessous.
https://fr.wikipedia.org/wiki/Fonction_de_hachage_cryptographique

1. Les fonctions de hachage utilisées sont MD5 , SHA1 (pas suffisamment sûres) ou SHA-256.

   Voici un exemple de hachage d'un document .pdf

```
from hashlib import sha256
#Ouverture du fichier en lecture et en  mode binaire
with open("securite-crypto.pdf" ,"rb") as f:
    data=f.read()
    #calcul de l'empreinte
    h =sha256(data)
    #affichage de l'empreinte en hexadécimal
    print(h.hexdigest())
```

   Exécutez cette fonction et noter l'empreinte.

2. Modifier le document précédent (en ajoutant une annotation par exemple). Appliquer à nouveau la fonction de hachage.

3. Nous alons utiliser ici une fonction de hachage toute simple.

```
def somme_code_ascii(mot):
"""

"""
return sum(ord(lettre) for lettre in mot)
```

   Exécutez cette fonction avec les mots "Gauss" , "Einstein" , "Poincarre"

4. Exécuter à nouveau cette fonction avec les mots "pile" , "lipe" , "piel".
   Que constate-t-on?

5. Modifier la fonction de hachage précédente `somme_code_ascii(mot)` en tenant compte de la valeur du caractère ainsi que de sa position dans le texte. Le nom de la nouvelle fonction est `somme_code_ascii_v2(mot)` .

6. Exécuter la fonction précédente avec les messages:

   - "oiseau"
   - "Un poème de Victor Hugo"
   - "Un poème de Victor Hugo" * 1_000_000.

   Quels problèmes sous-jacents apparaissent lors de l'utilisation de cette fonction ?

7. Pour remédier au problème de la longeur de l'empreinte , on peut utiliser les résultats précédents modulo un entier .

   Faire les modifications nécessaires sur `somme_code_ascii_v2(mot)` puis exécutez sur les exemples précédents.

   Quel(s) problème(s) rest(ent) à résoudre?

Exercice 3

Au XVIe siècle, Blaise de Vigenère a modernisé le codage de César, très peu résistant, de la manière suivante. 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 XYZ . Pour crypter un texte, on code la première lettre en utilisant le décalage qui renvoie A sur X (la première lettre de la clé).

Pour la deuxième lettre, on prend le décalage qui envoie le A sur le Y (deuxième lettre de la clé) et ainsi de suite. Pour la quatrième lettre du texte (la clé en contient que 3) on reprend à partir de la première lettre de la clé. Sur l'exemple VIGENERE avec la clé XYZ, on obtient :

V	I	G	E	N	E	R	E
21	8	6	4	13	4	17	4
X	Y	Z	X	Y	Z	X	Y
23	24	25	23	24	25	23	24
18	6	5	1	11	3	14	2
S	G	F	B	L	D	O	C

La quatrième ligne donne le code de la clé (décalage à appliquer), la cinquième ligne donne la somme de la deuxième et la quatrième (modulo 26), la sixième ligne donne le texte crypté.

Ecrire une fonction codageVigenere qui prend en paramètres une liste représentant le texte à crypter et une liste d'entiers donnant la clé servant au codage, et qui retourne une liste contenant le texte crypté.

Exercice 4

Le cryptage XOR est un système de cryptage basique mais pas trop limité. Ainsi, il a beaucoup été utilisé dans les débuts de l'informatique et continue à l'être encore aujourd'hui car il est facile à implémenter, dans toutes sortes de programmes.

Le XOR est un opérateur logique qui correspond à un "OU exclusif" : c'est le (A ou B) qu'on utilise en logique .

Chaque caractère du message à coder est représenté par un entier, le code ASCII (Consulter sur le web une table de codage ASCII). Ce nombre est lui-même représenté en mémoire comme un nombre binaire à 8 chiffres (les bits). On choisit une clé que l'on place en dessous du message à coder, en la répétant autant de fois que nécessaire, comme dans le codage de Vigenère.

Le message et la clé étant converti en binaire, on effectue un XOR, bit par bit, le 1 représentant VRAI et le 0 FAUX. Le résultat en binaire peut être reconverti en caractères ASCII et donne alors le message codé.

Exemple avec le texte « MESSAGE » et la clé « CLE » :

Lettre	M	E	S	S	A	G	E
Code ASCII	77	69	83	83	65	71	69
Code binaire	01001101	01000101	01010011	01010011	01000001	01000111	01000101
Clé en binaire	01000011	01001100	01000101	01000011	01001100	01000101	01000011
Lettre codée	00001110	00001001	00010110	00010000	00001101	00000010	00000110
Lettre codée ascii	SO Shift out	HT (horizontal tab)	SYN Sync idle	DLE (Data Link ESC)	CR (Carriage return)	STX Start of text	ACK

1. Donner l'expression permettant de calculer le caractère codé d'une lettre l en utilisant le principe de codage XOR avec une lettre c d'une clé.

   Pour rappel , en Python l'opérateur ^ désigne un XOR.

2. Ecrire une fonction codageXOR qui prend en paramètres le texte à crypter et la clé servant au codage, et qui retourne le texte crypté selon le codage XOR.

3. Vérifier que A **XOR** (A **XOR** B) = A et B **XOR** (A **XOR** B) = B

   En déduire une méthode de décodage connaissant la clé.