Exercices corrigés (Série 5)

Algorithmes de cryptage

Exercice 1 :

*On veut crypter une chaine de caractères données **CH** dont la taille ne dépasse pas 50 caractères en une chaine résultat **Res** de la manière suivante : parcourir la chaine **CH** de gauche à droite en comptant le nombre d'occurrences successives de chaque caractère de la chaine **CH**, puis de ranger la chaine **Res**, ce nombre suivi du caractère e question.*

*Ecrire un programme Python permettant de saisir la chaine **CH** qui doit être non vide et formée uniquement par des lettres alphabétiques, puis de former et d'afficher la chaine **Res** selon le principe décrit précédemment.*

- ***Exemple :***

*Si **CH**='aaaFyBsssssssssssazz' alors la chaine **Res** qui sera affichée est '3a1F1y1B12s1a2z'*

⇒ *Corrigé*

```
def crypter(CH):
    res=''
    if len(CH)<=50:
        cpt=1
        for i in range(len(CH)-1):
            if CH[i]==CH[i+1]:
                cpt+=1
            else:
                res+=str(cpt)+CH[i]
                cpt=1
        res+=str(cpt)+CH[-1]
    return res
```

Exercice 2 :

On se propose d'écrire un programme qui permet de saisir et de crypter un mot M non vide, composé uniquement par des lettres majuscules et d'afficher le mot crypté MC.

La méthode de cryptage est la suivante :

- Pour chaque lettre, déterminer son nombre d'occurrence (apparition) n dans le mot M.
- Déterminer K qui est égal à 2*n si n est impair et sera égal à (n DIV 2) si n est pair.
- Remplacer chaque lettre par $K^{ième}$ lettre qui la suit dans l'intervalle de l'alphabet ['A'…'Z'].
- Pour les dernières lettres, on rend dès le début, par exemple si K=3, on remplacera 'A' par 'D','B' par 'E', 'C' par 'E'… 'Y' par 'B' et 'Z' par 'C'.

- ***Exemple :***

Pour le mot 'HAPPY'

	'H'	'A'	'P'	'P'	'Y'
Nombre d'occurrence	1	1	2	2	1
La valeur de K	1*2=2	1*2=2	2 DIV 2=1	2 DIV 2=1	1*2+2
La lettre de remplacement	'J'	'C'	'Q'	'Q'	'A'

Le mot crypté sera :'JCQQA'

- ***Travail à faire :***

Ecrire un programme Python qui permet de saisir un mot non vide et composé uniquement par des lettres majuscules, puis d'afficher le mot crypté selon le principe décrit ci-dessus.

⇒ *Corrigé*

```
def crypter_mc(s):
    alpha='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    res=''
    for lettre in s:
        occurence=s.count(lettre)
        pos=0
        if occurence%2==0:
            pos=occurence//2
        else:
            pos=occurence*2
        ordre=alpha.find(lettre)
        indice=pos+ordre
        if indice>25:
            indice=indice%26
        res+=alpha[indice]
    return res
```

Exercice 3 :

Un des plus anciens systèmes de cryptographie (aisément déchiffrable) consiste à décaler les lettres d'un message pour le rendre illisible. Ainsi, les A deviennent des B, les B des C, etc.

Et les Z deviennent des A

➲ ***Exemple :***

*Si **CH**='ABCCZABEY' alors la chaine **Res** crypté qui sera affichée est 'BCDDABCEZ'*

Travail à faire :

*Ecrivez un programme qui demande une chaine **CH** à l'utilisateur et qui la code dans une chaine **Res** selon ce principe.*

⇒ *Corrigé*

```
def crypter(ch):
    alpha='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    res=''
    for caractere in ch:
        indice=alpha.find(caractere)
        pos=indice+1
        if pos==26:
            pos=0
        res+=alpha[pos]
    return res
```

Exercice 4- le chiffre de César :

*Une amélioration (relative) du principe utilisé dans **l'exercice 2** consiste à opérer avec un décalage non de 1, mais d'un nombre quelconque de lettres. Ainsi, par exemple, si l'on choisit un décalage de 3, les A deviennent des E, les B des E, etc.*

Et les Z deviennent C

Réalisez un programme python sur le même principe que le précédent, mais qui demande en plus quel est le décalage à utiliser.

⇒ *Corrigé*

```
def cesar(s,d):
    alpha='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    res=''
    for lettre in s:
        ordre=alpha.find(lettre)
        indice=ordre+d
        if indice>25:
            indice=indice%26
        res+=alpha[indice]
    return res
```

Exercice 5 :

Une technique ultérieure de cryptographie consista à opérer non avec un décalage systématique, mais par une substitution aléatoire. Pour cela, on utilise un alphabet-clé, dans lequel les lettres se succèdent de manière désordonnée, par exemple :

HYLUJPVREAKBNDOFSQZCWMGITX

C'est cette clé qui va servir ensuite à coder le message. Selon notre exemple, les A deviendront des H, les B des Y, les C des L, etc.

➲ ***Exemple :***

*Si **CH**='ABCDEFZ' alors la chaine **Res** crypté qui sera affichée est 'HYLUJPX'*

Travail à faire :

*Ecrire un programme Python qui effectue ce cryptage (l'alphabet-clé sera saisi par l'utilisateur, et on suppose qu'il effectue une saisie correcte) sur une chaine de caractères **CH** et stocker le résultat dans **Res**.*

⇒ *Corrigé*

```
def crypter_alea(msg):
    cle='HYLUJPVREAKBNDOFSQZCWMGITX'
    alpha='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    res=''
    for caractere in msg:
        indice=alpha.find(caractere)
        res+=cle[indice]
    return res
```

Exercice 6 - le chiffre de Vigenère :

Un système de cryptographie beaucoup plus difficile à briser que les précédents fut inventé au XVIe siècle par le français Vigenère. Il consistait en une combinaison de différents chiffres de César.

On peut en effet écrire 25 alphabets décalés par rapport à l'alphabet normal :

- L'alphabet qui commence par B et finit par …YZA
- L'alphabet qui commence par C et finit par …ZAB
- etc.

Le codage va s'effectuer sur le principe du chiffre de César : on remplace la lettre d'origine par la lettre occupant la même place dans l'alphabet décalé.

Mais à la différence du chiffre de César, un même message va utiliser non un, mais plusieurs alphabets décalés. Pour savoir quels alphabets doivent être utilisés, et dans quel ordre, on utilise une clé.

Si cette clé est "VIGENERE" et le message "Il faut coder cette phrase", on procèdera comme suit :

La première lettre du message, I, est la 9e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la première lettre de la clé, V. Dans cet alphabet, la 9e lettre est le D. I devient donc D.

La deuxième lettre du message, L, est la 12e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la deuxième lettre de la clé, I. Dans cet alphabet, la 12e lettre est le S. L devient donc S, etc.

Quand on arrive à la dernière lettre de la clé, on recommence à la première.

Ecrire un programme python qui effectue un cryptage de Vigenère, en demandant bien sûr au départ la clé à l'utilisateur.

⇒ *Corrigé*

```
def Vigenere(msg,cle):
    alpha='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    res=''
    i=0
    for caractere in msg:
        debut=alpha.find(cle[i])
        pos=alpha.find(caractere)
        indice=pos+debut
        if indice>25:
            indice-=26
        res+=alpha[indice]
        i+=1
        if i>=len(cle):
            i=0
    return res
```