CoAnim - Cryptologie : Partie 1

1- LE CHIFFREMENT DE CESAR :

La cryptologie est utilisée depuis l'Antiquité, principalement dans le domaine militaire, pour éviter que les informations sur une armée ou sur un plan d'attaque ne tombent dans les mains de l'ennemi. La plus célèbre méthode de chiffrement de l'Antiquité est le chiffre de César, utilisé par Jules César. Cette méthode, aussi appelée **chiffrement par décalage**, consiste à décaler chaque lettre d'un message par la lettre de l'alphabet située à une distance fixée. Par exemple, si la distance est 3, la lettre A est remplacée par la lettre D, la lettre B par E, etc. ; et la lettre Z par C.

Par exemple, le message clair : ATTAQUEZ A L AUBE

est transformé, en utilisant le chiffre de César avec une distance de 3, en :

DWWDTXHC D O DXEH

La **distance** est un nombre compris entre 0 et 25**.** Si on choisit la distance 0, on ne change aucune lettre et le message chiffré reste identique au message clair, et donc lisible. L'alphabet latin étant composé de 26 lettres, si on choisit une distance de 26, on ne change aucune lettre. Au-delà de 26, le chiffrement ne sera pas différent de celui de la distance modulo 26. Par exemple, une distance de 29 donnera un message chiffré identique à une distance de 29 modulo 26 = 3.

- alphabet = "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"
- codage à la main :

A	T	T	A	Q	U	E	Z		A		L		A	U	B	E

Exercice 1. :

Compléter le script de la fonction *chiffrementCesar()* donnée ci-contre.

L'exécution de ce code donnera dans la console :

```
# script des fonctions
def chiffrementCesar(texte,cle,code) :
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    newTexte = ""
    ⇨ Script à compléter
    return newTexte

# programme principal
texteEnClair = "attaquez a l aube"

texteChiffre = chiffrementCesar(texteEnClair,3,True)
print(texteChiffre)

texteDeChiffre = chiffrementCesar(texteChiffre,3,False)
print(texteDeChiffre)
```

```
>>> (executing file "chriffrementDeCesar.py")
dwwdtxhc d o dxeh
attaquez a l aube
```

⇨ Ecrire le code dans un fichier nommé *chiffrementDeCesar_votreNom.py* .

On rappelle ci-dessous quelques fonctions natives de pythons, utiles ici :

Point Cours :

- Méthode *index()* : retourne l'index de l'élément mis en argument …

  Si `mot = "cesaretcleopatre"`

  Alors `mot.index("a")` renvoie la valeur `3`

- Opérateur *modulo* `%` : renvoie le reste de la division euclidienne

4%26 =	28%26 =
26%26 =	0%26 =

- Parcours d'un string :

```
mot = "string"
for caractere in mot :
    print(caractere , end=" ")
```

  donne dans la console : `s t r i n g`

- Récupérer le caractère d'index *i* dans un string :

  Si `mot = "cesaretcleopatre"`

  Alors : `mot[3]` renvoie la valeur `'a'`

- Pour **concaténer** des strings, on utilise l'opérateur + :

  `"cesar"+"et"+"cleopatre"` renvoie

  `'cesaretcleopatre'`

CORRIGE

```
def chiffrementCesar(texte,cle,code) :
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    newTexte = ""
    if not code : cle = -cle
    for c in texte :
        if c != " ":
            i = alphabet.index(c)
            I = (i+cle)%26
            newC = alphabet[I]
        else : newC = c
        newTexte += newC
    return newTexte
```

2- ATTAQUE PAR FORCE BRUTE SUR UN TEXTE CHIFFRE EN « CESAR »:

La clé a seulement 26 valeurs possibles. Comme chaque valeur de clé donne un texte chiffré différent, il y a 26 messages chiffrés possibles pour un texte clair, selon la valeur de la clé.

Exercice 2. : Dans le même fichier *chiffrementDeCesar.py* , écrire le script d'une autre fonction nommée *forceBruteCesar()* . Cette fonction prend en argument un texte crypté « en César ». En l'exécutant, le texte décrypté pour les 26 valeurs différentes de clés, s'affiche dans la console.

Par exemple en exécutant dans la console

```
>>> forceBruteCesar("dwwdtxhc d o dxeh")
```

On obtient :

```
avec la cle 1 : cvvcswgb c n cwdg
avec la cle 2 : buubrvfa b m bvcf
avec la cle 3 : attaquez a l aube
avec la cle 4 : zsszptdy z k ztad
avec la cle 5 : yrryoscx y j yszc
avec la cle 6 : xqqxnrbw x i xryb
avec la cle 7 : wppwmqav w h wqxa
avec la cle 8 : voovlpzu v g vpwz
avec la cle 9 : unnukoyt u f uovy
avec la cle 10 : tmmtjnxs t e tnux
avec la cle 11 : sllsimwr s d smtw
avec la cle 12 : rkkrhlvq r c rlsv
avec la cle 13 : qjjqgkup q b qkru
avec la cle 14 : piipfjto p a pjqt
avec la cle 15 : ohhoeisn o z oips
avec la cle 16 : nggndhrm n y nhor
avec la cle 17 : mffmcgql m x mgnq
avec la cle 18 : leelbfpk l w lfmp
avec la cle 19 : kddkaeoj k v kelo
avec la cle 20 : jccjzdni j u jdkn
avec la cle 21 : ibbiycmh i t icjm
avec la cle 22 : haahxblg h s hbil
avec la cle 23 : gzzgwakf g r gahk
avec la cle 24 : fyyfvzje f q fzgj
avec la cle 25 : exxeuyid e p eyfi
```

En lisant les 26 textes, on constate que le seul texte qui a du sens est celui de clé égale à 3.

CORRIGE

```
def forceBruteCesar(texteChiffre) :
    for i in range(1,26):
        texteDeChiffre = chiffrementCesar(texteChiffre,i,False)
        print(f"avec la cle {i} : {texteDeChiffre}")
```

Exercice 3. : On a retrouvé dans de vieilles armoires du lycée, la phrase qui suit. Apparemment, elle est cryptée « en César » : « *jz cv evq uv tcvfgrkiv vlk vkv gclj tflik kflkv cr wrtv ul dfeuv rlirzk tyrexv*»

⇨ Décrypter ce texte et rechercher sur google son auteur, et le sens à lui donner (réponse attendue sous forme de commentaires dans le fichier *chiffrementDeCesar.py*

CORRIGE

```
>>> (executing file "chriffrementDeCesar.py")
jz cv evq uv tcvfgrkiv vlk vkv gclj tflik kflkv cr wrtv ul
dfeuv rlirzk tyrexv
si le nez de cleopatre eut ete plus court toute la face du
monde aurait change
```

⇨ Déposer ce fichier *chiffrementDeCesar_VotreNom.py* dans le répertoire Devoir sur U:/