EXEMPLES DE CODAGES

1) Généralités

Dans cette partie, on va voir comment coder un texte, écrit en majuscules.
Aux lettres A, B, …..,Z, on fait correspondre les nombres 0, 1, ……, 25.
On ne va donc manipuler que les 26 nombres de l'ensemble (E) = {0, 1, 2, ….,25}.
On dit que l'on va travailler "modulo 26".
Définition: deux entiers relatifs a et b sont **congrus modulo 26** lorsque a et b ont **le même reste** dans la division par 26. On note $a \equiv b \pmod{26}$.
Exemple $30 \equiv 4 \pmod{26}$.
Donc, dès qu'un résultat sort de (E), on le remplace par l'élément de (E) qui lui est congru (le reste de la division par 26).
En PYTHON on obtient facilement ce reste par **%26.** Exemple 30%26 = 4.
Les deux autres fonctions indispensables sont **chr** et **ord**:
chr(nombre) renvoie le caractère dont le code ASCII est nombre, ord est la fonction réciproque.
Exemple chr(65)="A" et ord("A") = 65.
Et enfin, pour parcourir les lettres d'une chaînes en PYTHON, on utilise une boucle **FOR**:
"for lettre in phrase:"…….

2) Code de César

La clef de ce code est un nombre choisi entre 1 et 25, appelé **décalage.**
Chaque lettre est alors décalée de ce nombre.
Exemple: considérons un décalage de 3. Le code sera noté cesar(3),
A numéro 0, devient donc 0 + 3 = 3, c'est-à-dire D. Il est pratique de noter: $A \rightarrow 0 \rightarrow 3 \rightarrow D$.
$Y \rightarrow 24 \rightarrow 27 \equiv 1 \pmod{26} \rightarrow B$.
Pour décoder un message, on applique à nouveau un code de César:
un message codé avec cesar(11) sera décodé en y appliquant un cesar(15).

Exercice:

Ecrire un programme qui vous demande un message (court et en majuscules), un décalage à appliquer.
En sortie il vous donnera le message codé.
Tester votre programme en appliquant par exemple un code cesar(7) à un message, puis un cesar(19) au message codé. Vous devez retrouver le message initial!
Pour vous aider voici la fonction qui code une lettre:

```
def cesar(caractere, decalage):
    if ord(caractere)>64 and ord(caractere)<91: #on ne change que les lettres majuscules
        caractere = chr(((ord(caractere)-65+decalage)%26)+65)
    return caractere
```

Le message codé est initialisé par :

```
message_code="" #chaîne vide
```

A chaque passage dans la boucle for qui décrit le message initial, il est modifié (ajout d'une lettre à chaque fois) par:

```
message_code = message_code + cesar(lettre,decalage)
```

Vous améliorerez ensuite ce programme, afin qu'il lise un fichier texte en entrée (exemple "toto.txt"), et qu'il crée en sortie un fichier codé, nommé par exemple "toto_cesar7.txt".
Utilisez votre programme pour décoder **"secret01.txt"** disponible sur le site ISN.

3) Codage affine

Pour le codage des nombres de 0 à 25 on utilise une fonction affine f, définie sur (E) par
$f(x) \equiv ax + b \pmod{26}$.
Le code sera noté affine(a,b). La clef de codage est (a,b).
Exemple: affine(3,14) $f(x) = 3x + 14$
$f(0) = 14$ donc $A \rightarrow 0 \rightarrow 14 \rightarrow O$
$f(5) = 29 \equiv 3 \pmod{26}$ donc $F \rightarrow 5 \rightarrow 3 \rightarrow D$
Remarque: b peut prendre toutes les valeurs de (E).
a ne peut prendre que les valeurs impaires autres que 13 (voir rubrique décodage).

Il est très facile d'écrire un programme de codage, avec lecture-écriture dans un fichier!
Vous reprenez votre programme "CESAR", et vous changez juste la fonction de codage!
Exercice: coder Germinal avec affine(15,7). Votre fichier de sortie doit être nommé germinal_affine(15,7).txt.

4) Décodage

Il s'agit de trouver la fonction réciproque g de la fonction f de codage.
On a y = f(x), on aimerait avoir x = g(y).
Prenons la clef (15,7) et étudions d'abord le problème dans **R**:

$$y = 15x + 7 \Leftrightarrow y - 7 = 15x \Leftrightarrow x = \frac{y-7}{15} \Leftrightarrow x = \frac{1}{15}y - \frac{7}{15}$$

Le problème est que nous ne devons utiliser que les nombres de (E)! Pas de négatifs et encore moins de fractions!!!!

Etape 1: On doit enlever le 7 du membre de droite
Dans **R**, on ajoute aux deux membres **l'opposé** de 7 qui est -7 (car 7 + (-7) = 0).
Dans (E) on ajoute **l'opposé de 7 modulo 26**, qui est 19. En effet, $7 + 19 = 26 \equiv 0 \pmod{26}$.
On obtient alors $y + 19 \equiv 15x$

Etape 2: On doit enlever le 15 devant x
Dans **R**, on multiplie les deux membres par **l'inverse** de 15 qui est $\frac{1}{15}$ (car $15 \times \frac{1}{15} = 1$).
Dans (E), on multiplie les deux membres par **l'inverse de 15 modulo 26** qui est 7.
En effet $15 \times 7 = 105 \equiv 1 \pmod{26}$.
On a donc $7(y + 19) \equiv 7(15)x$, soit $7y + 3 = x$ (car $7 \times 19 = 133 \equiv 3$ et $7 \times 15 = \equiv 1$).
On a alors la fonction affine de décodage: $y \rightarrow x = 7y + 3$.
La clef de décodage est donc (7,3).

5) Exercices

a) Décoder votre fichier "germinal_affine(15,7).txt" avec la clef (7,3). Le miracle doit avoir lieu!
b) Dresser la liste des inverses modulo 26 , s'ils existent (avec un programme python).
La sortie sera:
0 n'a pas d'inverse, 1 a pour inverse 1, 2 n'a pas d'inverse, 3 a pour inverse 9, ……
c) Ecrire un programme qui donne la clef de décodage correspondant à une clef de codage donnée:
La sortie sera: "Pour décoder affine(15,7) on utilise affine(7,3)"
Il sera utile d'utiliser un **dictionnaire** (voir python.org) des inverses:
inverse={1:1,3:9,5:21,….} crée un dictionnaire, et alors inverse[5] vaut 21.
d) Décoder le fichier **"secret02.txt"** disponible sur le site ISN. Une analyse des fréquences permet de trouver la clef en résolvant un système. Vous pouvez aussi faire une attaque en "**force brute**", c'est-à-dire tester tous les codes affines possibles en boucle avec sortie à chaque fois de la première ligne du texte pour voir si ça ressemble à quelque chose.