Chapitre 6 : Chiffrement

I Unicode

1.1 Définition

Le système Unicode permet d'attribuer à chaque caractère un code international. Il s'agit de la façon la plus utilisée pour coder les caractères. Tous les caractères peuvent être codés en utilisant le système unicode mais ici, on n'utilisera que les lettres minuscules non accentuées de l'alphabet latin. La lettre **a** a pour code Unicode **97**, la lettre **b** a pour code Unicode **98**, ...la lettre **z** a pour code Unicode **122**. On n'utilisera donc que des textes écrits en minuscules sans accent ni espace.

1.2 Commandes Python

- La commande `ord` s'applique à une chaîne de caractères constituée d'un seul caractère. Elle donne le code Unicode du caractère. Par exemple `ord('a')` renvoie `97`.
- La commande `chr` s'applique à un entier. Elle donne le caractère ayant pour code Unicode cet entier. Par exemple `chr(97)` renvoie `'a'`.

On pourra également utiliser les commandes Python donnant le reste de la division euclidienne de a par b : `a%b` et le quotient de la division euclidienne de a par b : `a\\b`.

II Chiffrement de César

Le chiffrement de César est le premier chiffrement ayant existé. La méthode est simple : il suffit de décaler toutes les lettres de l'alphabet du même nombre de lettres. Par exemple, en choisissant un décalage de 3, le a devient d, le b devient e, et ainsi de suite.

2.1 Fonction de chiffrement

```
def cesar(m,n) :
    c=''
    for i in m :
        c=c+chr((ord(i)+n-97)%26+97)
    return c
```

Dans cette fonction, on considère que `m` est une chaîne de caractères représentant le message que l'on doit coder et que `n` est un entier représentant le décalage. On initialise `c` comme étant une chaîne de caractères vide. On concatène au fur et à mesure chaque lettre codée de `m`.

Il faut rajouter `n` à la valeur de chaque code Unicode mais il ne faut pas sortir de la plage des lettres minuscules c'est-à-dire de l'intervalle $[\![97,122]\!]$, c'est pourquoi on utilise des divisions euclidiennes.

2.2 Fonction de déchiffrement

Le principe est analogue à celui de la fonction de chiffrement.

```
def dechiffre_cesar(m,n) :
    c=''
    for i in m :
        c=c+chr((ord(i)-n-97)%26+97)
    return c
```

2.3 Limites du chiffrement de César

Ce type de chiffrement est très simple à décoder car il n'y a que 26 valeurs possibles pour le décalage. Il n'est donc pas sécurisé.