TRAVAUX PRATIQUES 3

CRYTOGRAPHIE

Voici les instructions éventuellement utiles dans ce TP :

`len(L) ; len(S)`	donne la longueur de `L`; de la chaîne `S`
`L=[] ; S=''`	`L` est la liste vide; `S` est la chaîne vide
`L[i] ; S[i]`	donne la valeur de l'élément d'indice `i` dans la liste `L` ou la chaîne `S`
`M+N`	concatène (juxtapose) les listes ou les chaînes`M` et `N`
`L.append(a)`	ajoute `a` à la fin de la liste
`S+'a'`	ajoute `a` à la fin de la chaîne
`M.index(a)`	donne la première occurrence de `a` dans la liste ou la chaîne `M`
`chr(i)`	donne le caractère ASCII d'indice `i`
`ord('c')`	donne l'indice dans la table ASCII du caractère `c`
`n%p`	donne le reste de la division de `n` par `p`
`D={clé1:valeur1}`	`D` est le dictionnaire contenant une seule `valeur1` associé à une `clé1`
`D[clé]`	donne la valeur de l'élément du dictionnaire `D` associé à la `clé`
`D[clé2]=valeur2`	ajoute la `valeur2` associée à la `clé2` dans le dictionnaire `D`

1. **Création d'un alphabet**

   La table ASCII contient les lettres de l'alphabet sous forme de chaînes de caractères.

   Utiliser cette table pour définir une fonction `alphabet()`, sans argument, qui renvoie une liste contenant les lettres de l'alphabet en minuscule sous la forme de chaînes de caractères. Le premier élément de cette liste contiendra le caractère *Space*, à l'indice 32 dans la table ASCII, permettant de créer l'espace entre deux mots.

2. **Codage Jules César**

   Le principe du codage est mono-alphabétique. On cherche à coder un message. Pour cela un nombre N est choisi. Chaque lettre du message d'origine est alors remplacée par la lettre de l'alphabet qui est située N places plus loin dans l'alphabet. Si en faisant cette manipulation, on "sort" de l'alphabet, alors on reprend l'alphabet au début. Par ailleurs, l'alphabet est complété par une lettre supplémentaire qui permettra de coder les espaces entre les mots.

   Par exemple, si le nombre N choisi au départ est 5, et que le message à coder est :

   `'je vois et je comprends'`

   alors le message codé est :

   `'oje  tnxejyeojehtruwjsix'`

   (a) Ecrire une fonction `codage_cesar(message,N)`, prenant en arguments un message et un nombre et retournant le message codé en suivant le principe ci-dessus.

   Ecrire une fonction `decodage_cesar(message_code,N)` prenant en arguments un message codé par la méthode César et un nombre, qui retourne le message décodé. Tester son fonctionnement sur le message suivant codé avec $N = 10$:

   `'obyzojaobcojsmsjocjbojaozybo'`

   (b) Trouver, quitte à tester toutes les valeurs de N possibles, le message caché derrière le message crypté :

   `'buhhqwuphusguipuipsgneidwgqbbuphdciptuhphncdcnbuh'`

3. **Codage Blaise de Vigenère**

   Le principe du codage est cette fois poly-alphabétique. Un mot, appelé **clef**, est choisi au départ. Chaque lettre du message à coder sera remplacée par la lettre de l'alphabet qui est située N places plus loin dans l'alphabet suivant le principe suivant :

   - ⋆ pour la première lettre à coder, N est le numéro correspondant à la première lettre de la clef,
   - ⋆ pour la deuxième lettre à coder, N est le numéro correspondant à la deuxième lettre de la clef,
   - ⋆ pour la troisième lettre à coder, N est le numéro correspondant à la troisième lettre de la clef...

   Dès que l'on arrive à la dernière lettre de la clef, on revient au début de ce mot-clé.

   En quelques sortes, tout se passe comme si on "ajoutait" les lettres du mot-clé à celles du message à coder.

   (a) Ecrire une fonction `codage_vigenere(message,clef)` qui retourne le message codé suivant ce principe.

   Tester le fonctionnement de la fonction avec la clef `'python'` et le message suivant :

   `'je vois et je comprends'`

   (b) Ecrire une fonction `decodage_vigenere(message_code,clef)`, prenant en argument un message codé par la méthode Vigenère et la clef utilisée pour le codage, qui retourne le message décodé.

   Tester le fonctionnement de la fonction avec la clef `'informatique'` et le message :

   `'kwkbmronnqyfwff  wmnhwuzemsf  rvozxhgfbwwiw'`

4. **Nombre d'apparition des lettres dans un texte**

   Pour décoder un message codé avec le codage Vigenère, sans connaître la clé, il est nécessaire de faire une étude statistique sur le message codé, en comptant le nombre d'apparition de chaque lettre.

   Pour réaliser ce comptage, nous allons utiliser un dictionnaire.

   (a) On initialise une variable `alpha` au format dictionnaire par : `alpha={' ':0}`.

   La clé est l'espace et son effectif est initialisé à 0. Compléter la variable `alpha` pour qu'elle contienne comme clés, les lettres de l'alphabet associés aux valeurs 0 :

```
>>> alpha
{' ': 0, 'a': 0, 'b': 0, 'c': 0, 'd': 0, 'e': 0, 'f': 0, 'g': 0, 'h': 0, 'i': 0
, 'j': 0, 'k': 0, 'l': 0, 'm': 0, 'n': 0, 'o': 0, 'p': 0, 'q': 0, 'r': 0, 's':
0, 't': 0, 'u': 0, 'v': 0, 'w': 0, 'x': 0, 'y': 0, 'z': 0}
```

   (b) Soit `S` une chaîne de caractères. On suppose, pour simplifier, que `S` ne contient que des lettres minuscules et non accentuées.

   Ecrire une fonction `compte(S:str)->dict` qui prend un argument un texte au format d'une chaîne de caractères et qui renvoie la variable `alpha` qui donne le nombre d'apparition de chaque lettre dans le texte. Par exemple :

```
S='demain des l aube a l heure ou blanchit la campagne je partirai vois tu je sais que
tu m attends j irai par la foret j irai par la montagne je ne puis demeurer loin de toi
plus longtemps je marcherai les yeux fixes sur mes pensees sans rien voir au dehors sans
entendre aucun bruit seul inconnu le dos courbe les mains croisees triste et le jour pour
moi sera comme la nuit je ne regarderai ni l or du soir qui tombe ni les voiles au loin
descendant vers harfleur et quand j arriverai je mettrai sur ta tombe un bouquet de houx
vert et de bruyere en fleur'
```

```
>>> compte(S)
{' ': 116, 'a': 37, 'b': 8, 'c': 9, 'd': 15, 'e': 69, 'f': 4, 'g': 4, 'h': 6, '
i': 34, 'j': 10, 'k': 0, 'l': 21, 'm': 15, 'n': 30, 'o': 25, 'p': 9, 'q': 4, 'r
': 41, 's': 33, 't': 26, 'u': 33, 'v': 6, 'w': 0, 'x': 3, 'y': 2, 'z': 0}
```