LMI 1 - Semestre 1

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO1 – Algorithmique et Programmation 1 – Groupe 4

Fiche de TP numéro 3 - Cryptage et décryptage de messages

Dans ce TP, vous allez écrire un programme en le décomposant en plusieurs fonctions. Chaque fonction est décrite dans un des exercices en bas. Écrivez toutes les fonctions dans un seul fichier (ex. `tp04.py`).

Le chiffre de César

Le chiffre de César est une méthode de chiffrement qui consiste en substituer chaque caractère d'un message par un autre, qui se trouve à un décalage de n positions dans l'alphabet. Ce décalage est choisi par l'utilisateur. On dit que c'est la clé du chiffrement. Par exemple, si votre message est :

```
help me obi wan kenoby you are my only hope.
```

et si la clé secrète est égale à 3, alors le message chiffré est :

```
khos ph rel zdq nhqreB Brx duh pB rqoB krsh.
```

Chaque caractère du message est décalé de 3. Seul l'utilisateur connaît la clé secrète pour pouvoir déchiffrer le message.

Exercice 1 : À certains endroits dans ce programme, vous aurez besoin d'une chaîne de caractères contenant toutes les lettres de l'alphabet. Vous devez donc créer une constante `ALPHABET` qui contiendra toutes les lettres (les minuscules, les majuscules, les lettres avec accent et les chiffres). Pour simplifier votre tâche, vous pouvez (mais vous n'êtes pas obligé) utiliser le module `string` et les constantes qui y sont définies : `ascii_letters` et `digits` (pourtant, il faudra y rajouter les lettres accentuées à la main).

Votre définition de `ALPHABET` peut donc ressembler à ceci :

```
from string import ascii_letters, digits
ACCENTS = 'àâäèéêëîïòôöùûüç'
SYMBOLES = ',.!?:; \n\t&"\'@%+-/\\*_()[]{}'
ALPHABET = ascii_letters + ACCENTS + digits
```

ou bien à ceci :

```
LETTRES = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
ACCENTS = 'àâäèéêëîïòôöùûüç'
CHIFFRES = '0123456789'
SYMBOLES = ',.!?:; \n\t&"\'@%+-/\\*_()[]{}'
ALPHABET = LETTRES + ACCENTS + CHIFFRES
```

Vous pouvez récupérer la première version dans le fichier "importations-chaines-caracteres.py" disponible sur Moodle (faites un copié-collé du contenu dans votre fichier).

Exercice 2 : Spécifiez puis écrivez une fonction `input_mode` qui permet à l'utilisateur de choisir le mode d'exécution du programme. La fonction demande à l'utilisateur de choisir entre `c`, pour chiffrer un message et `d` pour déchiffrer un message. La fonction doit poser la question à l'utilisateur jusqu'à ce qu'il réponde soit par `c` soit par `d` (en minuscule ou en majuscule). La fonction retourne le choix de l'utilisateur, en minuscule. Test :

```
>>> choix = input_mode()
Voulez-vous chiffrer ou déchiffrer un message (c/d) ? y
Option invalide !
Voulez-vous chiffrer ou déchiffrer un message (c/d) ? C
>>> choix
'c'
```

Exercice 3 : Spécifiez puis écrivez une fonction `input_cle()` qui demande à l'utilisateur de donner sa clé de chiffrement. La valeur de la clé doit être comprise entre 1 et la taille de notre alphabet (c'est-à-dire, la longueur de `ALPHABET`). Test :

```
>>> choix = input_cle()
Entrez la clé de chiffrement (1-104) : 178
Clé invalide !
Entrez la clé de chiffrement (1-104) : -1
Clé invalide !
Entrez la clé de chiffrement (1-104) : 28
>>> choix
28
```

Exercice 4 : Spécifiez puis écrivez la fonction `pos(c)` qui retourne la position du caractère `c` dans l'alphabet. Si `c` n'est pas une lettre de l'alphabet, la fonction retourne -1. (Détaille : la position de la lettre dans l'alphabet de votre programme dépend de comment vous avez défini la constante `ALPHABET`). Test :

```
>>> pos('G')
32
>>> pos('a')
0
>>> pos('azerty')
-1
```

Exercice 5 : Spécifiez puis écrivez la fonction `car(n)` qui retourne le caractère à la position `n` dans l'alphabet. (Encore une fois, la position de la lettre dans votre programme dépend de comment vous avez défini la constante `ALPHABET`). Test :

```
>>> car(23)
'x'
```

Exercice 6 : Spécifiez puis écrivez la fonction `chiffre_car(c,n)` qui retourne le chiffrement du caractère `c` par la clé `n`. C'est-à-dire, qui retourne le caractère de l'alphabet qui se trouve à `n` positions de `c`. Attention, dans notre alphabet, on considère que la lettre suivant la dernière lettre (`'9'` est `'a'`).

```
>>> chiffre_car('W',23)
'3'
>>> chiffre_car('g',23)
```

```
'D'
```

Exercice 7 : Spécifiez puis écrivez la fonction `cesar(message,mode,cle)` qui chiffre ou déchiffre (selon la valeur de `mode` : `'c'` ou `'d'`) la chaîne de caractères `message` à l'aide de `cle`.

```
>>> cesar('help me obi wan kanobi you are my only hope.','c',23)
'EBIMdJBdLyFdTxKdHxKLyFdVLRdxOBdJVdLKIVdELMB{'
>>> cesar('EBIMdJBdLyFdTxKdHxKLyFdVLRdxOBdJVdLKIVdELMB{','d',23)
'help me obi wan kanobi you are my only hope.'
```

Le chiffre de Vigenère

Le chiffre de Vigenère est une méthode de chiffrement plus sûre que celle du César. Elle demande à l'utilisateur d'entrer un mot-clé de plusieurs lettres. Chaque lettre est, en effet, une clé du chiffre de César. Pour chaque caractère du message, le chiffre de Vigenère utilise une des clés données par le mot-clé. Ceci lui confère donc un peu plus de sécurité.

Regardons plus en détail. Nous considérons toujours le même message à chiffrer (`help me obi wan ...`), et cette fois la clé est un mot : `python`. On va associer à chaque lettre du message une lettre de la clé en la répétant autant de fois que nécessaire :

```
help me obi wan kanobi you are my only hope.
pyth on pyt hon python pyt hon py thon pyth
```

Nous allons maintenant considérer que ce qui nous intéresse dans la clé `python`, c'est en fait la position dans l'alphabet de chacune des lettres de la clé. Si on fait correspondre tout cela (des espaces ont été ajoutés pour simplifier la lecture) :

```
h  e  l  p    m  e     o  b  i  w a  n  ...
p  y  t  h    o  n     p  y  t  h o  n  ...
15 24 19 7 84 14 13 84 15 24 19 7 14 13 ...
```

Nous avons maintenant la clé de chiffrement pour chaque caractère du message initial : nous allons chiffrer `h` par 15, `e` par 24, `l` par 19, etc.. Avec cette méthode, une même lettre n'est pas toujours chiffrée avec le même décalage.

Exercice 8 : Spécifiez puis écrivez la fonction `input_methode` qui demande à l'utilisateur de choisir entre deux chiffrements : César (`'c'`) ou Vigenère (`'v'`).

```
>>> choix = input_methode()
Quelle méthode voulez-vous utiliser : Cesar (c) ou Vigenere (v) ? z
Option invalide !
Quelle méthode voulez-vous utiliser : Cesar (c) ou Vigenere (v) ? c
```

Exercice 9 : Spécifiez puis écrivez la fonction `vigenere(message, mode, mot_cle)` qui chiffre ou déchiffre, suivant la valeur de `mode`, `message` à l'aide de la clé `mot_cle`.

```
>>> vigenere('help me obi wan kanobi you are my only hope.','c','python')
'wCEw(zteHiw_LyG%ynCMup(LDS}hFr)KR%CAAW}oCCt}'
>>> vigenere('wCEw(zteHiw_LyG%ynCMup(LDS}hFr)KR%CAAW}oCCt}','d','python')
'help me obi wan kanobi you are my only hope.'
```

Exercice 10 : Spécifiez puis écrivez la fonction `main()` qui sera la fonction principale du programme. Elle permet à l'utilisateur de choisir une méthode de chiffrement, de donner son message et sa clé, de choisir un mode d'exécution (chiffrement ou déchiffrement), fait le travail et affiche le résultat.

```
Entrez votre message :
Vive Python !
Voulez-vous chiffrer ou déchiffrer un message (c/d) ? c
Quelle méthode voulez-vous utiliser : Cesar (c) ou Vigenere (v) ? v
Entrez le mot-clé : guido
Résultat :
```