TP noté

PTSI Lycée Eiffel

22 décembre 2017

Quelques remarques avant de commencer :

- **Les notes de cours sont autorisées (mais PAS le recours à des programmes tout faits trouvés sur Internet).**
- **Les seules fonctions Python sur les listes que vous avez le droit d'utiliser sont *len* et la méthode *append*. En particulier, PAS de *sort* ou de *sum*.**
- **Les étudiants rendront à la fin du TP un fichier Python (extension .py) contenant leurs programmes que l'enseignant récuperera directement sur une clé USB. Il est bien entendu autorisé et même conseillé de commenter les programmes écrits sur ce fichier (les commentaires en Python s'introduisent avec un #).**
- **On ne se préoccupera pas de l'efficacité des algorithmes programmés, et on aura bien entendu le droit de réutiliser certains des programmes écrits précédemment dans des questions ultérieures du sujet.**

I. Quelques algorithmes élémentaires sur les chaînes de caractères.

Nous allons nous intéresser dans ce TP noté à quelques méthodes élémentaires de cryptage (et de décryptage) de textes. Nous commencerons donc par écrire quelques fonctions Python manipulant des chaînes de caractères. On utilisera régulièrement dans ce TP le codage ASCII des caractères, et nous aurons pour ce faire besoin des deux fonctions Python suivantes :

- **chr(n)** renvoie le caractère dont le code ASCII est l'entier n. Ainsi, chr(72) renverra le caractère 'H'.
- au contraire, **ord(c)** renvoie le code ASCII d'un caractère c. Ainsi, ord('Z') renverra l'entier 90.

Les lettres minuscules de l'aphabet latin ont des codes ASCII compris entre 97 et 122 (97 pour le a, jusqu'à 122 pour le z). Les lettres majuscules de ce même alphabet ont des codes ASCII compris entre 65 et 90. Ces deux fonctions seront utiles pour certaines des questions de cette première partie (mais pas forcément les premières !).

Question 1 : Écrire une fonction Python **converliste(c)** qui prend comme argument une chaîne de caractères c et renvoie une liste contenant les caractères de c.

Exemple : converliste('Python') doit retourner la liste ['P','y','t','h','o','n'].

Question 2 : Écrire une fonction Python **converchaine(l)** qui effectue l'opération inverse (conversion en chaîne de caractères d'une liste l dont les éléments sont des caractères).

Exemple : converchaine(['N','o','ë','l',' ','!']) doit retourner la chaîne de caractères 'Noël !'.

Question 3 : Écrire une fonction Python **sommeascii(c)** qui renvoie la somme des codes ASCII des caractères de la chaîne c.

Exemple : sommeascii('Python') doit retourner la valeur 642.

Question 4 : Écrire une fonction Python **miroir(c)** qui renvoie une chaîne de caractères étant une copie retournée de la chaîne *c*.

Exemple : miroir('Python') doit retourner la chaîne 'nohtyP'.

Question 5 : Écrire une fonction Python **palindrome(c)** qui détermine si la chaîne c est un palindrome (c'est-à-dire une chaîne qui est identique à celle obtenue en la retournant). Cette fonction renverra True ou False.

Exemple : palindrome('radar') doit retourner True.

II. Cryptages par substitution.

La plupart des méthodes simples de cryptage de textes fonctionnent sur un principe de substitution : chaque lettre du texte de départ est remplacée par une autre lettre de l'alphabet. Le plus simple est de procéder à un décalage constant des rangs des lettres dans l'alphabet. Ainsi, le codage de César consiste à décaler de trois unités chacune des lettres (A devient D, B devient E, etc, sachant bien entendu que pour les trois dernières lettres on reviendra au début de l'alphabet : X devient A, Y devient B et Z devient C). **Dans toute la suite de l'énoncé, on supposera que les chaînes de caractères à coder ou à décoder sont simplement constituées d'une suite de lettres majuscules (sans ponctuation).**

Question 6 : Écrire une fonction Python **decodecesar(c)** qui renvoie le décodage d'un texte codé à l'aide de la méthode de César. On fera évidemment attention à gérer correctement les dernières lettres de l'alphabet.

Exemple : decodecesar('MRBHXAQRHO') doit renvoyer 'JOYEUXNOEL'.

Question 7 : Écrire plus généralement une fonction Python **decodedecalage(c,n)** qui décode un texte codé à l'aide d'un décalage du rang alphabétique de n unités (n étant un entier compris entre 1 et 25 ; la fonction decodedecalage(c,3) doit donc effectuer exactement la même chose que décodecesar(c)).

Question 8 : Écrire une fonction **codemiroir(c)** qui code un texte en remplaçant chacune de ses lettres par la lettre symétrique dans l'ordre alphabétique : A devient Z, B devient Y etc. Cette fonction permet aussi de décoder un texte codé suivant le même principe.

On souhaite décoder un texte codé à l'aide d'une méthode de décalage mais on ignore de combien d'unités les lettres ont été décalées. On se propose d'effectuer le décodage de la façon suivante (qui fonctionnera en pratique correctement sur des textes suffisamment longs) : on repère dans le texte codé la lettre apparaissant le plus souvent, et on effectue l'hypothèse qu'il s'agit du codage de la lettre E dans le texte initial. On décode conformément à cette hypothèse.

Question 9 : Écrire une fonction Python **nombreE(c)** déterminant le nombre de fois qu'apparait la lettre E dans la chaîne de caractères c.

Question 10 : Écrire une fonction Python **plusfrequent(c)** qui détermine le caractère apparaissant le plus de fois dans la chaîne de caractères c (toujours supposée ne contenir que des caractères alphabétiques en majuscule).

Question 11 : En déduire une fonction **decodedecalageinconnu(c)** qui effectue le décodage du texte c en utilisant la méthode décrite plus haut.

Pour se détendre avant la dernière partie, on pourra décoder le texte suivant à l'aide du programme précédent : 'FUJYJFQFQDEUBGKQDTJKTUISUDTHQITKSYUBQLUSTUIZEKUJIFQHCYBBYUHIDEKRBYUIFQICEDFUJYJIEKBYUH'.

Question 12 : Une variante du codage par décalage constant consiste à décaler les lettres du texte avec un décalage qui évolue : la première lettre est décalée d'une unité, la deuxième de deux unités etc (si le texte est suffisamment long, la 27ème lettre sera à nouveau décalée d'une unité). Écrire une fonction Python **decodedecalageprogressif(c)** permettant de décoder un texte codé par cette méthode.

III. Cryptage de Vigenere.

La méthode de cryptage de Vigenere fonctionne également sur le principe du décalage alphabétique, mais en y ajoutant le principe d'une clé de cryptage donnant les valeurs des décalages effectués sur les caractères du texte. En général, cette clé est un mot, dont les rangs alphabétiques des différentes lettres donnent les valeurs des décalages effectués. Ainsi, si on code un texte avec la clé 'PYTHON', on décalera la première lettre du texte de 16 unités (car P est la 16ème lettre de l'alphabet), la deuxième de 25 unités (correspondant au Y), etc, en revenant à un décalage de 16 unités pour la septième lettre du texte à coder, puis pour la treizième etc.

Question 13 : Écrire une fonction Python **vigenere(c,cle)** qui prend comme argument une chaîne de caractères c (toujours consitutée de caractères majuscules) et une clé (elle aussi une chaîne de caractères) et qui code le texte c à l'aide de la clé en utilisant la méthode de Vigenere.

Exemple : vigenere('CETPESTVRAIMENTTOTALEMENTTRIVIAL','PYTHON') doit renvoyer 'RCMWSFITKHWZTLMACGPJXTSAIRKPJVPJ'.

Pour décoder un texte codé à l'aide de cette méthode (sans connaitre la clé, naturellement), on utilise en général le test de Friedman pour déterminer la longueur probable de la clé : l'indice de coïncidence de deux chaînes de caractères est la proportion de caractères des deux chaînes se trouvant en position identique (ainsi, 'ABRACADABR' et 'ALEATOIRES' ont un indice de coïncidence de 20% puisque, sur dix lettres, il y en a deux identiques à la même position, les deux premiers A de chaque chaîne). Le test de Friedman consite à calculer l'indice de coïncidence du texte à décoder avec ce même texte décalé de n caractères (il ne s'agit pas ici de décalage du rang alphabétique, mais simplement d'enlever les n premiers caractères du texte et de les placer à la fin), et de considérer que l'entier n pour lequel cet indice est le plus grand est la longueur de la clé. En pratique, on supposera la clé de longueur inférieure ou égale à 10 (il faut bien limiter les valeurs de n à tester).

Question 14 : Écrire une fonction Python **indicecoincidence(c1,c2)** calculant l'indice de coïncidence de deux chaînes de caractères (de longueur éventuellement différentes, dans ce cas, on divisera le nombre de lettres communes par la longueur de la plus longue des deux chaînes).

Question 15 : Écrire une fonction Python **longueurcle(c)** qui prend comme argument un texte codé à l'aide de la méthode de Vigenere et renvoie la longueur probable de la clé de chiffrement.

Question 16 : Écrire une fonction Python **decodevigenere(c)** qui décode un texte suffisamment long codé à l'aide de la méthode de Vigenere. On cherchera d'abord la longueur de la clé, puis on utilisera le programme de la question 11 sur les lettres du texte ayant subi un même décalage (par exemple, si la clé est de longueur 6, il s'agit des lettres en position 1,7,13 etc, puis 2,8,14 etc) pour déterminer le décalage correspondant.

Si vous êtes arrivés au bout de ce TP, vous devriez être capables de décoder chez vous (car je doute un peu que vous ayiez le courage de le recopier sur Python de toute façon) le message suivant, codé à l'aide de Vigenere :

'ZZYHHBPDDMECCCDCCSSKWFFYRSBASWYCHZZGZICUMEUWZUWKTCSYCJWFJGIJVIT
KSHIRMEISMQCVYVGPPQIYTSMTHLPACSTIAPJTYISAOVTCCRIYESYPJMNUSVTOCIT
OXTOCIRIJXSLOLGJXBQWGOLPWBBLSFTGTTMFYHRMAPHBDBAZEHNGSALCOGDRMNVH
NTOVYVSHWSATKSTEOALVBXTAIYUSLJBIFGSLTBWPC'