Devoir d'informatique numéro 1

---

1 Instructions

Le DL sera rendu sous forme d'un fichier [NOM]_DL1.py contenant :

- Les fonctions demandées,
- Les lignes permettant de les tester (commentées par un #),
- les exercices étant séparés par """ EXERCICE n """ avec n le numéro de l'exercice.
- Les question qui ne sont pas du code (exercice 2 début de la question 1, etc...) sous forme de chaîne de caractères entre """.

Vous êtes vivement encouragé à me poser des questions si quelque chose ne vous semble pas clair.
Vous pouvez le rendre avant le 1er septembre.

2 Introduction

De tout temps l'être humain a cherché des moyens de protéger la confidentialité et l'authenticité de ses communications. Il y a généralement trois acteurs : un expéditeur, un destinataire, et un intercepteur potentiel sur le chemin. La cryptographie consiste à rendre le message inintelligible à toute personne autre que le destinataire. La cryptanalyse consiste à tenter de déchiffrer un message sans avoir les informations données au destinataire.
Le plus ancien document chiffré est une recette secrète de poterie qui date du XVIe siècle av. J.-C., qui a été découverte en Mésopotamie (dans l'actuelle Irak). La plus ancienne analyse cryptographique connue est une description de l'analyse fréquentielle par Al-Kindi au IXe siècle.
Le chiffrement d'un message peut être un jeu — par exemple la célèbre correspondance entre George Sand et Alfred de Musset, qui relève plutôt du message caché — ou une affaire d'État : chiffre de Marie Stuart, ou Enigma pendant la seconde guerre mondiale [1]. Plus récemment les affaires de violation de la vie privée [2] révélées par Snowden, mettent en jeu des questions de cryptographie.
Pour aller plus loin (les deux se lisent comme des romans polars) :

- Histoire des codes secrets, Simon Singh : un apperçu historique et une vulgarisation des techniques de chiffrement.
- Le déchiffrement du linéaire B, John Chadwick : comment Michael Ventris et John Chadwick ont réussi à déchiffrer les tablettes du XIVe siècle avant J.-C. à l'aide de techniques de cryptanalyse.

3 Chiffre de César

Jules César est le premier personnage connu de nos jours ayant utilisé le chiffrement par décalage. Selon Suétone, César utilisait un décalage de 3 lettres pour sa correspondance secrète, en particulier militaire.

3.1 Chiffrement

Vous avez déjà vu ce chiffrement lors du DL1 de PTSI, en janvier dernier. Le premier exercice code rapidement le chiffrement de César.

1. On fera attention à ne pas prendre pour vérité historique le dernier film sur le sujet. Cf. Wikipedia ou toute bonne encyclopédie.
2. Article 9 du code civil : « Chacun a droit au respect de sa vie privée. ». C'est un des fondements de liberté.

Exercice 1

Soit n un entier relatif. On écrit un programme qui code un mot en décalant chaque lettre de l'alphabet de n lettres. Par exemple pour $n = +3$, le décalage sera le suivant :

alphabet clair	a	b	c	d	...	x	y	z
alphabet crypté	D	E	F	G	...	A	B	C

Le mot *oralensam* devient ainsi *RUDOHQVDP*.
On ne s'occupera que des lettres, en supposant qu'elles sont toutes en minuscules dans le texte clair et toutes en majuscules dans le texte codé.

1) Définir une chaîne de caractères, `alpha`, contenant toute les lettres dans l'ordre alphabétique minuscules. On pourra au besoin utiliser `chr(n)`, où n est le code UTF8 de la lettre voulue (la commande inverse de `chr` est `ord`). Codes : « a » à « z » : 97 à 122.

2) On donne la fonction :

```
def cesar(n, texte):
    ntexte = ''
    for car in texte:
        if car in alpha:
            ntexte += alpha[(alpha.index(car) + n) % 26].upper()
        else:
            ntexte += car
    return ntexte
```

Comment décoder un message que vous venez de coder ? On pourra utiliser la commande `lower` qui permet de passer en minuscules : `'BLABLAblo'.lower()` retourne `'blablablo'`.

3.2 Cryptanalyse

Le but de cette partie est de décrypter un message chiffré par décalage — on parle de « casser » le code.

Exercice 2 (Attaque par force brute)

1) On sait que le texte suivant (présent sur ma page web) est chiffré par décalage :

« WODRYNOLBEDOPYBMOYXDOCDODYEDOCVOCYCCSLSVSDOC »

Combien de chiffrements par décalage différents existe-t-il ?

Proposer une méthode de résolution, et casser le code. Il est vivement conseillé de faire une boucle (rappel de syntaxe de print : `print(n, texte[i:j])`).

2) Expliquer ce que fait le code suivant (et changer le nom de la fonction) :

```
def mystere(nom_fichier):
    with open(nom_fichier, 'r') as fichier_python:
        resultat = fichier_python.read()
    return resultat
```

Cf. http://docs.python.org/3.4/tutorial/inputoutput.html

3) Casser le code du texte 1 reçu par courrier électronique et donner l'auteur.

4 Autres chiffrement par substitution monoalphabétique

4.1 Chiffrement

Exercice 3

1) La fonction de numpy `np.random.randint(a,b)` tire un entier au hasard (selon une loi uniforme) dans `range(a,b)` (i.e. $[\![a, b-1]\!]$).

À l'aide de cette fonction, coder une fonction `creer_permut` qui prend en argument un entier n et qui retourne une liste de n entiers tirés au hasard dans $[\![0, n-1]\!]$, sans répétition. On pourra utiliser une boucle `while`.

2) Combien y a-t-il de telles listes ?

Exercice 4

Vous vous donnez une permutation des lettres de l'alphabet (par exemple codée par une liste `permut` des entiers entre 0 et 25 dans un ordre arbitraire : il faudra envoyer la `i`-ème lettre sur la `permut[i]`-ème).

1) Créer une liste `permut`

2) Écrire une fonction `crypter_permut(per, texte)` qui prend en argument cette liste et le texte à crypter, et qui retourne le texte crypté par cette méthode.

3) Comment décoder le texte en connaissant la permutation ?

4) Peut-on casser le chiffrement par force brute, par une méthode analogue à celle utilisée pour casser le code de César ? En supposant qu'il suffise d'une seconde pour tester chaque possibilité, combien faut-il de jours (ou d'années) pour tester toutes les possibilités (vous avez le droit d'utiliser autant de gens que vous voulez, dans la limite des stocks disponibles) ?

4.2 Cryptanalyse

Exercice 5 (analyse fréquentielle)

Les fréquences d'apparition des différentes lettres ne sont pas les mêmes, dans un texte en français suffisament long. Par exemple, le « e » est la lettre (en moyenne [3]) la plus fréquente.
Si le texte a été crypté par permutation monoalphabétique, la lettre la plus fréquente sera probablement celle qui correspond au « e ».

1) Écrire une fonction `frequences(texte)` qui retourne une liste des fréquences des 26 lettres dans le texte `texte`, le tester sur le texte 2 reçu par mail.

2) Affichage graphique : pour afficher sous forme de barres la liste des fréquences calculées à la question précédente, on utilise `matplotlib.pyplot`. Exemple :

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(range(26))
y = [np.random.randint(100) for i in range(26)]
y_0 = np.array([(i-13)**2 for i in range(26)])
w = .3
plt.bar(x, y, w, 0)
plt.bar(x+w, y_0, w, 0, color='red')
plt.xticks(x, list(alpha))  #Légende sur l'axe des x

plt.show()
```

3) On donne le tableau de fréquences suivant (pour le français) :

`[9.42, 1.02, 2.64, 3.39, 15.87, 0.95, 1.04, 0.77, 8.41, 0.89, 0.00, 5.34, 3.24, 7.15, 5.14, 2.86, 1.06, 6.46, 7.90, 7.26, 6.24, 2.15, 0.00, 0.30, 0.24, 0.32]`

Évitez de le retaper : récupérez-le sur ma page web.

Affichez sur une même figure les deux listes de fréquences. Conjecturer quelle lettre correspond au « e ».

4) (bonus) Les lettres n'ont pas toutes les mêmes voisines, avec les mêmes fréquences. Par exemple les lettres doublées (cf bigramme) sont, par fréquence décroissante : e, s, l, t, n, m, r, p, f, c (et d'autres plus rares).

Écrire une fonction qui classe par fréquence décroissante les lettres doubles du texte 2.

5) (bonus) Écrire un programme qui

3. En moyenne : pas dans La disparition, par exemple...

- affiche les histogrammes de la question 3 et le texte crypté,
- demande à l'utilisateur (`input('blabla')`) de rentrer une lettre chiffrée et sa traduction en clair.
- effectue le remplacement dans le texte (avec les vérifications ad hoc).
- recommence au premier point.

Grâce à ce programme (et aux lettres doublées), craquer le code du texte 2.

5 Le chiffre de Vigenère

5.1 Chiffrement

Nous venons de voir que les chiffrements par substitution sont vulnérables à l'analyse fréquentielle.
Le chiffrement de Vigenère fonctionne sur le principe suivant : on dispose d'une clé qui est une chaîne de caractères (en général un mot), et on décale la première lettre du texte d'un rang correspondant à celui de la première lettre de la clé dans l'alphabet et ainsi de suite. Par exemple, si la clé est « python », la première lettre sera décalée de 15 (p est la 15ème lettre de l'alphabet, en partant de 0), le deuxième de 24 etc. On revient au début de la clé si le texte à crypter est plus long que la clé).
Exemple :

texte clair	u	n	e	x	e	m	p	l	e	d	e	c	h	i	f	f	r	e	m	e	n	t
clé	p	y	t	h	o	n	p	y	t	h	o	n	p	y	t	h	o	n	p	y	t	h
décalage	15	24	19	7	14	13	15	24	19	7	14	13	15	24	19	7	14	13	15	24	19	7
texte crypté	J	L	X	E	S	Z	E	J	X	K	S	P	W	G	Y	M	F	R	B	C	G	A

Ce chiffrement, décrit au XVIe siècle, a été considéré comme incassable de sa popularisation au XVIIe siècle jusqu'au milieu du XIXe siècle.

Exercice 6

1) Écrire un programme Python prenant comme argument une clé `cle` et le texte à crypter `texte` (deux chaînes de caractères) et effectuant le chiffrement.

2) Comment déchiffrer le texte en connaissant la clé ? (écrire la fonction)

5.2 Cryptanalyse

Grâce à la clé, une même lettre peut être codée par différentes lettres dans le texte chiffré (cf le « e » dans l'exemple précédent). Donc ce code n'est pas vulnérable à une analyse fréquentielle simple.

Exercice 7

1) En fonction de la longueur N de la clé, donner le nombre de tests à effectuer par une méthode « force brute ». Que constate-t-on ?

2) On suppose désormais que l'on connaît la longueur N de la clé. Vous testerez vos programme avec le texte 3, et une clé de longueur $N = 5$.

a) Cette question utilise le découpage en tranche avec un pas (que l'on testera en console) :

`s[i:j:k]` slice of `s` from `i` to `j` with step `k`

Écrire une fonction qui découpe un texte en N textes, tous chiffrés à l'aide d'un chiffre de César.

b) À l'aide de la fonction affichant des histogrammes (exercice 5), détecter le décalage pour chacun des N textes (sans avoir à utiliser de force brute). En déduire la clé.

c) Finir de décoder le texte 3 (les espaces et la ponctuation manquent, mais avec la première phrase vous devriez retrouver le texte sur le web...).

Mais il reste à connaître la longueur de la clé pour casser le code dans le cas général.

3) (bonus) Sur l'exemple suivant, il y a des répétitions : « une petite petition pour illustrer ou reside le probleme »

texte clair	unepet	itepet	itionp	ourill	ustrer	ouresi	delepr	obleme
clé	python	python	python	python	python	python	python	python
texte crypté	JLXWSG	XRXWSG	XRBVBC	DSKPZY	JQMYSE	DSKLGV	SCELDE	DZELAR

Ces répétitions apparaissent à une distance un multiple de la longueur de la clé, et permettent donc de trouver la longueur de la clé lorsque le texte est assez long.

a) Écrire une fonction qui compte les motifs de trois lettres ou plus qui apparaissent deux fois au moins dans le texte. Elle retournera un tableau avec le nombre d'occurrences de chaque motifs, classé par ordre décroissant.

b) Écrire une fonction qui prend les 10 motifs les plus fréquents et affiche pour chacun le ppcm (on cherchera sur le web la commande pour le ppcm) des écarts entre chaque occurrence.

c) Conclure.