Faculté des Sciences
Aix*Marseille Université

Licence Sciences et humanités, 3e année

Deuxième TP

vendredi 25 novembre 2016
T. Coulbois

Des chaines de caractères et de la cryptographie

D'abord il vous faut un environnement de programmation. Nous avons travaillé la semaine passée avec des environnements en ligne : le Jupyter proposé par `sagemath.org` Cette semaine, nous avons de vrais ordinateurs et nous pouvons donc utiliser un vrai IDE (environnement de développement intégré).

Lancez Geany depuis les applications dans le menu de KDE (le K en bas à droite).

1 Compter les lettres dans une chaine de caractère

On se donne une chaine de caractère :

`message="BVBFMNOHXNCIEXBMEIQMWOZBHEOGBIFYKIFROCYIFYZIKOHWWMIMBWNIVB"`

Écrire un programme qui compte le nombre de `B` dans cette chaine.

Écrire un programme qui renvoie une liste du nombre des occurences pour chaque lettre de l'alphabet.

Pour cela il vous faudra manipuler les caractères de l'alphabet latin et leur code ASCII. Les fonctions `chr()` et `ord()` sont faites pour convertir entre caractères et codes ASCII.

```
for i in range(65,91):
        print(chr(i), end=' ')
```

```
for c in "BVBFMNOHXNCIEX":
        print(ord(c), end=' ')
```

2 Modifier une chaine de caractère

Attention en python les chaines de caractères sont immuables : on peut accéder à ses composants ou à ses sous-chaines mais on ne peut pas les modifier.

```
print(message[3])
print(message[2:7])
```

```
message[2]='Z'
```

Écrire un programme qui insère un espace tous les 5 caractères de la chaine `message` .

Écrire un programme qui renvoie la chaine de caractère `message` ci-dessus en remplaçant les `M` par des `m` et les `B` par des `z`.

3 Déchiffrons

Le texte ci-dessous est un texte chiffré ou chaque lettre a été remplacée par une autre (par exemple tous les `B` ont été remplacés par un `K`).

```
GLPKKNVAHRLBALYGRNANLCLPOOFVWFVAHCNPKKNHVARFPLCNZV
YNGNKPZLONHHNOAFMOPRFANNMFVATVNWNYNZAVHHNMLHTVNKNY
HNRSKNGNHZIFHNHTVFYYNGFPOMLHULPANNHOPYGPCPHPSKNNOT
VNGNHWNVYNHUPKKNHTVPRLYTVNYOGNANHMNZOLKLCPNPKKNHHN
UVHHNYOOFVOGVYZFVMLAANONNHMLAGNHHZAVMVKNHTVLYGPKHL
BPOGNMKLPHPAHMKVHONYOLONVAHTVNGNHLVONAMLAGNHHVHVYF
ZOFBNYLPAN
```

Pour le déchiffrer, on dispose d'un indice précieux : le texte est écrit en français (et on a supprimé les accents, les espaces et la poncutation). Et en français les lettres ont des fréquences assez variées :

E	A	S	I	N	T	R	L	U	O	D	C	P
17,26	8,40	8,08	7,34	7,13	7,07	6,55	6,01	5,74	5,26	4,18	3,03	3,01
M	V	G	F	B	Q	H	X	J	Y	Z	K	W
2,96	1,32	1,27	1,12	1,06	0,99	0,92	0,45	0,31	0,30	0,12	0,05	0,04

Les bigrammes les plus fréquents sont dans l'ordre décroissant :

`ES DE LE EN RE NT ON ER TE EL AN SE ET LA AI IT ME OU EM IE`

En vous servant des programmes écrits précédemments, essayez de deviner par quelle lettre sont codées le `E`, le `A`, le `S` etc. Puis décodez ce message.

Pour effectuer la translitération, il sera peut-être utile d'utiliser des dictionnaires.

4 Avec des fichiers

Il serait plus agréable de lire les données (les textes chiffrés) dans des fichiers.

```
fichier = open('test.txt','w')
fichier.write('bonjour')
fichier.close()
```

```
fichier = open('test.txt','r')
ligne = fichier.readline()
fichier.close()
print(ligne)
```

Décriptez par l'analyse des fréquences le fichier `http://www.i2m.univ-amu.fr/~coulbois/2017/llc/bonsens-chiffre.txt`.

5 Au secours !

Il y a beaucoup de documentation et d'aide en ligne.

Connectez-vous sur `docs.python.org` et en particulier le tutoriel : `docs.python.org/3/tutorial`.

D'abord, sur Jupyter ou `cloud.sagemath.org`, en ligne de commande taper la touche de tabulation 

complète la commande par les commandes connues de Python. Tous les environnement de programmation ont une complétion automatique des commandes. Ensuite, toujours sur Jupyter, en ligne de commande, éxécuter une commande suivi du point d'interrogation affiche la documentation de cette commande : `max?`.

```
Docstring:
max(iterable[, key=func]) -> value
max(a, b, c, ...[, key=func]) -> value

With a single iterable argument, return its largest item.
With two or more arguments, return the largest argument.
Type:      builtin_function_or_method
```

Les commandes que nous avons rencontrées jusqu'ici :

Opérations arithmétiques : `+`, `-`, `*`, `/`, `//` (la division euclidienne), `%` (le reste de la division euclidienne), `**`, `^` (les deux pour les puissances).

Notations des nombres et des chaines de caractères :