Chiffrage

1 Échauffement

1.1 Construction de listes par compréhension

Q1 : Définir les trois listes suivantes à l'aide d'une construction par compréhension :

```
Liste1 = [0., 0.02, 0.04, ... 3.]
```

```
Liste2 = [1, 2, 5, 10, 17, 26, ... 10001]=[0**2+1 1**2+1 ...]
```

```
Liste3 = [sin(1), sin(2), sin(5), sin(10), sin(17), ... sin(10001)]
```

Q2 : Expliquer l'évaluation de l'expression suivante :

```
[chr(65+i) for i in range(26)]
```

1.2 Parcours de listes

Q3 : Écrire une fonction qui renvoie la somme de tous les termes d'une liste de nombres notée `Ln`.

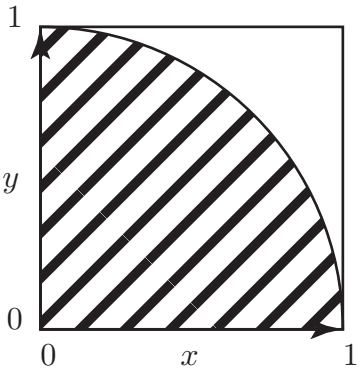
Q4 : Écrire une fonction qui vérifie si le caractère "e" ou le caractère "E" sont présents dans une chaine de caractères donnée.

1.3 Méthodes de Monté-Carlo

Les méthodes de Monté-Carlo forment une catégorie de méthode de résolution où, plutôt qu'utiliser une méthode analytique pour trouver un solution, on utlilise une méthode probabilistique.

Un exemple simpliste consiste à déterminer la surface d'un lac [source wikipedia] : Soit une zone carrée dont les côtés sont de longueur unitaire. Au sein de cette aire se trouve un lac dont la superficie est inconnue. Grâce aux mesures des côtés de la zone, on connaît l'aire du carré. Pour trouver l'aire du lac, on demande à une armée de tirer X coups de canon de manière aléatoire sur cette zone. On compte ensuite le nombre N de boulets qui sont restés sur le terrain ; on peut ainsi déterminer le nombre de boulets qui sont tombés dans le lac : $X - N$. Il suffit ensuite d'établir un rapport entre les valeurs :

$$\frac{\text{superficie}_{\text{ terrain}}}{\text{superficie}_{\text{ lac}}} = \frac{X}{X-N}$$



On considère la figure ci-contre. On cherche l'aire hachurée. Evidemment, la solution est $\frac{\pi}{4}$. On utilisera la fonction `random` du module `random`, qui génère et renvoie un nombre aléatoire y entre 0 et 1 inclus (utiliser l'instruction `help(random)` pour afficher l'aide associée à cette fonction).

1. Créer une liste de deux nombres aléatoires compris entre 0 et 1. Ils représentent les coordonnées en x et y du point.
2. Pour savoir si le point est dans l'aire recherchée, il suffit de vérifier $x^2 + y^2 < 1$
3. Générer cent points aléatoires. Compter ceux qui sont qui sont dans l'aire recherchée. En déduire une estimation de cette aire. En déduire une estimation de π.

2 Chiffre de César

2.1 Présentation

En cryptographie, le chiffrement par décalage, aussi connu comme le chiffre de César, est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes (ce qui explique le nom « chiffre de César »).

Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une permutation circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue la clé du chiffrement qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message. Dans le cas de l'alphabet latin, le chiffre de César n'a que 26 clés possibles (y compris la clé nulle, qui ne modifie pas le texte).

2.2 Structure informatique

On utilisera des textes encodés en majuscules, sans ponctuation et sans accents. On n'écrira par ailleurs pas les espaces dans les textes codés et décodés.

Les objets manipulés seront d'une part des chaînes de caractères, dont on rappelle qu'elles ne sont pas **mutables** ; et d'autre part des entiers qui représenteront un code ASCII d'un caractère.

Q1 : Ecrire une fonction qui prend en entrée un caractère (majuscule) et un décalage et renvoie un caractère.

Cette fonction pourra par exemple récupérer le code ASCII du caractère (à l'aide de la fonction ord(caractere)) ; appliquera le décalage souhaité, calculera le code ASCII

du caractère modulo 26 dans l'intervalle [65,90] et retransformera le code ASCII en caractère (s'inspirer du I en cas de problème syntaxique).
Elle pourra alternativement travailler sur l'indice d'un caractère dans la chaîne `s="AB..Z"`.

2.3 Cryptage d'un texte

On suppose que l'on veut chiffrer un texte qui respecte les conventions présentées. Ainsi "Je programme tous les jours pour m'améliorer" sera écrit sous la forme : "JEPROGRAMMETOUSLESJOURSPOURMAMELIORER"

Q2 : Écrire une fonction de chiffrage d'une chaîne de caractères. Elle renverra une chaîne de caractères et aura pour en-tête :

```
def ChiffrageCesar(chaine_a_coder,decalage):
```

Q3 : Chiffrer le texte suivant avec un décalage de 9 : "JEPROGRAMMETOUSLES-JOURSPOURMAMELIORER".
On doit trouver : "SNYAXPAJVVNCXDBUNBSXDABYXDAVJVNURXANA".

2.4 Décryptage d'un texte

Q4 : Écrire une fonction qui déchiffre une chaîne de caractères qui respecte les conventions présentées

On réfléchira dans un premier temps à l'en-tête de la fonction. Puis on réfléchira tout court...

Q5 : Décoder le texte précédemment encodé avec le même décalage à des fins de vérifications.
Décoder NYHCL avec un décalage de 7 et de 20
On doit trouver : "TENIR" puis "GRAVE".

3 Analyse fréquentielle du chiffre de César

Le chiffre de César n'est malheureusement pas très utile pour réellement chiffrer des données. Même lorsque la clé n'est pas connue, tester à la main les 26 possibilités (25 en réalité...) permet à un opérateur humain d'en déduire dans la plupart des cas le texte décodé.

Nous allons étudier une autre méthode ici, qui s'appuie sur une détection automatique de la clé la plus probable de déchiffrement.

3.1 Principe

Les lettres ont des fréquences moyennes d'apparition en français qui, bien que dépendant de la langue utilisée, des corpus de textes étudiés, sont approximativement connues. Voici un tableau résumant celles-ci :

Lettre	f	Lettre	f	Lettre	f	Lettre	f
a	0.0768	h	0.0064	o	0.0534	v	0.0127
b	0.0080	i	0.0723	p	0.0324	w	0.0000
c	0.0332	j	0.0019	q	0.0134	x	0.0054
d	0.0360	k	0.0000	r	0.0681	y	0.0021
e	0.1776	l	0.0589	s	0.0823	z	0.0007
f	0.0106	m	0.0272	t	0.0730		
g	0.0110	n	0.0761	u	0.0605		

Notre objectif sera de casser un code de César en regardant les 26 décalages possibles, et en cherchant, à l'aide de la minimisation d'une quantité, quel est le décalage le plus probable. Nous allons minimiser la somme des écarts des fréquences au carré.

Plus exactement, si $f_t(c)$ désigne la fréquence d'apparition théorique moyenne d'un caractère c dans la langue française, et si $f(c)$ désigne sa fréquence d'apparition dans un texte donné, l'écart des carrés est défini par :

$$S(e) = \sum_{c \in \mathcal{A}} (f_t(c) - f(c))^2$$

où $\mathcal{A}$ désigne l'alphabet.

En notant f_d les fréquences obtenues dans un texte par un décalage de d caractères dans le déchiffrage de César, nous allons chercher d qui minimise :

$$S_d(e) = \sum_{c \in \mathcal{A}} (f_t(c) - f_d(c))^2$$

Q1 : Écrire en pseudo-code les étapes de l'algorithme que nous allons implémenter.

3.2 Fonctions à écrire

Les fréquences f_t(c) de chaque lettre sont stockées dans le dictionnaire ou la liste ci-dessous (à copier-coller).

```
Dic_Freq={'A': 0.0768, 'H': 0.0064, 'O': 0.0534, 'V': 0.0127, 'B': 0.008,
 'I': 0.0723, 'P': 0.0324, 'W': 0.0, 'C': 0.0332, 'J': 0.0019, 'Q': 0.0134,
'X': 0.0054,'D': 0.036, 'K': 0.0, 'R': 0.0681, 'Y': 0.0021, 'E': 0.1776,
 'L': 0.0589, 'S': 0.0823, 'Z': 0.0007, 'F': 0.0106, 'M': 0.0272,
'T': 0.073, 'G': 0.011, 'N': 0.0761, 'U': 0.0605}

List_Freq=[0.0768, 0.008, 0.0332, 0.036, 0.1776, 0.0106, 0.011, 0.0064, 0.0723,
0.0019, 0.0, 0.0589,0.0272, 0.0761, 0.0534, 0.0324, 0.0134, 0.0681, 0.0823,
 0.073, 0.0605, 0.0127, 0.0, 0.0054, 0.0021, 0.0007]
```

Q2 : Pour un texte donné (encodé selon nos conventions), calculer les fréquences d'apparition des lettres du texte. On renverra le résultat sous la forme d'une liste (de 26 nombres). On pourra commencer par initialiser une liste contenant 26 zéros que l'on créera par compréhension

Q3 : En utilisant une approche similaire, recalculer ces fréquences d'apparition, en renvoyant cette fois le résultat sous la forme d'un dictionnaire dont les clés seront les lettres de l'alphabet. (cette question n'est pas utile pour la résolution de l'exercice ; elle nous permet de réviser l'usage des dictionnaires).

Q4 : Pour une liste de fréquences donnée, écrire une fonction qui calcule la quantité S.

Q5 : Pour un texte et un décalage donné, écrire une fonction qui calcule la quantité S. On se servira des deux fonctions précédentes.

Q6 : Pour une chaîne de caractères, dont le décalage est inconnu, écrire une fonction qui renvoie une liste contenant les quantités S(d) pour les différents décalages d possibles (d prend les valeurs entière de 0 à 25)

Q7 : Écrire une fonction qui cherche le minimum d'une liste de nombres et renvoie la position de ce minimum.

Q8 : Écrire une fonction qui renvoie la chaîne de caractères la plus probable dans le cas d'un chiffrage de César de décalage inconnu à l'aide de la méthode de l'analyse fréquentielle

3.3 Test de validation

Q9 : Décoder le texte suivant en utilisant les fonctions précédentes. Quel était le décalage employé ?

"SJRVNVJWPNAMNBVJAAXWBUNBXRAJDLXRWMDOND"

3.4 Test plus difficile depuis un fichier externe

La Disparition est un roman de Georges Perec écrit avec la contrainte très particulière de ne pas contenir la lettre e. Un fichier externe ladisparitioncodee.txt est disponible sur le site.

Q10 : Après avoir ouvert ce fichier en lecture, et avoir stocké la chaîne de caractères contenue dans ce fichier, essayez de décoder le texte par analyse fréquentielle.

Rappels de syntaxe :

`monfichier=open(Chemin,'r')` *Ouvre un fichier en lecture*

`s=monfichier.read()`: *La chaîne de caractères contenue dans le fichier est stockée dans s*

`monfichier.close()` *Ferme le fichier*