TP9

Partie 1 — Cryptographie

Ex. 1 — Codage de César

Le Codage de César est un procédé de cryptographie permettant d'encoder en message en chiffrant chaque lettre du message en une autre lettre, décalée dans l'ordre alphabétique par un nombre de caractères connu à l'avance. On appelle ce nombre de caractères la clé d'encryption du message.

L'objectif de cet exercice est de créer des fonctions d'encodage et de décodage suivant la méthode du codage de César.

Dans le détail, le codage de César est basé sur la substitution des caractères suivant leur position dans l'ordre alphabétique.

Prenons comme exemple la clé `n=3` . Avec cette clé, chaque lettre non accentuée devient après encodage la lettre située trois positions après dans l'ordre alphabétique.

Ainsi `'a'` devient `'d'`, `'f'` (la sixième lettre de l'alphabet) devient `'i'` (la neuvième lettre) et `'x'` (24ème) devient `'a'` (car `1 = (24 + 3) % 26`).

Le codage fonctionne également avec une clé négative. Il faut alors reculer dans l'ordre alphabétique, tout en conservant le côté cyclique du codage. `'D'` devient `'Y'` avec la clé de cryptage `-5`.

Pour encoder un message complet (de plusieurs caractères) avec une certaine clé, - chaque lettre non accentué minuscule devient la lettre minuscule décalée par la clé - chaque lettre non accentué majuscule devient la lettre majuscule décalée par la clé - tous les autres caractères (lettre accentuées, ponctuation, espaces) restent identiques

Question 1 : **Écrivez** le code de la fonction `code_char(c,cle)`, qui **renvoie** le caractère correspondant à l'encodage de César du caractère `c` par la clé `cle`.

Pour ce faire, il est pratique d'utiliser les fonctions `ord` et `chr`, permettant de transformer un caractère en un entier le représentant (et respectivement un entier en le caractère correspondant). Dans la console, essayez les commandes suivantes et déduisez-en comment encoder un caractère.

```
ord('a'), ord('d'), ord('z')
ord('a') - ord('a'), ord('d') - ord('a'), ord('z') - ord('a')
ord('A'), ord('Z')
ord(' '), ord('.')
chr(ord('G'))
chr(ord('a') + 5)
```

Remarque: une des propriétés importantes de ord est d'avoir les 26 lettres minuscules représentés par des entiers consécutifs. C'est également le cas pour les lettres majuscules.

Question 2 : **Écrivez** la fonction `encode(s,cle)` qui **renvoie** la chaîne de caractère `s` encodée avec la clé `cle`. Pour ce faire, créez une autre chaîne de caractère initialement vide et ajoutez les caractères encodés un par un.

Question 3 : **Écrivez** la fonction `decode(s,cle)` qui décode une chaîne `s` encodée avec la clé `cle`, et **renvoie** le message décodé.

Question 4 : À l'aide de la fonction `decode`, trouvez quel est la clé décodant le message `"Sbrl, ql zbpz avu wèyl."`. Pour cela, vous pouvez par exemple tester toutes les clés possibles, et lire le résultat de chaque tentative de décodage. Seul l'un d'entre eux fait sens.

Correction

```
#Q1
def code_char(c,cle):
    n = ord(c)
    if(n >= ord('a') and n <= ord('z')):
        return chr(ord('a') + ((n - ord('a') + cle) % 26))
    elif(n >= ord('A') and n <= ord('Z')):
        return chr(ord('A') + ((n - ord('A') + cle) % 26))
    else:
        return c


def print_code_char(c,cle):
    print("\'" + c + "\' est codé en \'" + code_char(c,cle) + "\' avec la clé " + str(cle))

print("Question 1:")
print_code_char('d',5)    #doit transformer 'd' en 'i'
print_code_char('i',-15)  #doit transformer 'i' en 't'
print_code_char('P',20)   #doit transformer 'P' en 'J'
print_code_char(' ',5)    #doit transformer ' ' en ' '

#Q2

def encode(s,cle):
    s2 = ""
    for i in range(len(s)):
        s2 += code_char(s[i],cle)
```

```
    return s2

print()
print("Question 2:")
print(encode("La vie est un long fleuve tranquille.",3))
print(encode("La vie est un long fleuve tranquille.",11))
print(encode("La vie est un long fleuve tranquille.",-7))

#Q3
def decode(s,cle):
    return encode(s,-cle)

print()
print("Question 3:")
print(decode("Wl gtp pde fy wzyr qwpfgp eclybftwwp.",-15))

#Q4
message_a_decoder = "Sbrl, ql zbpz avu wèyl."
cle_decodage = -7

print()
print("Question 4:")
print(encode(message_a_decoder,cle_decodage))
```

Ex. 2 : Codage Vigenère

Le Codage de Vigenère est un autre encodage substituant chaque caractère d'un message par un autre afin de l'encrypter.

Plutôt que de toujours utiliser la même méthode d'encodage des caractères, il est basé sur l'utilisation d'une clé sous la forme d'une chaîne de caractères (avec uniquement des lettres minuscules).

Chaque lettre du message se voit associer un caractère de la chaîne `cle` suivant le nombre de lettres (non accentuées) qui le précède: - la première lettre du message se voit associer la première lettre de la clé - la seconde lettre du message se voit associer la seconde lettre de la clé - la troisième lettre du message se voit associer la troisième lettre de la clé - … - Dès que l'on a fini d'associer toutes les lettres de la clé, on recommence à associer les lettres du début de la clé de manière cyclique.

Ensuite, on effectue un encodage de chaque caractère `c` avec un décalage dans l'ordre alphabétique correspondant au même décalage transformant `'a'` en la lettre de la clé associée à `c`.

Ainsi, avec la clé `"arbre"`, on obtient les lettres associés, et les encodages suivants:

```
Message       : "Ceci est un message a encoder."
Cles associées: "arbr ear br earbrea r brearbr "
Message codé  : "Cvdz isk ve qejtrke r fegoufi."
```

- Le `'C'` est associé à `'a'` et donc est décalé de `0`, pour donner `'C'`.
- Le `'e'` est associé à `'r'` et donc est décalé de `14`, pour donner `'v'`.
- Le `'c'` est associé à `'b'` et donc est décalé de `1`, pour donner `'d'`.

Question 1 : **Copiez** (ou réécrivez) la fonction `code_char` utilisée pour le codage de César dans le premier exercice.

Question 2 : **Écrivez** la fonction d'encodage de Vigenère, `encode(s,cle)`, qui encode `s` avec la chaîne de caractère `cle`. Attention à bien différencier les lettres (faisant avancer dans la clé) des autres caractères.

Question 3 : **Écrivez** la fonction de décodage de Vigenère, `encode(s,cle)`, qui décode une chaîne codée `s` avec la chaîne de caractère `cle`. Il faut alors faire l'opération inverse de l'encodage pour chaque caractère.

Correction

```
#Q1 -- reprendre code_char du codage de César.

def code_char(c,cle):
    n = ord(c)
    if(n >= ord('a') and n <= ord('z')):
        return chr(ord('a') + ((n - ord('a') + cle) % 26))
    elif(n >= ord('A') and n <= ord('Z')):
        return chr(ord('A') + ((n - ord('A') + cle) % 26))
    else:
        return c

#Q2
def encode(s,cle):
    s2 = ""
    j = 0
    for i in range(len(s)):
        if((s[i] >= 'a' and s[i] <= 'z') or (s[i] >= 'A' and s[i] <= 'Z')):
            s2 += code_char(s[i],ord(cle[j]) - ord('a'))
            j = (j + 1) % len(cle)
        else:
            s2 += s[i]
```

```
    return s2

print()
print("Question 2:")
print(encode("Ceci est un message a encoder.","arbre"))  #l'encodage attendu est "Cvdz isk v
print(encode("Peu lui importe de quoi demain sera fait","petitfrere")) #l'encodage attendu e


#Q3
def decode(s,cle):
    s2 = ""
    j = 0
    for i in range(len(s)):
        if((s[i] >= 'a' and s[i] <= 'z') or (s[i] >= 'A' and s[i] <= 'Z')):
            s2 += code_char(s[i],ord('a') - ord(cle[j]))
            j = (j + 1) % len(cle)
        else:
            s2 += s[i]
    return s2

print()
print("Question 3:")
print(decode("Diepe dorr n'owg naj k vrnnvr","kenyarkana"))
```

Ex. 3 : Codage spartiate

Le codage Spartiate est une méthode d'encryption par transposition. Contrairement à César ou Vigenère, les caractères sont tous conservés, mais déplacés les uns par rapport aux autres rendant le message irreconnaissable.

Ce codage nécessite une clé de transposition `n`, qui est un nombre entier positif. À partir de cette clé, un message à coder est découpé en `n` segments de même longueurs (on ajoute des espace à la fin du message autant que nécessaire). Ensuite on construit le message codé de la façon suivante:

- Les `n` premières caractères du message codé sont:
  - Le premier caractère du segment n°`1`
  - Le premier caractère du segment n°`2`
  - …
  - Le premier caractère du segment n°`n`
- Les `n` caractères suivants sont:
  - Le second caractère du segment n°`1`
  - Le second caractère du segment n°`2`
  - …

  - Le second caractère du segment n°`n`
- Ainsi de suite jusqu'à avoir utilisé tous les caractères des segments.

Une façon de représenter ce codage est d'observer la matrice de `n` lignes et `ceil(len(s)/n)` colonnes, avec ceil la fonction donnant la partie entière supérieure d'un nombre réel. On écrit ensuite le message, un caractère par case. Le message codé est alors la suite des caractères écrits colonne par colonne.

Prenons l'exemple de `s="La vie c'est comme une boîte de chocolats, on ne sait jamais sur quoi on va tomber."`, avec `n=6`:

```
La␣vie␣c'est␣c
omme␣une␣boîte
␣de␣chocolats,
␣on␣ne␣sait␣ja
mais␣sur␣quoi␣
on␣va␣tomber.␣
```

Le message codé est alors `Lo  moamdoan meni ve  svi cn aeuhes  no utcecsro' oa mebliqbsoatuetît or tsji.ce,a`.

Question 1 : **Écrivez** la fonction `encode(s,n)`, qui renvoie `s` encodée par la méthode spartiate avec la clé `n`.

Vous pouvez utiliser la fonction `ceil` de la bibliothèque math, qui donne la partie entière arrondie au supérieur de son argument (si jamais vous avez besoin de la partie inférieur, la fonction `floor` existe.), notamment pour calculer à l'avance le nombre de caractères par ligne de la matrice.

Question 2 : **Écrivez** la fonction `decode(s,n)`, qui renvoie la chaîne décodée, en supposant que le code utilisé est la méthode spartiate avec la clé `n`.

Question 3 : **Trouvez** la clé permettant de décoder le message `"L n eq.ae f u  sllti vtoerl i nual eugvne "`, si on vous assure qu'un message en français a été encodé avec le codage spartiate.

Correction

```
from math import *

#Q1

def encode(s,n):
    s2 = ""
    m = ceil(len(s) / n)
    for i in range(m):
        for j in range(n):
            if i + j * m < len(s):
```

```
                s2 += s[i + j * m]
            else:
                s2 += " "
    return s2

print("Question 1:")
print(encode("La vie c'est comme une boîte de chocolats, on ne sait jamais sur quoi on va to
## Résultat attendu: "Lo  moamdoan meni ve  svi cn aeuhes  no utcecsro' oa mebliqbsoatuetît


#Q2
def decode(s,n):
    return encode(s,ceil(len(s)/n))

print()
print("Question 2:")
print(decode("Càq s' up emea?so r titl   ue ",5))
print(decode("L, snru s  ekjutp.eeioè ",6))

#Q3
message_a_decoder = "L n eq.ae f u  sllti vtoerl i nual eugvne "
cle_de_decalage = 7

print()
print("Question 3:")
print(decode(message_a_decoder,cle_de_decalage))
```

Partie 2 — Autoreproduction

Ex. 4 — Les fichiers

Fichiers

L'informatique pourrait être définie comme la science du stockage, de la transmission, et du traitement de l'information. Cette information peut provenir de différentes sources.

Pour un programme, l'information peut provenir par exemple : * de l'utilisateur (cf. l'instruction `input()` en python) ; * d'un sous-programme (cf. l'instruction `return` d'une fonction python) * du réseau ; * d'un fichier…

Au cours de cet exercice nous allons apprendre à lire un fichier, puis écrire dans un fichier, après avoir manipulé son contenu.

Lecture

Pour lire un fichier appelé `nomdufichier.type` il faut

1. Ouvrir le fichier en lecture: `f=open("nomdufichier.type","r")`.
2. Lire le contenu du fichier: `contenu=f.read()`.
3. Fermer le fichier: `f.close()`.

La variable `f` contient un "handler" vers le fichier ouvert. C'est-à-dire toutes les informations dont votre programme a besoin pour manipuler fichier. On peut y penser comme étant le "petit nom" de `nomdufichier.type`, en qui concerne le programme.

Question 1 : Tenter d'ouvrir un fichier inexistant. Qu'observez vous? Ouvrir le fichier `message.txt` et afficher son contenu.

Manipulation

En 2060, l'Académie Française décide de suivre l'exemple de la "Real academia española". Comme c'est le cas depuis longtemps pour l'espagnol, l'orthographe de la langue française sera donc simplifié au maximum. L'objectif, à terme, est d'obtenir un orthographe phonétique. Toutefois, afin de ne pas choquer la frange la plus conservatrice de la population, seul un premier volet de la réforme est introduit.

Voici la liste officielle des transformations à appliquer: * "er", "ez" -> "é" * "eau", "au", "ô" -> o * "ill" -> "y" * "ge" -> "j"

Vous êtes chargé de moderniser l'orthographe des "Fleurs du mal" de Charles Baudelaire. Pour vous aider, sacher que si `contenu` est une chaine de caractère, alors `contenu.replace("truc","machin")` en est une autre, obtenue en replaçant dans `contenu` toutes les occurences de "truc" par "machin".

Question 2 : Au lieu d'ouvrir `message.txt` comme à la question précédente, ouvrez `fleursdumal.txt`. Appliquez une série d'opérations de la forme `contenu=contenu.replace(...)` afin d'accomplir la transformation. Affichez le résultat.

Ecriture

Pour écrire dans un fichier appelé `nomdufichier.type` il faut

1. Ouvrir le fichier en lecture: `g=open("unautrefichier.type","r")`.
2. Ecrire une chaîne de caractère dans le fichier: `g.write(contenu)`.
3. Fermer le fichier: `g.close()`.

Question 3 : Ecrire dans un fichier `fleursdumal-moderne.txt` le résultat de la manipulation précédente. Lire par exemple le poème "La géante" pour vérifier le résultat.

Correction

```
#Q1
f=open("message.txt","r")
contenu=f.read()
f.close()
print(contenu)

#Q2
f=open("fleursdumal.txt","r")
contenu=f.read()
f.close()
contenu=contenu.replace("er ","é ")
contenu=contenu.replace("ez ","é ")
contenu=contenu.replace("eau","o")
contenu=contenu.replace("au","o")
contenu=contenu.replace("ô","o")
contenu=contenu.replace("ill","y")
contenu=contenu.replace("ge","je")
print(contenu)

#Q3
g=open("fleursdumal-moderne.txt","w")
g.write(contenu)
g.close()
```

Ex. 5 — Commandes systèmes

De tous les programmes qui s'exécutent sur votre ordinateur, le plus important est le système d'exploitation ("Operating System") : MacOs, Linux, Windows. Techniquement, c'est en fait un ensemble de programmes qui agissent de concert pour fournir toutes les fonctionnalités de bases : gestion du microprocesseur, de la mémoire, des fichiers, des périphériques, etc.

Grâce aux modules `os` et `sys`, vos programmes `python` peuvent intéragir avec le système d'exploitation, en lui envoyant des commandes et en observant ce qu'il renvoie. Par exemple l'instruction

```
system("commande")
```

demande au système d'exploitation d'exécuter la commande en question.

Question 1 : Demander au système d'exploitation d'exécuter la commande `pwd`.

`pwd` signifie "print working directory". La commande affiche l'endroit, dans le système de fichier entier de l'ordinateur, dans lequel vous êtes en train de travailler actuellement.

Question 2 : Afficher la valeur `argv[0]`. Que contient-elle?

`argv[0]` contient le nom entier du programme courant. Si le programme est appelé avec un argument, `argv[1]` contient cet argument. C'est la manière dont le système d'exploitation informe le programme des conditions dans lesquelles il a été exécuté.

Question 3 : Demander au système d'exécuter un autre programme `./myscript.py`. Qu'observez-vous?

Sous Linux du moins, vous observerez que ce fichier n'a pas le droit d'être exécuté.

Question 4 : Demander au système de vous donner les droits en exécution pour `./myscript.py`, avec la commande `chmod u+x myscript.py`. Demander au système d'exécuter le programme `./myscript.py`. Qu'observez-vous? Etudier le code de `myscript.py`. A quoi sert la première ligne?

Sous Linux du moins, la première ligne de `myscript.py` sert à spécifier que le texte contenu dans `myscript.py` n'est autre qu'un programme `python`.

Correction

```
import os, sys



#Q1
os.system("pwd")

#Q2
print(sys.argv[0])

#Q3
os.system("./myscript.py")

#Q4
os.system("chmod u+x myscript.py")
os.system("./myscript.py")
```

Ex. 6 — Vie

Dixit wikipedia. En biologie, une entité est traditionnellement considérée comme vivante si elle présente les activités suivantes, au moins une fois durant son existence : 1. Développement ou croissance : l'entité grandit ou mûrit jusqu'au moment où elle devient capable de se reproduire ; 2. Métabolisme : consommation, transformation et stockage d'énergie ou de masse; croissance en absorbant de l'énergie ou des nutriments présents dans son environnement ou en réorganisant sa masse, par production d'énergie, de travail et rejet de déchets ; 3. Motilité externe (locomotion) ou interne (circulation) ; 4. Reproduction : pouvoir créer de façon autonome d'autres entités similaires à soi-même. 5. Réponse à des stimuli : pouvoir détecter des propriétés de son environnement et d'agir de façon adaptée.

Question 1 : En combinant les réponses des exercices précédents, écrire quatre lignes de code qui fassent que votre programme: 1. Met son propore nom dans une variable `me`. 2. Lit le contenu de son propre fichier 3. Affiche ce contenu

Au danemark il fut courant que Christian appelle son fils Christiansen, Peder appelle son fils Pedersen, etc.

Question 2 : Créer une variable `son` dont le contenu est la chaine caractère `me`, où `".py"` a été remplacé par `"sen.py"`.

Question 3 : En combinant les réponses des exercices précédents, écrire trois lignes de code qui fassent que le contenu de votre programme s'écrive dans un fichier ayant comme nom celui retenu par la variable `son`.

Question 4 : Demander au système de donner les droits en exécution au fichier créé à la question précédente.

Question 5 : Demander au système d'exécuter le fichier créé à la question précédente. Avant de lancer l'exécution de votre programme, ouvrez le répertoire courant dans un explorateur de fichier, et apprêtez vous à interrompre le processus avec `Crtl+C`. Lancez. Qu'observez-vous dans l'explorateur de fichier? Pourquoi? A quoi sert la ligne `time.sleep(1)` qui précède?

Correction

```
#!/usr/bin/python

import sys,os,time

#Q1
me=sys.argv[0]
f=open(me,"r")
contenu=f.read()
```

```
f.close()
print(contenu)

#Q2
son=me.replace(".py","sen.py")
print(son)

#Q3
g=open(son,"w")
g.write(contenu)
g.close()

#Q4
os.system("chmod u+x "+son)


time.sleep(1)

#Q5
os.system(son)
```