TP 16 : Gestion des fichiers

Exercice 1 *Calculs sur des données contenues dans un fichier.*

Vous commencerez par copier le fichier **`rfxcmd.txt`** *qui vous a été envoyé dans le même répertoire que celui où vous écrivez vos programmes* PYTHON.

On a enregistré dans un fichier nommé **`rfxcmd.txt`** des mesures de température effectuées à différents instant par plusieurs sondes. On donne ci dessous un extrait de ce fichier :

```
2,1392837237,FE02,20.7
4,1392837278,FE02,20.7
6,1392837319,FE02,20.7
9,1392837360,FE02,20.7
10,1392837364,B904,20.5
12,1392837401,FE02,20.7
13,1392837407,B904,21
15,1392837442,FE02,20.7
16,1392837450,B904,21.2
17,1392837465,9C01,20.6
         ...
```

Chaque ligne de ce fichier est constituée d'un identifiant (nombre entier), d'une date (également sous la forme d'un nombre entier), de la référence de la sonde ayant effectué la mesure et enfin de la mesure de température proprement dite. Ces informations sont séparées par des virgules. Pour répondre aux questions suivantes, on pourra utiliser la fonction **`split`** sur les chaines de caractères :

```
>>> s = "1,2,3,abc"
>>> L = s.split(',')
>>> print(L)
['1', '2', '3', 'abc']
>>> s = "1 2 3 ab,c"
>>> L = s.split(' ')
>>> print(L)
['1', '2', '3', 'ab,c']
```

(a) Écrire les commandes permettant d'ouvrir le fichier, parcourir toutes les lignes et enregistrer dans une liste **`T`** toutes les mesures de température ainsi obtenues.

(b) Écrire les commandes permettant d'afficher la moyenne des températures contenues dans le fichier.

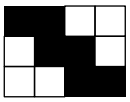
(c) Écrire les commandes permettant d'afficher les moyenne des températures contenues dans le fichier et concernant uniquement la sonde **`B904`**.

(d) Écrire les commandes permettant d'enregistrer dans un fichier **`rfxmoy.txt`** les moyennes des sondes **`B904`** et **`FE02`** sous la forme :

`B904`, *moyenne de la sonde B904*
`FE02`, *moyenne de la sonde FE02*

Exercice 2 *Écrire une image au format PBM*. On considère une image rectangulaire constituée uniquement de pixels (points) noirs et blancs. La largeur de l'image est de ℓ pixels et et hauteur est de n pixels. Cette image sera représentée par une liste PYTHON et on désire l'enregistrer dans un fichier en utilisant le format PBM. Par exemple :

L'image :



(largeur 4 et hauteur 3).

La liste PYTHON :

```
L = [[1,1,0,0],
     [0,1,1,0],
     [0,0,1,1]]
```

Le fichier PBM :

```
P1
4 3
1 1 0 0
0 1 1 0
0 0 1 1
```

Un fichier au format PBM est construit de la manière suivante : la première ligne est la chaîne de caractères `"P1"`, la deuxième donne la largeur et la hauteur de l'image (séparées par un espace) et chaque ligne suivante correspond à une ligne de l'image en donnant les couleurs de pixels (1 pour noir et 0 pour blanc), séparées par des espaces également.
Écrire une fonction **`ecrire_pbm(L,nom)`** qui construit le fichier nommé *nom* à partir de la liste L en respectant le format PBM.

Exercice 3 *Compter les mots*.

(a) Écrire une fonction **`nombre_mots(s)`** qui compte combien de mots il y a dans la chaîne de caractères *s*. *Note : pour simplifier cette première question, on supposera que la chaîne de caractère s est composée uniquement de mots séparés par un espace simple, par exemple*

$$s = \texttt{"Une phrase simple"}$$

(b) Utiliser cette fonction pour compter combien de mots comporte le premier chapitre de *L'Île Mystérieuse* de Jules Verne (fichier **`IleMysterieuseChap1.txt`**).

(c) On veut reprendre la fonction **`nombre_mots`** de la première question pour qu'elle fonctionne dans un cas plus général. On définit pour cela la liste :

```
Separateurs = [" ",",",";",".","!","?",":","(",")","-","\n"]
```

(`"\n"` est le caractère « retour à la ligne »). Si *s* est une chaîne de caractères, on appelle mot de *s* toute chaîne non vide, contenue dans *s* et délimitée par deux éléments de la liste ci-dessus. Modifier la fonction **`nombre_mots`** pour qu'elle compte les mots de *s* avec cette nouvelle définition.

Corrections

Ex 1. (a)

```
nomfichier = 'rfxcmd.txt'
```

```
T = []
f = open(nomfichier,'r')
# On parcourt toutes les lignes du fichier f :
for s in f.readlines():
    # On décompose la chaîne de caractères
    L = s.split(",")
    T.append(float(L[3]))
f.close()
print(T[:10])
```

```
[20.7, 20.7, 20.7, 20.7, 20.5, 20.7, 21.0, 20.7, 21.2, 20.6]
```

(b)

```
def moyenne(L):
    return sum(L)/len(L)

print(moyenne(T))
```

```
18.61489361702128
```

(c)

```
T = []
f = open(nomfichier,'r')
# On parcourt toutes les lignes du fichier f :
for s in f.readlines():
    # On décompose la chaîne de caractères
    L = s.split(",")
    if L[2]=="B904":
        T.append(float(L[3]))
f.close()
print(moyenne(T))
```

```
20.81612903225807
```

(d)

```
T1 = [] # Températures de B904
T2 = [] # Températures de FE02
f = open(nomfichier,'r')
# On parcourt toutes les lignes du fichier f :
for s in f.readlines():
    # On décompose la chaîne de caractères
    L = s.split(",")
    if L[2]=="B904":
        T1.append(float(L[3]))
    elif L[2]=="FE02":
        T2.append(float(L[3]))
f.close()
```

```
nomfichiermoy = 'rfxmoy.txt'
```

```
f = open(nomfichiermoy,'w')
f.write("B904,"+str(moyenne(T1))+"\n")
f.write("FE02,"+str(moyenne(T2))+"\n")
f.close()
```

Ex 2.

```
def ecrire_pbm(L,nom):
    f = open(nom,'w')
    f.write("P1\n")
    # Largeur de l'image = len(L[0])
    # Hauteur de l'image = len(L)
    f.write(str(len(L[0]))+" "+str(len(L))+"\n")
    for ligne in L:
        for pixel in ligne :
            f.write(str(pixel)+" ")
        f.write("\n")
    f.close()
ecrire_pbm([[1,1,0,0],[0,1,1,0],[0,0,1,1]],'test.pbm')
```

Le contenu du fichier :

```
P1
4 3
1 1 0 0
0 1 1 0
0 0 1 1
```

Ex 3. (a) Vu les conditions données dans l'énoncé, le nombre de mots dans une chaîne de caractères est égal au nombre d'espaces plus 1.

```
def nombre_mots(s):
     n = 0
     for i in range(len(s)):
         if s[i]==" ":
             n = n+1
     return n+1
print(nombre_mots("Une phrase simple"))
```

```
3
```

(b) Il suffit maintenant de prendre le fichier ligne à ligne et totaliser le nombre de mots de chaque ligne :

```
FICHIER = "IleMysterieuseChap1.txt"
```

```
f = open(FICHIER,'r')
N = 0
s = f.readline()
while s!="":
    N = N+nombre_mots(s)
    s = f.readline()
f.close()
print(N)
```

```
1996
```

(c) On va modifier la fonction **nombre_mots** pour que son fonctionnement soit plus réaliste. On utilise une chaîne de caractères intermédiaire notée **ch_int**. On parcourt la chaîne *s* et, si le caractère rencontré n'est pas un séparateur, on l'ajoute à **ch_int**. Si le caractère rencontré est un séparateur, on regarde le contenu de la chaîne **ch_int** qui contient les caractères rencontrés depuis le dernier séparateur. Si cette chaîne est vide, c'est que l'on se trouve entre deux séparateurs adjacents et il ne faut pas augmenter le nombre de mots. Si elle n'est pas vide, alors on augmente le nombre de mots et on vide la chaine **ch_int**.

```
def nombre_mots(s):
    Separateurs = [" ",",",";",".","!","?",":","(",")","-","\n"]
    ch_int = ""
    n = 0
    for i in range(len(s)):
        if s[i] in Separateurs:
            if ch_int!="":
                n = n+1
            ch_int = ""
        else:
            ch_int = ch_int+s[i]
    return n
print(nombre_mots(" Une chaine, sans doute... plus complexe !\n"))
```

```
6
```

```
f = open(FICHIER,'r')
N = 0
s = f.readline()
while s!="":
    N = N+nombre_mots(s)
    s = f.readline()
f.close()
print(N)
```

```
1967
```