Chap. 7 : Lecture et écriture dans des fichiers

I Fichiers textes.

En Python, l'accès à un fichier est assuré par l'intermédiaire d'un objet particulier appelé objet-fichier et créé à l'aide de l'instruction ***open(nom,mode)*** comme ceci :

```
>>> objet=open('exemple.txt','r')
>>>
```

Une fois l'objet-fichier créé, il est doté de méthodes spécifiques, qui permettent notamment de lire et d'écrire dans le fichier, suivant le mode d'ouverture qui a été précisé dans le paramètre mode. Ce paramètre de type « chaîne de caractères » peut prendre plusieurs valeurs :

- **r** : ouverture pour lecture seul
- **w** : ouverture pour écriture : si le fichier n'existe pas, il est créé, sinon son contenu est écrasé
- **a** : ouverture pour ajout : si le fichier n'existe pas, il est créé, sinon l'écriture s'effectue à la suite du contenu déjà existant ;
- **+** : ouverture pour lecture et écriture.

Une fois la lecture ou l'écriture dans le fichier terminée, on referme le fichier à l'aide de l'instruction ***close***.

Pour écrire dans un fichier, on peut utiliser l'instruction ***write*** qui écrit une chaîne de caractères en les concaténant.

Pour lire l'intégralité du contenu d'un fichier, on dispose de l'instruction ***read*** :

```
# Création d'un nouveau fichier et écriture
nom = 'mon_fichier.txt'
fichier = open(nom, 'w')
for i in range(1, 4):
    fichier.write('Ceci est la ligne {}\n'.format(i))
fichier.close()

# Lecture
fichier = open(nom, 'r')
liste = fichier.read()
print(liste)
fichier.close()

# Ajout en fin de fichier
fichier = open(nom, 'a')
for i in range(4, 6):
    fichier.write('Ceci est la ligne {}\n'.format(i))
fichier.close()

# Lecture
fichier = open(nom, 'r')
liste = fichier.read()
print(liste)
fichier.close()
```

Voici le résultat :

```
>>>
Ceci est la ligne 1
Ceci est la ligne 2
Ceci est la ligne 3

Ceci est la ligne 1
Ceci est la ligne 2
Ceci est la ligne 3
Ceci est la ligne 4
Ceci est la ligne 5
```

Si l'on souhaite récupérer une liste contenant l'ensemble des lignes du fichier, on utilise l'instruction ***readlines*** :

```
# Création d'un nouveau fichier et écriture
nom = 'mon_fichier.txt'
fichier = open(nom, 'w')
for i in range(1, 4):
    fichier.write('Ceci est la ligne {}\n'.format(i))
fichier.close()


# Variante pour la lecture
fichier = open(nom, 'r')
liste = fichier.readlines()
print(liste)
fichier.close()
```

Voici le résultat :

```
>>>
['Ceci est la ligne 1\n', 'Ceci est la ligne 2\n', 'Ceci est la ligne 3\n']
>>>
```

Lorsque l'on manipule un fichier, il est recommandé d'utiliser le mot-clé ***with*** qui a l'avantage de fermer proprement le fichier une fois le bloc d'instructions correspondant exécuté. De plus, il permet une syntaxe plus concise :

```
# Création d'un nouveau fichier et écriture
with open('mon_fichier','w') as f:
   for i in range(1, 4):
        f.write('Ceci est la ligne {}\n'.format(i))
f.closed


# Variante pour la lecture
with open('mon_fichier','r') as f:
    liste = f.readlines()
print(liste)
f.closed
```

Exercice 1 : Ecrire une fonction qui renvoie le nombre de lignes d'un fichier.

Exercice 2 : Ecrire une fonction qui renvoie la liste de tous les mots d'un fichier. On utilisera la méthode split qui s'applique à une chaîne de caractères et dont la documentation officielle se trouve ci-dessous :

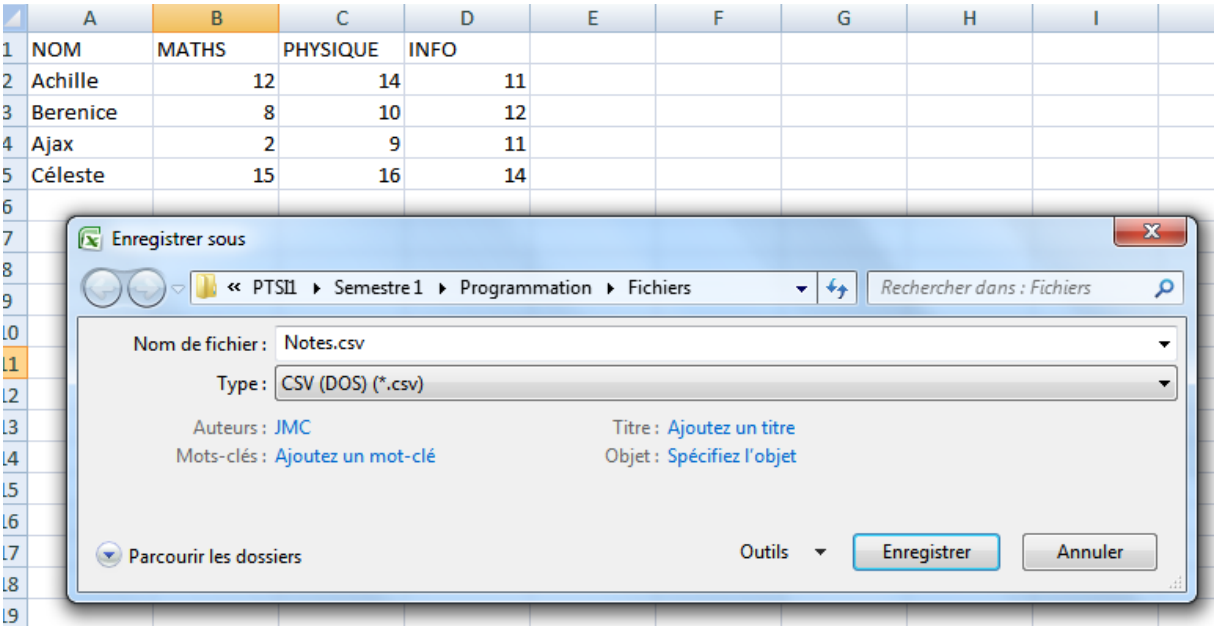
> str.split() Return a list of the words in the string, using *sep* as the delimiter string.
> For example, ' 1 2   3 '.split() returns ['1', '2', '3'].

II Fichiers CSV.

Le format « Comma-separated values »(CSV) est un format informatique ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules. C'est le format le plus couramment utilisé pour importer ou exporter des données d'une feuille de calcul d'un tableur.

Un fichier CSV est un fichier texte dans lequel chaque ligne correspond à une rangée du tableau, les cellules d'une même rangée sont séparées par une virgule.

A partir d'un tableur, il suffit de sauvegarder une feuille de calcul en précisant le format CSV pour fabriquer un tel fichier.



Le fichier CSV correspondant sera :

```
NOM;MATHS;PHYSIQUE;INFO
Achille;12;14;11
Berenice;8;10;12
Ajax;2;9;11
Céleste;15;16;14
```

Pour ouvrir un fichier CSV avec Python, on importe le module CSV et on utilise la syntaxe suivante :

```
import csv
with open('Notes.csv',newline='') as f:
    lire=csv.reader(f)
    for ligne in lire:
        print(ligne)
```

On obtient :

```
>>>
['NOM;MATHS;PHYSIQUE;INFO']
['Achille;12;14;11']
['Berenice;8;10;12']
['Ajax;2;9;11']
['Céleste;15;16;14']
>>>
```

Pour écrire dans un fichier, on utilise l'instruction open avec l'option **'w'** pour écrire ou **'a'** pour ajouter du contenu.

```
table = [['Noms', 'maths', 'physique', 'chimie'],
['Jérome', '12', '14', '15'],
['Tintin', '17', '11', '9'],
['Milou', '15', '15', '16'],
['Alcibiade', '11', '13', '12'],
['Aspasie', '19', '15', '18']]

import csv
with open('test.csv', 'w', newline='') as f:
    ecrire = csv.writer(f)
    for ligne in table:
        ecrire.writerow(ligne)

with open('test.csv', 'a', newline='') as f:
    ecrire = csv.writer(f)
    ecrire.writerow(['Romeo', '14', '17', '17'])


with open('test.csv',newline='') as f:
    lire=csv.reader(f)
    for ligne in lire:
        print(ligne)
```

Attention !!! : avec l'option **'w'**, la méthode ***writer*** écrase le fichier s'il existe déjà.

Exercice 3 : Ecrire un script qui ajoute au fichier test.csv les moyennes de chaque élève.