TP5 - Fichiers

Dans ce TP, nous allons dans un premier temps apprendre à manipuler des chaînes de caractères, et dans un deuxième temps apprendre à lire, à créer ou à modifier un fichier texte et à utiliser les données du fichier.

I Les chaînes de caractères

I.1 Manipulation d'une chaîne de caractères

On considère `ch` et `txt` deux chaînes de caractères.

- ch[i] est la chaîne de caractères composée du caractère de `ch` d'indice `i`.

  Exemple : si `ch` vaut `"Bonjour"`, `ch[2]` vaut `'n'`.
- `len(ch)` renvoie le nombre de caractères de la chaîne de caractères `ch`.

  Exemple : si `ch` vaut `"Bonjour"`, `len(ch)` renvoie 7.
- `'a' in ch` renvoie `True` si le caractère `'a'` est dans `ch` et `False` sinon.

  Exemple : si `ch` vaut `"Bonjour"`, `'o' in ch` renvoie `True` et `'b' in ch` renvoie `False`.
- `ch + txt` est la chaîne de caractères résultant de la concaténation des deux chaînes de caractères `ch` et `txt`.

  Exemple : si `ch` vaut `"Bonjour"` et `txt` vaut `" !!!"`, `ch + txt` vaut `"Bonjour !!!"`.
- `ch.strip()` renvoie la chaîne de caractères `ch` dans laquelle les caractères d'espacement sont retirés en début et en fin de la chaîne. Les caractères d'espacement sont le caractère blanc `" "`, le caractère de tabulation `"t"`, le caractère de retour à la ligne `"n"`.

  Exemple : si `ch` vaut `"Bonjour \n"`, `ch.strip()` renvoie `"Bonjour"`.
- `ch.split(c)` renvoie une liste de chaîne de caractères telle que `ch` soit « scindé » suivant la chaîne de caractères `c`

  Exemple : si `ch` vaut `"Bonjour"`, `ch.split("o")` renvoie `["B", "nj", "ur"]`.

Q1. Compléter :

```
>>> phrase = "Bonjour tu vas bien ? \n"
>>> phrase_sans_RC = ...
>>> liste = ...
>>> liste
["Bonjour", "tu", "vas", "bien", "?"]
```

I.2 Parcourir une chaîne de caractères

Tout comme une liste, on peut parcourir la chaîne de caractères de deux façons :

- suivant les caractères :

```
ch = "Hier"
for elt in ch:
  print(elt)
```

- suivant les indices :

```
ch = "Hier"
for k in range(len(ch)):
  print(ch[k])
```

I.3 Exercices

Q2. Ecrire une fonction `occurence(mot : str, c : str) -> int` qui a pour paramètre une chaîne de caractères `mot` et un caractère `c` et qui renvoie le nombre de `c` qu'il y a dans `mot`.

Q3. Ecrire une fonction `nb_de_mots(phrase : str) -> int` qui a pour paramètre une chaîne de caractères `phrase` correspondant à une phrase ne comportant pas d'apostrophe et se terminant par un point et qui renvoie le nombre de mots qu'il y a dans la phrase. *On utilisera obligatoirement la méthode `split`.*

II Le module os

Dans la suite du TP, nous allons utiliser un fichier `planetes.csv` que l'on mettra dans un répertoire.

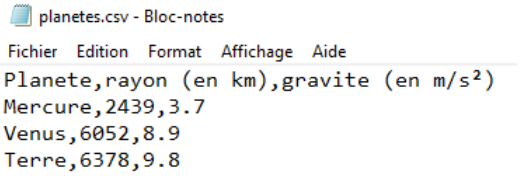
Un fichier `CSV` *(comma separated values)* est un fichier texte dont chaque ligne du texte correspond à une ligne du tableau et dont un caractère spécial (comme la virgule ou le point-virgule) correspond à la séparation entre les colonnes.

`planetes.csv`

Ouverture avec LibreOfficeCalc

	A	B	C
1	Planete	rayon (en km)	gravite (en m/s²)
2	Mercure	2439	3.7
3	Venus	6052	8.9
4	Terre	6378	9.8

Ouverture avec Bloc-notes



La fonction `getcwd()` du module `os` *(operating system)* permet d'obtenir le répertoire de travail courant *(current work directory)*.

```
>>> import os
>>> os.getcwd()
"C:\Users\Utilisateur"
```

On peut changer de répertoire courant en utilisant la fonction `chdir(chemin)` qui prend en argument une chaîne de caractères correspondant au chemin du répertoire dans lequel on souhaite travailler.

Par exemple, si `planetes.csv` est dans `"P:\python\TP4"`, on écrit dans la console :

```
>>> os.chdir("P:\python\TP4")
```

Q4. Faire en sorte que le répertoire de travail soit celui où se trouve le fichier `planetes.csv`.

III Fichier texte

III.1 Ouverture du fichier texte en mode lecture

L'ouverture d'un fichier se fait grâce à la fonction `open(nom, mode)`.

La fonction `open` a pour paramètres :

- `nom` de type `str` correspondant au nom du fichier ;
- `mode` de type `str` correspondant au mode d'accès au fichier.
  `mode` peut prendre comme argument `"r"` (read) pour la lecture du fichier, `"w"` (write) pour l'écriture du fichier et `"a"` (append) pour l'ajout de texte en fin du fichier.

La fonction `open` renvoie un objet de type `File`.

Exemple :

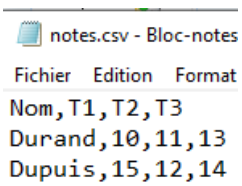
```
f = open("planetes.csv", "r")
```

`f` est un objet de type `File`.

Remarque : Une fois qu'on a fini de traiter le fichier, il faut penser à le refermer avec la méthode `.close()`.

```
f = open("planetes.csv", "r")
# Traitement
...
f.close()
```

On donne le fichier `notes.csv` suivant :



Les méthodes de la classe `File` permettant de lire le fichier sont :

- `.read()` renvoie un `str` correspondant à tout le fichier.

  Exemple :

```
>>> f = open("notes.csv", "r")
>>> ch = f.read()
>>> ch
"Nom,T1,T2,T3\nDurand,10,11,13\nDupuis,15,12,14"
```

- `.readline()` renvoie un `str` correspondant à une ligne du fichier.

  Exemple :

```
>>> f = open("notes.csv", "r")
>>> ch1 = f.readline()
>>> ch2 = f.readline()
>>> ch2
"Durand,10,11,13\n"
```

- `.readlines()` renvoie une `list` de toutes les lignes du fichier.

  Exemple :

```
>>> f = open("notes.csv", "r")
>>> lst = f.readlines()
>>> lst
["Nom,T1,T2,T3\n", "Durand,10,11,13\n", "Dupuis,15,12,14"]
```

Remarque : : L'objet `f` de type `File` est **itérable**, ce qui signifie qu'on peut parcourir le fichier ligne par ligne à l'aide d'une boucle `for`.

```
f = open("notes.csv", "r")
ch = ""
for elt in f:
    ch = ch + elt
f.close()
```

```
>>> ch
"Nom,T1,T2,T3\nDurand,10,11,13\nDupuis,15,12,14"
```

Q5.

1. Ouvrir le fichier `planetes.csv`.
2. Lire la première ligne.
3. Récupérer la gravité dans une liste `gravite`.
4. Fermer le fichier.
5. Calculer la gravite moyenne (à l'aide d'une boucle `for`) et l'afficher.

III.2 Ouverture du fichier texte en mode écriture

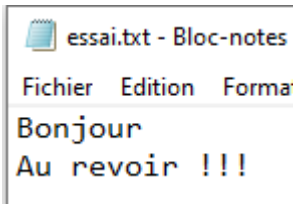
L'ouverture d'un fichier en mode `"w"` crée un nouveau fichier *(s'il existe déjà un fichier du même nom, celui-ci sera effacé).*

L'écriture commence au début du fichier et il faut refermer le fichier pour que celui-ci soit enregistré.

Pour écrire dans le fichier, on utilise la méthode `.write(ch)` qui a pour paramètre une chaîne de caractères `ch` que l'on souhaite écrire dans le fichier.

Exemple :

```
f = open("essai.txt", "w")
f.write("Bonjour \nAu revoir")
f.write(" !!!")
f.close()
```



Q6. On souhaite rajouter une colonne Annee, à la fin du fichier `notes.csv` dans laquelle il y aura la moyenne des 3 trimestres.

1. Ouvrir le fichier `notes.csv` en mode lecture.
2. On initialise une variable `ch` à une chaîne de caractères vide, que l'on modifiera au fur et à mesure de la lecture du fichier. Ainsi, à la fin de la lecture, `ch` contiendra tout le texte du fichier modifié.
3. Lire la première ligne du fichier. La rajouter dans `ch`, puis rajouter `"Annee"`
4. Lire la suite du fichier ligne par ligne avec une boucle `for`
   a) A chaque fois, rajouter la ligne dans `ch`, puis rajouter la moyenne.
5. Fermer le fichier.
6. Ouvrir le fichier `notes.csv` en mode écriture et écrire le contenu de la variable `ch`.
7. Fermer le fichier `notes.csv`

III.3 Exercices

Q7. Faire un histogramme.
On considère le fichier `histogramme.csv` qu'on ouvrira.
Il contient 4 colonnes : Nom, T1, T2, T3 correspondant aux moyennes trimestrielles des élèves d'une classe.

On souhaite faire l'histogramme des notes de la classe obtenues au premier trimestre.

1. Récupérer les notes du premier trimestre et les stocker dans une liste `t1` d'entiers. Faire de même pour le 2e et le 3e trimestre.
2. Le code ci-dessous permet d'afficher l'histogramme :

```
import matplotlib.pyplot as plt
res = plt.hist(t1,range=(8, 20), bins = range(8, 21),\
               edgecolor="black", label = "T1")
plt.legend()
```

   Explications : La fonction `hist` de la bibliothèque `matplotlib.pyplot` a pour paramètres :

   - un tableau contenant les données à représenter.
   - `range` est un couple `(a, b)` donnant la valeur minimale `a` et la valeur maximale `b` des données à représenter.
   - `bins` est une liste $[v_0, v_1, \cdots, v_n]$ donnant les classes de l'histogramme. Il y a donc n classes : $[v_0, v_1[$, $[v_1, v_2[$, ..., $[v_{n-1}, v_n]$.

   Afficher l'histogramme des notes aux différents trimestres.
3. Ecrire une fonction `moyenne(lst)` qui a pour paramètre une liste et qui renvoie la moyenne des valeurs de `lst`. Puis afficher la moyenne des notes du trimestre 1, du trimestre 2 et du trimestre 3.

4. Ecrire une fonction `ecart_type(lst)` qui a pour paramètre une liste `lst` et qui renvoie l'écart-type des valeurs de `lst`. Puis afficher l'écart-type des notes du trimestre 1, du trimestre 2 et du trimestre 3.

   Rappel : l'écart-type des valeurs $x_1, x_2, \cdots, x_n$ est donné par $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$ avec $\bar{x}$ la moyenne des valeurs $x_1, x_2, \cdots, x_n$.

5. Compléter la fonction `mediane(lst)` qui a pour paramètre une liste et qui renvoie la médiane des valeurs de `lst`. On pourra distinguer le cas où le nombre de valeurs est pair ou impair. Puis afficher la médiane des notes du trimestre 1, du trimestre 2 et du trimestre 3.

```
def mediane(lst):
    lst_trie = sorted(lst) # lst_trie est la liste lst trie
```

6. Ecrire dans un fichier `statistiques.csv` les différentes statistiques qu'on vient de calculer.