MANIPULATION DE FICHIERS

Informatique Tronc Commun – 1ère année

E.Clermont

INTÉRÊT DE LA MANIPULATION DE FICHIERS

- La lecture ou l'écriture dans un fichier texte peut servir pour réaliser beaucoup de tâches.

Exemples : effectuer des calculs, tracer des graphiques.

- Ce que vous serez amenés à réaliser le plus souvent consistera à extraire de fichiers textes des listes de données numériques issues d'un système de mesures quelconque permettant l'exportation au format texte (exemple : accélérations d'un accéléromètre permettant de remonter à la vitesse et la position…).

- Selon les logiciels utilisés, les données seront regroupées et séparées différemment. Souvent, à chaque temps de mesure ou chaque mesure sera associée une ligne. Pour chaque ligne, les différentes données peuvent être séparées par différents « séparateurs », virgule, point-virgule, espace, tabulation…

- **Objectifs du cours** : apprendre à
  - Lire et extraire des données notamment sous forme de listes,
  - Créer un fichier et y ajouter les valeurs d'une liste.

LECTURE D'UN FICHIER

- **Ouverture d'un fichier**

Pour ouvrir un fichier texte en lecture en python, on peut écrire :

```
fichier = open(Nom_Fichier, "r")
```

- A la variable fichier est associé le fichier texte

  L'objet fichier créé est ce qu'on appelle **un flux.**

- L'option r signifie« read » : le fichier est ouvert uniquement en lecture.

(=>on ne peut donc pas écrire à l'intérieur).

LECTURE D'UN FICHIER

- **Ouverture d'un fichier** `fichier = open(Nom_Fichier, "r")`
- Nom_Fichier est une variable contenant le chemin du fichier à ouvrir :
  - En **chemin relatif** : dans le même dossier sont présents le code python et le fichier texte.

Exemple : `Nom_Fichier = "Exemple.txt "`

Ce chemin relatif peut également être constitué du nom du fichier précédé d'un ou plusieurs noms de dossiers.

Exemple : `Nom_Fichier = "Files\\resu\\ Exemple.txt"`

(Le fichier Exemple.txt se trouve alors dans le dossier resu qui se trouve dans le dossier File)

  - En **chemin absolu** : on veut ouvrir un fichier dans l'ordinateur à un endroit spécifique, on précise alors le chemin complet.

Exemple : `Nom_Fichier = "C:\\Users\\ec\\Desktop\\ Exemple.txt"`

Veiller à doubler les anti-slashs

- **Remarque** : On pourrait ne pas passer par la variable Nom_Fichier mais écrire directement :
  `fichier = open("Exemple.txt", "r")`

LECTURE D'UN FICHIER

- **Lecture complète ou partielle**

- Les caractères sont lisibles uniquement les uns après les autres, sans possibilité de retour en arrière ni de saut en avant. => pas très pratique.
- Besoin de convertir ce flux en chaîne de caractères.
  Un moyen simple est d'utiliser la fonction read, qui lit le flux dans son entier et le convertit en chaîne de caractères.
- Soit le fichier Exemple.txt :

CPGE - Informatique tronc commun
Moyenne de la classe S1 :
10
Moyenne de la classe S2:
12

```
fichier = open("Exemple.txt","r")
chaine =  fichier.read()
fichier.close()
```

chaine vaut : 'CPGE - Informatique tronc commun \nMoyenne de la classe S1:\n 10\nMoyenne de la classe S2:\n12'

Rappel :Dans une chaîne de caractères, le passage à la ligne est représenté par le caractère spécial "\n"

LECTURE D'UN FICHIER

- **Lecture complète ou partielle**

- Cette méthode est la plus simple, mais n'est évidemment adaptée que si le fichier contient un texte court.
- Si nécessaire, on peut préciser en argument de la fonction read le nombre de caractères que l'on souhaite lire.
- Exemple de lecture des caractères par groupes de 5 :

CPGE - Informatique tronc commun
Moyenne de la classe S1 :
10
Moyenne de la classe S2:
12

```
fichier = open("Exemple.txt","r")
liste = []
txt = fichier.read(5)
while len(txt) > 0:
    liste.append(txt)
    txt = fichier.read(5)
fichier.close()
```

- Affichage obtenu :
  ['CPGE ', '- Inf', 'ormat', 'ique ', 'tronc', ' comm', 'un\nMo', 'yenne', ' de l', 'a cls', 'se S1', ' :\n1', '0\nMoy', 'enne ', 'de la', ' clas', 'se S2', ' :\n12']

LECTURE D'UN FICHIER

- **Lecture par lignes**
  - **Lecture ligne par ligne**

- On peut parcourir chacune des lignes du fichier à l'aide d'une boucle for :

  for Ligne in fichier :

  Ligne est alors associée à une ligne du fichier au format chaine de caractères (str).

- Exemple de lecture ligne par ligne:

CPGE - Informatique tronc commun
Moyenne de la classe S1 :
10
Moyenne de la classe S2:
12

```
fichier = open("Exemple.txt","r")
for Ligne in fichier:
    print(Ligne)
fichier.close()
```

Cette écriture permet de parcourir toutes les lignes du fichier, les unes après les autres.
Attention, Ligne n'est pas un nombre, c'est le contenu d'une ligne du fichier !

LECTURE D'UN FICHIER

- **Lecture par lignes**
  - **Lecture ligne par ligne**

- Un retour à la ligne est spécifié en fin de ligne par "\n" (visible en affichant la liste des lignes par la méthode readlines).
- On peut tester la **présence d'une ligne vide** (hormis la dernière) à l'aide de la commande : **if** Ligne == '\n'**:** . Cela peut permettre de mettre fin à une lecture de fichier dont la fin contient plusieurs lignes vides par exemple.

Pour la dernière ligne, on peut écrire : **if** Ligne == ''**:** ou **if** Ligne == ""**:**

- **Si besoin**, on peut ajouter un compteur afin d'identifier les numéros de lignes pour n'en traiter qu'une partie.

- **Attention** : lorsque l'on parcourt le fichier avec for Ligne in fichier:

  le fichier a été parcouru et ne peut pas l'être une 2ème fois.

Le programme ne renvoie pas d'erreur pour autant, le for n'exécute juste rien.

Pour parcourir 2 fois le fichier, il faut ouvrir une 2ème fois le fichier.

LECTURE D'UN FICHIER

- **Lecture par lignes**
  - **Lecture complète de l'ensemble des lignes**

- Il est possible de récupérer directement l'ensemble des lignes d'un fichier en écrivant :
  Liste_Lignes = fichier.readlines( )

Attention : Il faudra toutefois effectuer le post-traitement de chaque ligne par la suite. C'est très pratique car en 3 lignes, on obtient toutes les lignes qu'il faudra post-traiter.

```
fichier = open("Exemple.txt","r")
Liste_Lignes = fichier.readlines()
print(Liste_Lignes ) # cette instruction peut être realisée avant ou apres fichier.close()
fichier.close()
```

génère l'affichage :

['Informatique tronc commun \n', 'Moyenne de la classe S1 :\n', '10\n', 'Moyenne de la classe S2:\n', '12']

- Chaque élément de la liste Liste_Lignes est alors une chaîne de caractères contenant une des lignes du texte ouvert.
- Attention, la fin de chaque ligne, sauf la dernière contient un "\n".
- Remarque :le fichier se termine après le nombre 12 sans nouvelle ligne vide, ie sans retour à la ligne.

LECTURE D'UN FICHIER

- **Lecture par lignes**
  - **Lecture complète de l'ensemble des lignes**

- **Attention** : lorsque l'on parcourt le fichier avec fichier.readlines( ), le fichier a été parcouru et ne peut pas l'être une 2ème fois.

Il ne renvoie pas d'erreur pour autant, le for n'exécute juste rien…

Si vous voulez parcourir 2 fois le fichier, il faut ouvrir 2 fois le fichier… Sinon, aucune erreur n'est affichée, mais le fichier n'est pas relu

- **Remarque** : Après avoir ouvert un fichier sous Python, écrire au singulier fichier.readline(), permet de lire une ligne.

On peut ainsi aisément ne pas lire la première ligne en écrivant

fichier.readline()

puis

Lignes = fichier.readlines() pour récupérer toutes les autres lignes du fichier.

=> peut être utile si le fichier comporte un en-tête qui ne sont pas utiles à récupérer.

LECTURE D'UN FICHIER

- **Découpage des données de chaque ligne**

- Il faut alors s'adapter au cas de figure afin d'en extraire les données importantes. La fonction utile dans notre cas est la fonction **split**.

- Exemples d'utilisation :

Ligne = "10 20 30"

Ligne.split() **Résultat : ['10', '20', '30']**

Ligne = "10;20;30"

Ligne.split(";") **Résultat : ['10', '20', '30']**

- Sans arguments dans la fonction split, il y a un découpage chaque fois qu'il y a un ou plusieurs espaces, ou des tabulations
- Avec argument, il y a découpage chaque fois que l'argument mis entre guillemets est rencontré. Dans l'exemple : ;

LECTURE D'UN FICHIER

- **Découpage des données de chaque ligne**

- Exemple:

```
Liste_Totale = []
fichier = open("Exemple.txt","r")
for Ligne in fichier:
    Liste_Ligne = Ligne.split() #Liste_Ligne recoit tous les mots séparés par un espace
    print ("Liste_ligne : ",Liste_Ligne)
    Liste_Totale.append(Liste_Ligne) #Remplissage d'une liste globale
fichier.close()
print("Liste_Totale : ",Liste_Totale)
```

- Ce programme génère l'affichage :

```
Liste_ligne :  ['CPGE' ,'Informatique', 'tronc', 'commun']
Liste_ligne :  ['Moyenne', 'de', 'la', 'classe', 'S1', ':']
Liste_ligne :  ['10']
Liste_ligne :  ['Moyenne', 'de', 'la', 'classe', 'S2:']
Liste_ligne :  ['12']
Liste_Totale :  [['Informatique', 'tronc', 'commun'], ['Moyenne', 'de', 'la', 'classe', 'S1', ':'], ['10'], ['Moyenne', 'de', 'la', 'classe', 'S2:'], ['12']
```

- On obtient alors une liste de chaînes de caractères liste_Totale.
- => On peut récupérer les valeurs associées en appelant chaque terme de la liste et **en le transformant (si besoin)** en entier (int) ou flottant (float).

LECTURE D'UN FICHIER

- **Suppression du retour à la ligne**

- Chaque ligne d'un fichier contient à la fin un retour à la ligne `\n`. La dernière ligne peut d'ailleurs ne pas en avoir.
- Ainsi, pour le retirer, il existe 2 solutions :
- Ligne = Ligne[0,len(Ligne)-1] ou Ligne[:len(Ligne)-1] qui ne fonctionnera que si la ligne contient un retour à la ligne

(Attention : la dernière ligne des fichiers peut ne pas en contenir)

- **Remarque** : Quand une donnée numérique est stockée sous forme de chaine de caractères et que l'on utilise une conversion de type telle que int() ou float() car on a besoin de manipuler une valeur numérique,
  le « \n » sera automatiquement supprimé

```
>>> ch= "1\n"

>>> float(ch)
1.0

>>> int(ch)
1
```

LECTURE D'UN FICHIER

- **Fermeture d'un fichier**

- Après avoir lu un fichier texte, il faut le refermer sous Python avec la fonction **close** :

  fichier.close()

- **Attention** : Lorsqu'un fichier a été ouvert en Python avec la commande open sans être refermé avec la commande close,
  => Risque d'avoir des problèmes pour le supprimer ultérieurement.

LECTURE D'UN FICHIER

- **Ouverture d'un fichier**

- Mode ajout : ouvre le fichier et ajoute du texte après ce qui est présent (impossible de lire ce qui est avant avec ce mode) :

```
fichier = open(Nom_Fichier, "a")
```

- Mode écrasement : ouvre le fichier en écrasant son contenu.

```
fichier = open(Nom_Fichier, "w")
```

Remarques :

- Si le fichier n'existe pas, il est créé.
- Avec ces deux modes d'ouverture, les fichiers ne peuvent être lus en même temps.
- En mode a, python se place à la fin du fichier pour y ajouter des données.

LECTURE D'UN FICHIER

- **Ouverture d'un fichier**

Il existe en Python un autre moyen d'ouvrir un fichier avec le mot-clé with qui permet d'ouvrir et de fermer un fichier de manière efficace.

Si l'ouverture ou la lecture du fichier conduit à une erreur, l'utilisation de with garantit la bonne fermeture du fichier.

- **Exemple :**

```
with open("Exemple.txt","r") as fichier :
    for Ligne in fichier:
        print(Ligne)
```

CRÉATION ET ÉCRITURE DANS UN FICHIER

- **Ajout de lignes**

- Pour ajouter du texte à un fichier texte, il suffit d'utiliser la commande **write**

Exemple dans lequel on ajoute l'un des termes d'une liste de nombres

```
fichier.write( str(Liste[i]) )
```

- **Attention**: Il faut que l'argument de la fonction write soit une chaîne de caractères.

- Si ce que l'on ajoute doit être une ligne, on ajoute "\n" à la fin pour indiquer un retour à la ligne :

```
fichier.write( str(Liste[i]) + "\n" )
```

Pour ajouter uniquement un retour à la ligne, il suffit donc d'écrire :

```
fichier.write("\n")
```

- Remarque : pour afficher la chaîne « \n » dans un texte et donc ne pas créer de retour à la ligne, il faut doubler l'antislash : fichier.write("\\n").

CRÉATION ET ÉCRITURE DANS UN FICHIER

- **Fermeture**

- Pour finaliser un fichier texte, il faut le fermer avec la commande **close**:

```
fichier.close()
```

MANIPULATION DE FICHIERS

Informatique Tronc Commun – 1ère année

E.Clermont