INF1 : Algorithmique et Programmation

Cours 9 : Fichiers

Domitile Lourdeaux

Université de technologie de Compiègne

Printemps 2024

Motivations

Jusqu'à maintenant

- Entrées / sorties limitées au clavier et à l'écran
- Impossible de traiter des données en grand nombre
- Impossible de conserver les données et résultats produits : après exécution, tout est perdu ...

Solution

- Lire ou écrire dans un fichier (sur un disque dur par exemple)

Définition

Définition

Un fichier est :

- une collection d'informations stockée sur un support physique (disque dur, bande, CD, DVD, clé USB ...).

Un fichier permet de conserver durablement l'information

- Données, résultats, programmes, ...

L'information persiste à l'arrêt du programme

Différents types de fichier

Fichiers texte

- Ne contiennent que des caractères imprimables (et des marqueurs de fin de ligne ou de fin de fichier)

Fichiers binaires

- Contiennent des octets
- Lorsqu'on enregistre des données, il faut conserver leur type pour pouvoir les relire

Python utilise par défaut des fichiers texte

- On peut toutefois utiliser aussi des fichiers binaires
- Le module pickle permet d'enregistrer les données en conservant leur type

Fichiers de texte

Texte

- Suite de caractères alphanumériques (et éventuellement de marqueurs)
- Ensemble de lignes

Marqueur de fin de ligne

- Un ou deux caractères spéciaux suivant le système d'exploitation
  - Windows : CR LF (0D, 0A en hexadécimal) ou \r \n
  - Linux ou macOs : LF seulement
- Python unifie cela avec \n qui fonctionne pour tout système d'exploitation

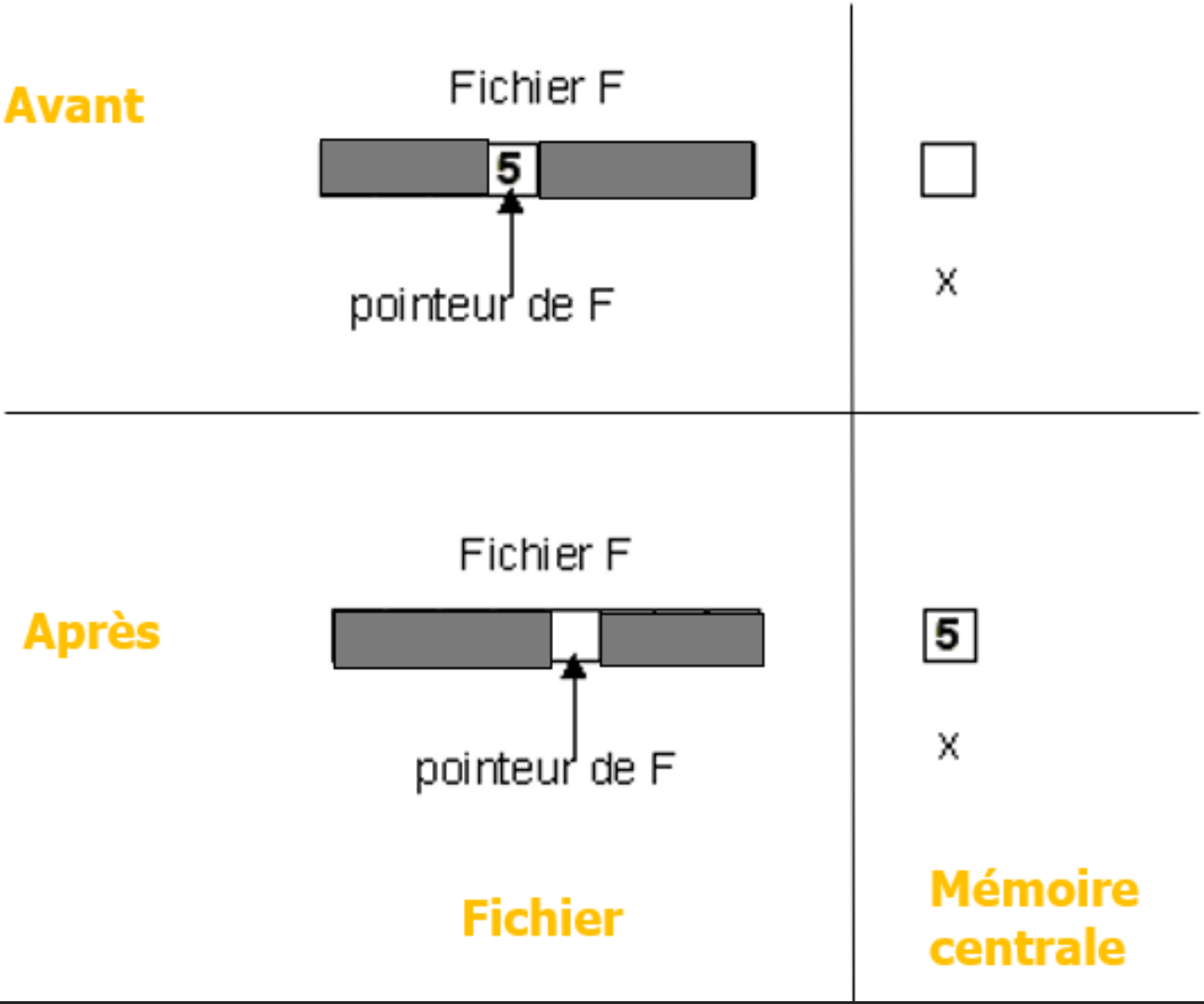
Principe général d'accès aux fichiers

1. Ouverture du fichier et création d'un « objet fichier »
   - Lecture
   - Ecriture
   - Ajout
2. Instruction(s) de lecture ou d'écriture (ou de positionnement)
3. Fermeture du fichier

Pointeur sur fichier (1)

Lecture d'un élément du fichier

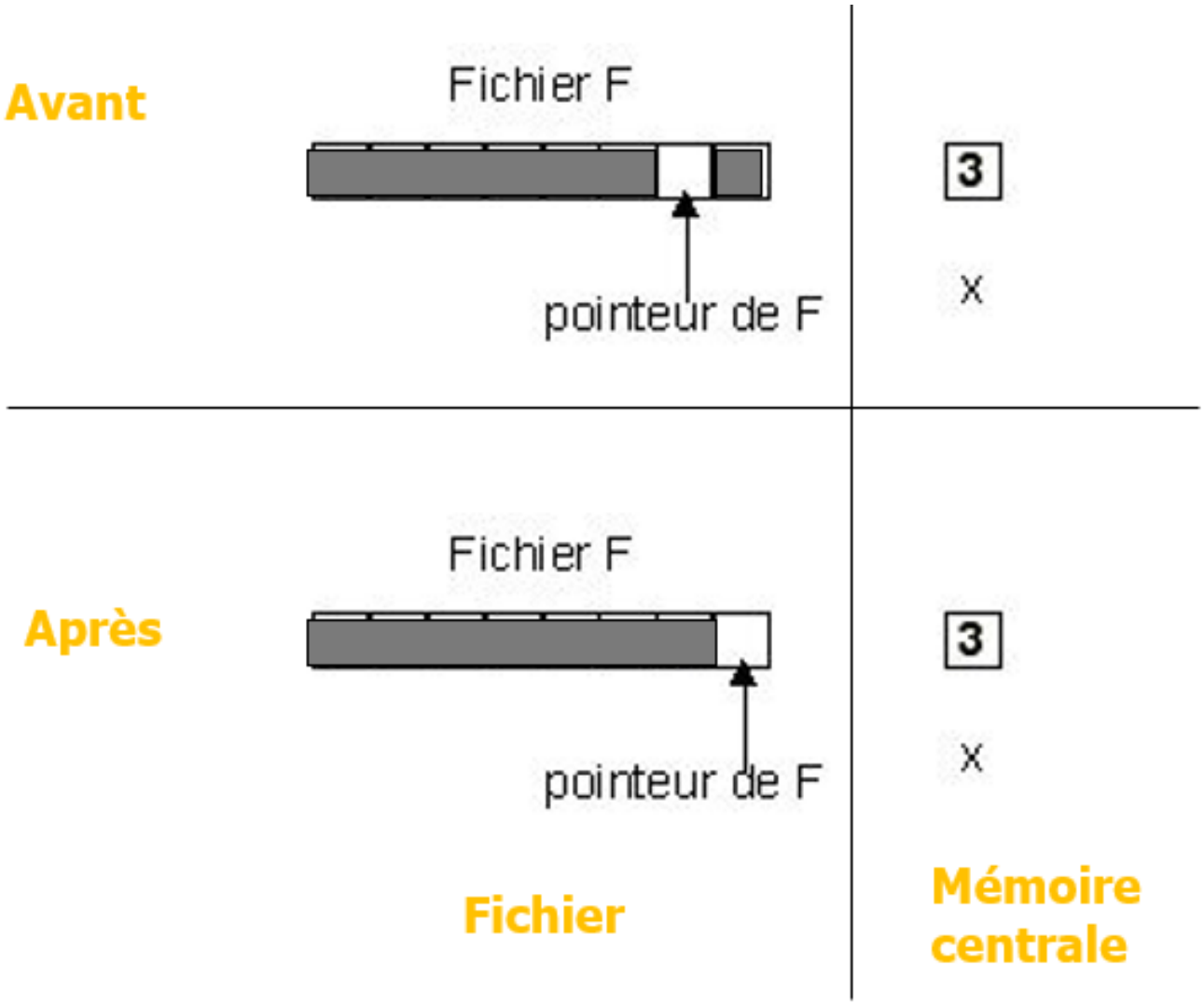
lire(F, x)



Pointeur sur fichier (2)

Ecriture d'un élément dans un fichier

ecrire(F, x)



Principe général d'accès aux fichiers

1. **Ouverture du fichier et création** d'un « objet fichier »
2. Instruction(s) de **lecture** ou d'**écriture**
3. **Fermeture** du fichier
   - file.close()

Ouverture des fichiers en Python (1)

- Lecture : ‘r’
  - file = open(‘my_file.txt’, ‘r’)
- Ecriture ‘w’
  - file = open(‘my_file.txt’, ‘w’)
  - Le fichier est créé s’il n’existe pas et écrasé sinon
- Ajout : ‘a’
  - file = open(‘my_file.txt’, ‘a’)
  - Les enregistrements sont ajoutés à la fin du fichier

Lecture	'r' lecture seule 'r+' lecture et écriture	Erreur si le fichier n'existe pas ou n'est pas du bon type
Ecriture	'w' écriture seule 'w+' écriture et lecture	Le fichier est créé s'il n'existe pas et vidé de son contenu s'il existe
Ajout	'a' écriture et lecture 'a+' écriture et lecture	Le fichier est créé s'il n'existe pas et son contenu est conservé s'il existe
Mode binaire	'b'	Doit être combiné avec les modes précédents (Ex : 'rb' ou 'wb')

Ouverture des fichiers en Python (2)

Encodage

- objetFichier = open(**nom_fichier**, **mode**, encoding = '...')

Remarque

- L'encodage est optionnel
- S'il n'est pas précisé, c'est l'encodage de la plateforme qui est utilisé
- Pour l'interopérabilité, il est préférable d'utiliser 'utf-8' : encoding = 'utf-8'

Ouverture des fichiers en Python (3)

Ouverture du fichier 'my_file.txt' en lecture

- file = open('my_file.txt', 'r', encoding = 'utf-8')

Remarque

Si le fichier n'est pas dans le répertoire courant, il faut indiquer le chemin d'accès complet (ou relatif).

- file = open('D :/my_directory/my_file.txt', 'r')

Chemin d'accès

getcwd() permet d'avoir le chemin du répertoire courant (working directory) :

```
>>> from os import getcwd
... wd = getcwd()
... print(wd)
```

Ou :

```
>>> import os
... wd = os.getcwd()
```

Pour changer de répertoire : chdir("C :/Users/xxx/mon-fichier")

Lecture séquentielle d'un fichier texte

Principe général

- Boucle de lecture : on lit successivement des éléments jusqu'à ce que la fin du fichier soit atteinte
- Question : comment détecter la fin du fichier ?
  - Fin du fichier : ligne = ""

Plusieurs modes possibles

- Un ou plusieurs caractères à la fois
- Ligne par ligne
- Lecture du fichier complet en une seule fois

Algorithme de lecture caractère par caractère

Algorithm 1 Lecture de caractères avec détection de la fin de fichier

Ouvrir(fichier)
fin ← *False*
`tant que` *fin* == *False* `faire`
    Lire (*caractere*) dans le fichier
    `si` *caractere* `alors`
        Traitement de *caractere*
    `sinon`
        fin = *True*
    `fin`
`fin`
Fermer(fichier)

Lecture caractère par caractère en python

Lecture avec la méthode read(nbcaracteres)

```
file = open("fichier_peintres.txt", "r")
fin = False
while not fin :
    c = file.read(1)
    if c :
        if c != '\n' :
            print(c, end="")
        else :
            print()
    else :
        fin = True
file.close()
```

Algorithmes de lecture de lignes

Algorithm 2 Lecture de ligne avec détection de la fin de fichier

Ouvrir(fichier)
Lire (*ligne*) dans le fichier
`tant que` *line* $\neq$ "" `faire`
  Traitement de *ligne*)
  Lire (*ligne*) dans le fichier
`fin`
Fermer(fichier)

Lecture avec ligne à ligne en python (1)

Lecture avec la méthode readline()

```
file = open("fichier_peintres.txt", "r")
ligne = file.readline()
while ligne != "":
    print(ligne)
    ligne = file.readline()
file.close()
```

```
Da Vinci

Michelangelo

Raphaël

Bellini
```

Remarque

- Chaque ligne se termine par \n
- Il y a une ligne vide après chaque ligne

Lecture avec ligne à ligne en python (2)

Lecture avec la méthode readline()

```
file = open("fichier_peintres.txt", "r")
ligne = file.readline()
while ligne != "":
    ligne = ligne.strip()
    print(ligne)
    ligne = file.readline()
file.close()
```

```
Da Vinci
Michelangelo
Raphaël
Bellini
```

```
file = open("fichier_peintres.txt", "r")
ligne = file.readline()
while ligne != "":
    print(ligne[:-1])
    ligne = file.readline()
file.close()
```

Lecture de toutes les lignes en python (1)

Lecture avec la méthode readlines()

```
file = open("fichier_peintres.txt", "r")
lignes = file.readlines()
for ligne in lignes:
    print(ligne)
file.close()
```

Remarque

readlines() retourne les lignes du fichier dans une liste

Lecture de toutes les lignes en python (2)

```
>>> file = open("fichier_peintres.txt", "r")
... print(file.readlines())
... file.close()
...
['Da Vinci\n', 'Michelangelo\n', 'Raphaël\n', 'Bellini\n']
```

L'objet fichier est un itérable (1)

Un objet fichier est un itérable » en python

On utilisera donc de préférence la méthode suivante :

Algorithm 3 Lecture de lignes si le fichier est un itérable

file = **Ouvrir**(fichier)
`pour chaque` *ligne* `in` *file* `faire`
  Traitement de *ligne*
`fin`
Fermer(fichier)

L'objet fichier est un itérable (2)

Un objet fichier est un « itérable » en python

On utilisera donc de préférence la méthode suivante :

```
file = open("fichier_peintres.txt", "r")
for ligne in file:
    print(ligne)
file.close()
```

Lecture et positionnement

Si f est l'objet fichier :

f.seek(n)	déplace le curseur de n octets
f.seek(n, orig)	si orig = 0 équivaut à f.seek(n) si orig = 1 déplace le curseur de n à partir de la position courante si orig = 2 déplace le curseur de n à partir de la fin du fichier
f.tell()	renvoie la position du curseur à partir du début du fichier

Et si le fichier n'existe pas ?

Dans ce cas une erreur se produit

Il faut donc sécuriser la lecture en interceptant cette erreur

```
try:
    file = open("fichier_peintres.txt", "r")
    for ligne in file:
        print(ligne)
    file.close()
except IOError:
    print("Erreur : le fichier n'existe pas")
```

Et si le fichier n'existe pas ?

Dans ce cas une erreur se produit

Il faut donc sécuriser la lecture en interceptant cette erreur

```
try:
    file = open("fichier_peintres.txt", "r")
    for ligne in file:
        print(ligne)
    file.close()
except:
    print("Erreur : le fichier n'existe pas")
```

Fonction de test

On peut aussi écrire une fonction pour tester l'existence d'un fichier

```
>>> def existe(filename):
...     try:
...         file = open("filename", "r")
...         file.close()
...         return True
...     except:
...         return False
... nom = input("Nom du fichier : ")
... if existe(nom):
...     print("Ce fichier existe")
... else:
...     print("Ce fichier n'existe pas")
```

Fonction de test

On peut aussi écrire une fonction pour tester l'existence d'un fichier

```
def existe(filename):
    try:
        file = open(filename, "r")
        file.close()
        return True
    except:
        return False

nom = input("Nom du fichier : ")
if existe(nom) :
    print("Ce fichier existe")
    # Traitement du fichier
else:
    print("Ce fichier n'existe pas")
```

Ecrire dans un fichier

Exemple

- Saisie d'un texte par l'utilisateur et écriture dans un fichier

Problème

- Convention d'arrêt pour l'utilisateur ? Ligne vide par exemple ?

Algorithm 4 Ecriture dans un fichier d'un texte saisi par l'utilisateur

---

Lire(ligne) depuis la console
`tant que` *ligne* $\neq$ "" `faire`
  Ecrire (*ligne*) dans le fichier
  Lire (*caractere*) depuis la console
`fin`

Remarque

Si le texte contient une ligne vide, il faut une autre convention

Ecriture des éléments d'un tableau (1)

Exemple : écriture d'un tableau de chaines de caractères

```
file = open("fichier_peintres.txt", "w")
peintres = ["Da Vinci", "Michelangelo", "Raphaël", "Bellini"]
for peintre in peintres :
    file.write(f"{peintre}")
file.close()
```

Contenu du fichier après écriture

```
Da VinciMichelangeloRaphaëlBellini
```

Ecriture des éléments d'un tableau (2)

Ajouter un séparateur de ligne

```
file = open("fichier_peintres.txt", "w")
peintres = ["Da Vinci", "Michelangelo", "Raphaël", "Bellini"]
for peintre in peintres :
    file.write(f"{peintre}\n")
file.close()
```

Contenu du fichier après écriture

```
Da Vinci
Michelangelo
Raphaël
Bellini
```

Ecriture de données de types divers

Pour lire et écrire des données d'un autre type que string, il faut les convertir

Ecriture de notes dans un fichier

```
t = [12.5, 12, 10, 9.5, 8.25, 17.75, 15]

file = open("notes", "w")
for note in t:
    file.write(str(note)+'\n')
file.close()
```

Lecture et calcul de moyenne

```
somme = 0
nb = 0
file = open("notes", "r")
for note in file:
    somme += float(note)
    nb += 1
file.close()
moyenne = somme / nb
print(f"Moyenne = {moyenne:.2f}")
```

Fichiers .csv

- Le format csv est utilisé pour lire ou écrire des données tabulaires compatibles avec des tableurs comme Excel
- Les données en colonnes sont séparées par des virgules (ou des pointsvirgules pour Excel)
- Exemple : Fichier Excel

	A	B	C	D	E	F
1	Bob Dylan	Greatest hits	3	Rainy day women	Blowin' in the wind	The times they are a changin'
2	David Bowie	Blackstar	2	Blackstar	Lazarus	

Fichier CSV produit par Excel:

```
1 Bob Dylan;Greatest hits;3;Rainy day women;Blowin' in the wind;The times they are a changin'
2 David Bowie;Blackstar;2;Blackstar;Lazarus;
```

Lecture d'un fichier .csv

```
# lecture du fichier csv des albums

import csv
f = open("albums2.csv")
albums = csv.DictReader(f, delimiter = ";")
print(albums)
f.close()
```

Exécution

```
[OrderedDict([('artiste', 'Bob Dylan'), ('titre', 'Greatest hits'), ('nb', '3'), ('morceau1', 'Rainy day women'), ('morceau2', "Blowin' in the wind"), ('morceau3', "The times they are a changin'")]), OrderedDict([('artiste', 'David Bowie'), ('titre', 'Blackstar'), ('nb', '2'), ('morceau1', 'Blackstar'), ('morceau2', 'Lazarus'), ('morceau3', '')])]
```

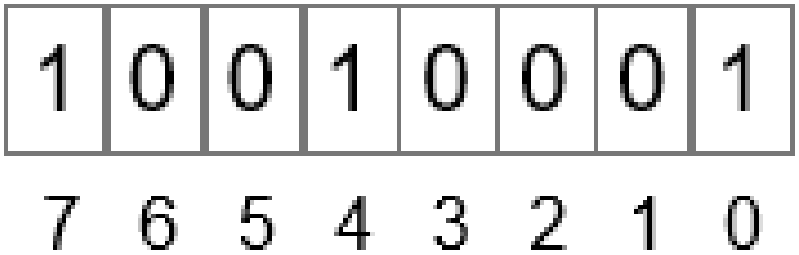
Ecriture d'un fichier .csv

```
# ecriture csv
import csv
enTetes = ["artiste", "titre", "nb","morceau1", "morceau2",
donnees = [{"artiste":"Bob Dylan", "titre":"Greatest hits",
           "morceau1":"Rainy day women",
           "morceau2":"Blowin' in the wind",
           "morceau3":"The times they are a changin'"},
          {"artiste":"David Bowie", "titre":"Blackstar","n
           "morceau1":"Blackstar",
           "morceau2":"Lazarus"}]
f = open("albums2.csv", "w")
w = csv.DictWriter(f, enTetes,delimiter=";")
w.writeheader()
w.writerows(donnees)
f.close()
```

Fichiers binaires

Exemple

- 145 peut s'écrire en utilisant les codes des caractères 1 4 et 5 de la façon suivante : 31 34 35
- Mais aussi sous forme binaire :
  - $1 * 2^7 + 1 * 2^4 + 1 * 2^0$
  - $128 + 8 + 1$



En Python, le module **pickle** permet de lire et d'écrire des fichiers binaires

Le module pickle

Ecriture en binaire

- Ouverture avec "wb"
- Ecriture d'une donnée avec **pickle.dump**
- Le type de la donnée est mémorisé
- Avec l'exemple précédent :

```
t = [12.5, 12, 10, 9.5, 8.25, 17.75, 15]

import pickle
filename = 'notes.bin'
outputfile = open(filename,'wb')
for note in t:
    pickle.dump(note, outputfile)
outputfile.close()
```

Contenu du fichier binaire

notes.bin (édité avec smultron sur mac)

```
80 03 47 40  29 00 00 00  00 00 00 2e  80 03 4b 0c  ..G@).........
2e 80 03 4b  0a 2e 80 03  47 40 23 00  00 00 00 00  ...K¶...G@#..
00 2e 80 03  47 40 20 80  00 00 00 00  00 2e 80 03  ....G@ .......
47 40 31 c0  00 00 00 00  00 2e 80 03  4b 0f 2e      G@1.........K
```

- Le contenu n'est pas directement lisible
- Mais on peut le relire à l'aide du module pickle
- Il contient les 7 nombres écrits sous forme binaire précédemment ainsi que les indications nécessaires pour les relire

Le module pickle

Lecture d'un fichier binaire

- Ouverture avec "rb"
- Lecture d'une donnée avec pickle.load
- Relecture du fichier précédent :

```
import pickle
filename = 'notes.bin'
inputfile = open(filename, 'rb')
eof = False
while not eof:
    try:
        print(pickle.load(inputfile))
    except:
        print("fin de fichier")
        eof = True
inputfile.close()
```

Exécution :

```
12.5
12
10
9.5
8.25
17.75
15
fin de fichier
```

Exercice : Enoncé

Ecrire une fonction permettant de recopier un fichier texte en enlevant les lignes commençant par #

Exercice : Correction

```
def sansCommentaires(source, destination):
    """ Fonction qui recopie un fichier texte en
            enlevant les lignes commen ant par #
        params: fichier texte source avec commentaires
        return: fichier texte destination sans commentaires
    """
    if existe(source):
        inputfile = open(source, "r")
        outputfile = open(destination, "w")
        for ligne in inputfile:
            if ligne[0] != "#":
                outputfile.write(ligne)
        inputfile.close()
        outputfile.close()
```

Questions...