MANIPULATION SUR LES FICHIERS

I- Notion de fichier :

01 **Définition :** Un fichier est un ensemble structuré de données **homogènes** (de même type), nommé et enregistré sur un support de stockage (disque dur, cd, ...). Il peut contenir du texte, des enregistrements, des programmes et/ou des valeurs d'**une façon permanente**.

N.B : En programmation on a deux types de fichier :

- **Fichier texte** : éditable avec un éditeur de texte, permet d'enregistrer des chaines de caractère.
- **Fichier typé (binaires en python)** : illisible par l'homme, mais pouvant mémoriser tout type d'objet (Entier, réel, etc.. des enregistrements).

02 Déclaration :

En Algorithme

T.D.N.T

Type
Nom_enreg = enregistrement Champ1 : type ….. champN : type Fin Nom_type_fiche = Fichier de Nom_enreg

T.D.O

Objet	T/N
Nom_logique_typé1	Nom_type_fichier
Nom_logique_typé2	Fichier de type_élément
Nom_logique_texte	Texte

En python

En python la déclaration se fait lors de l'ouverture (sera vu par la suite)

Le nom logique d'un fichier représente la variable de type fichier utilisé dans le programme, cependant le nom physique est le nom avec lequel est enregistré sur le support de stockage.

Exemple : T.D.N.T

Type
eleve = enregistrement id : entier nom, prenom, classe : chaine moy : réel Fin Fich = Fichier d'eleve

T.D.O

Objet	T/N
F1	Fich
F2	Fichier d'entier
F3	Texte

II. Les fichiers d'enregistrements : (fichier typé ou fichier de données)

En programmation, un fichier typé est une séquence **organisée** d'enregistrements (ou de données typées) représentant un ensemble d'entités de même type.

01 **Organisation :**

C'est la façon dont les enregistrements sont placés dans le fichier. Elle détermine la méthode de manipulation des informations du fichier, qu'on appelle **méthode d'accès**.

- Les données dans un fichier **typé** sont enregistrées les uns à la suite des autres, de façon linéaire.

Enregistrements dans le fichier →	E_1	E_2	E_3	.....	E_{n-1}	E_n
Numéros d'ordres →	0	1	2		n-2	n-1

- Les numéros d'ordres commencent par **0**, c-à-d l'enregistrement **1** admet le numéro d'ordre **0**.
- En python, Les numéros d'ordres sont de la forme **i*taille de l'objet écrite avec i de 0 à n-1**

02 Méthodes d'accès :

Question : d'après vous comment on peut accéder aux enregistrements d'un fichier typé ?

Réponse :

1ère méthode : On parcourt les enregistrements un par un jusqu'à l'enregistrement souhaité.

2ème méthode : On localise ou on pointe sur l'enregistrement souhaité directement.

Accès séquentiel :

Pour accéder à l'i^{ème} enregistrement d'un fichier, on doit passer par l'i^{ème} -1 d'abord. L'organisation correspondante est dite ***séquentielle*** **ou** ***consécutive***.

Accès direct :

On accède directement à l'i^{ème} enregistrement d'un fichier par l'intermédiaire d'un *numéro d'ordre*. L'organisation correspondante est dite **directe**.

03 Opérations sur les fichiers types:

a) Ouverture :

Syntaxe :

En algorithme	En Python
Ouvrir ("Chemin\Nom_physique", Nom_logique , "Mode")	Nom_logique = open ("Chemin\\Nom_physique" , "mode")

b) Mode d'ouverture :

Syntaxe :

"r"	Valeur par défaut. Ouvre le fichier en lecture. Erreur Si le fichier n'existe pas.
"w"	Ouvre le fichier en écriture. Si le fichier existe son contenu sera effacé sinon (le fichier n'existe pas) il sera créé.
"a"	Ouvre le fichier en mode ajout à la fin (APPEND). Si le fichier n'existe pas il sera créé.
Pour spécifier le type de fichier, on ajoute une deuxième lettre dans le mode d'ouverture	
"t"	Par défaut, fichier texte
"b"	Fichier binaire (pour nous, sera utilisé pour les fichiers typés)

Exemples :

En algorithme	En Paython
Ouvrir ("eleve.org", F, "wb")	*# Ouverture d'un fichier binaire (typé) sans chemin* **F = open ('eleve.org','wb')**
Ouvrir ("C:\4SI\eleve.org", F, "wb")	*# Ouverture d'un fichier binaire (typé) avec chemin* *# erreur si le chemin n'existe pas* **F = open ('c:\4SI\eleve.org','wb')**

c) Ouverture en lecture :

Ouvre le fichier en lecture. Le pointeur point l'enregistrement d'ordre 0.

Syntaxe :

En algorithme	En Python
Ouvrir ("Chemin\Nom_physique", Nom_logique , "rb")	Nom_logique = open ("Chemin\Nom_physique" , "rb")

d) <u>Ouverture en ajout</u> : Ouvre le fichier en écriture à la fin.

Syntaxe :

En algorithme	En Python
Ouvrir ("Chemin\Nom_physique", Nom_logique , "ab")	Nom_logique = open ("Chemin\Nom_physique" , "ab")

A chaque opération de lecture ou d'écriture d'une valeur, le pointeur du fichier avance automatiquement d'une position.

e) Ecriture dans un fichier : Ecriture dans un fichier déjà ouvert en mode écriture.

En algorithme	En Python
Ecrire (nom logique, variable)	**from pickle import *** **dump** (variable, Nom_logique)

f) Lecture à partir d'un fichier :

Lecture à partir d'un fichier déjà ouvert en mode lecture.

Syntaxe :

Algorithme	Python
Lire (nom_logique, variable)	**Variable = load** (nom_logique)

g) Test de fin fichier :

A chaque moment nous pouvons tester si nous avons atteint la fin d'un fichier en lecture.

Syntaxe :

Algorithme	Python

Fin_fichier (nom_logique)	**ok=False**
	while not ok:
	  try :
	  variable = load(nom_logique)
	  except : # Si fin fichier on arrête l'exécution
	  ok=True

III- Les fichiers textes :

Les informations sont sous un format texte qui est lisible par n'importe quel éditeur de texte.

Opérations sur les fichiers textes:

01 **Ecriture dans un fichier texte :** Ecriture dans un fichier déjà ouvert en **mode écriture**.

Syntaxe :

Algorithme	Python
Ecrire (nom logique, informations) **ou** **Ecrire_nl (nom logique, informations)**	**Nom_logique . write** (informations)

En algorithme : l'écriture se fait ligne par ligne

En Python : pour écrire une ligne on doit ajouter '**\n**' à la fin de l'information à écrire.

Exemple : f.write ("Foulen Ben Foulen\n")

Remarque

En python, on peut utiliser la structure **with** dans l'ouverture du fichier:

```
with open ("eleve.txt","w") as f:
    .....
```

✓ En lisant une donnée (chaine) dans un fichier texte le **pointeur** se met dans la ligne suivante automatiquement, le retour à la ligne implique un espace après la chaine donc son contenu change, pour cela on besoin parfois de supprimer l'espace avec la fonction sous_chaine (en python **rstrip ()**)

02 • **Lecture d'un fichier texte:** Lecture à partir d'un fichier déjà ouvert en **mode lecture**.

Syntaxe :

	Type de lecture	Rôle
Algo.	Lire (nom_logique, ch)	Lecture de la totalité d'un fichier
Python	Ch= nom_logique. read ()	
Algo.	Lire_ligne (nom_logique, ch)	Lecture ligne par ligne
Python	Variable = nom_logique. readline ()	
Python	**Variable = nom_logique.read** (nombre de caractères à lire)	Lecture d'un nombre de caractères à partir d'un caractère pointé

03 Test de fin fichier :

A chaque moment nous pouvons tester si nous avons atteint la fin d'un fichier en lecture.

Syntaxe :

Algorithme	Python
Fin_fichier (nom_logique)	la chaine lue et vide

Remarque La lecture en python se fait totalement ou ligne par ligne :

1ère méthode : Lecture avec read ( ) :

```
# ouverture du fichier en lecture
f = open ("eleve.txt","r")
# Lecture de la totalité du fichier
ch = f.read( )
# Affichage
print(ch)
f.close()
```

2ème méthode : Lecture ligne par ligne

```
    # ouverture du fichier en lecture
f = open ("eleve.txt","r")
ch = f.readline ( )
while ch !="":
    print (ch)
    ch = f.readline ( )
f.close()
```