Faculté des Sciences
كلية العلوم

Département d'Informatique
Filière MIP - Semestre 2
2023-2024

Module : Informatique II (Algorithmique II / Python)

Chapitre 4

Les enregistrements et fichiers

Pr. Khadija Louzaoui

Les enregistrements et fichiers

- Types personnalisés
- Enregistrements (Structures)
  - Définition d'un enregistrement
  - Déclaration d'un enregistrement
  - Déclaration des variables structurées
  - Manipulation des enregistrements
  - Passage d'un enregistrement en paramètre
  - Tableaux et enregistrement
- Fichiers

Types personnalisés

- Les types de données fondamentales:
  Entier, réel, caractère, chaine de caractères, et booléen
- **Peut-on définir nos propres types?** **Oui**

La déclaration des types structurés se fait dans une section spéciale des algorithmes appelée ***Type***, qui précède la section des variables.

Syntaxe:

```
Type nom_nouveau_type = définition_du_nouveau_type
```


Types personnalisés

Syntaxe:

```
Type nom_nouveau_type = définition_du_nouveau_type
```

Exemple 1:

```
Type int= entier
Variables n : int      {n est une variable de type int (entier)}
```

Exemple 2:

```
Type nom= chaine de caractères[50]
Variables
   ch : nom      {ch est une chaine de 50 caractères}
   T : Tableau[100] de nom   {T est un Tableau de 100 chaines de 50 caractères chacun}
```

Activité

Un établissement scolaire organise les informations concernant ses classes dans une liste identique à la suivante :

N°	code	Nom	Prénom	Moyenne	Observation
1	E5022	Alim	Ahmed	12	Néant
2	E1478	Marzi	Loubna	14	Redoublante
...	...	...	...	...	...
...	...	...	...	...	...
50	J4578	Razi	Amine	10	Dispensé du sport

Le directeur de l'établissement veut créer un programme permettant la saisie et le traitement de ces listes sachant que chaque classe comporte au maximum 30 élèves.

1. Donnez la structure de données nécessaire pour les objets à utiliser.
2. Donnez une déclaration algorithmique de ces objets.

5

Activité

Nous remarquons que cette liste comporte une information alphanumérique (**Code**), des informations numériques (**N°**, **Moyenne**) et d'autres alphabétiques (**Nom**, **Prénom**, et **Observations**).

	1	2						30
Num								
Code								
Nom								
Prenom								
Moyenne								
Observation								

6

Activité

Est-il possible de regrouper ces variables au sein d'un même tableau ?

Bien sûr que **NON** car un tableau ne peut contenir que des éléments de **même type**. Mais nous pouvons utiliser 6 tableaux différents déclarés comme suit :

```
Type Tab= Tableau[50] de chaine de caractères

Variables
    Numero : Tableau[ 30] de entier
    Code : Tableau[30] de Tab
    Nom : Tableau[30] de Tab
    Prenom : Tableau[30] de Tab
    Moyenne : Tableau[ 30] de réel
    Obesrvation : Tableau[ 30] de Tab
```

7

Activité

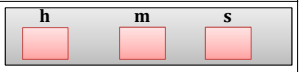
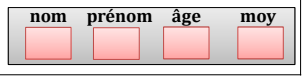
- Nous venons de voir que les variables simples ou les tableaux ne permettent pas de ranger des données de types différents.
- Si nous voulons établir par exemple une structure comportant en même temps des informations alphanumériques, numériques et alphabétiques, nous devons créer un nouveau **TYPE** qui permet de les regrouper.
- Nous allons voir une nouvelle structure appelée **ENREGISTREMENT** (**CLASSE** en Python) qui permet de réaliser cette tâche.

8

Définition d'un enregistrement

Un enregistrement est un type de données (**structure**) défini par l'utilisateur et qui permet de grouper un nombre fini d'éléments (ou **champs**) de **types** éventuellement **différents** (alphabétique, numérique, logique,…) sous un **nom commun**.

Exemple :

La structure « **Date** » est composée du jour, mois, et année.	**Date**	
La structure « **Temps** » est composée du heure, minute, et second.	**Temps**	
La structure « **Etudiant** » est composée du nom, prénom, âge, et moyenne.	**Etudiant**	

9

Définition d'un enregistrement

Les enregistrements en algorithmique sont des **structures** de données dont les éléments peuvent être de différents types et qui se rapportent **à la même entité sémantique**.

Les éléments qui composent un enregistrement sont appelés **champs**. Avant de déclarer **une variable enregistrement**, il faut avoir au préalable définit son **type**, c'est à dire le **nom** et le **type** des champs qui le composent.

Le type d'un enregistrement est appelé type structuré. (Les enregistrements sont parfois appelé structures)

10

Déclaration d'un enregistrement

La déclaration d'un enregistrement (les types structurés) se fait, dans les algorithmes, avant la déclaration des variables.

Syntaxe :

```
Type Nom_Enregistrement = Enregistrement
        Nom Champ 1 : Type Champ 1
        Nom Champ 2 : Type Champ 2
        ...
        Nom Champ n : Type Champ n
Fin Nom_Enregistrement
```

```
Type Nom_structure = Structure
        Nom Champ 1 : Type Champ 1
        Nom Champ 2 : Type Champ 2
        ...
        Nom Champ n : Type Champ n
Fin Nom_structure
```

11

Déclaration d'un enregistrement

Python

Le type enregistrement n'existe pas vraiment en Python, mais plusieurs solutions sont possibles : les tuples, les dictionnaires ou les classes.

Syntaxe : Dictionnaire

```
Nom_enregistrement = {
        " Nom Champ 1" : Type Champ 1,
        " Nom Champ 2 " : Type Champ 2,
          ...
        " Nom Champ n " : Type Champ n
}
```

```
Nom_enregistrement = { }    // signifie enregistrement Vide
```

12

Déclaration d'un enregistrement

Python

Le type enregistrement n'existe pas vraiment en Python, mais plusieurs solutions sont possibles : les tuples, les dictionnaires ou les classes.

Syntaxe : Classe

```
Class Nom_enregistrement :
        Nom Champ 1 : Type Champ 1
        Nom Champ 2 : Type Champ 2
        ...
        Nom Champ n : Type Champ n
```


Déclaration d'un enregistrement

Exemple :

Déclarer un enregistrement représentant un nombre complexe.

Algorithme

```
Type Complexe = Enregistrement
        r : réel
        im : réel
Fin
```

Dictionnaire

```
complexe = {
        "r" : float,
        "im": float
}
```

```
complexe = dict(
        r = float,
        im = float
)
```

Classes

```
class complexe :
        r : float
        im: float
```


Déclaration d'un enregistrement

Exemple :

Déclarer une structure Personne qui contient les champs : nom , prenom et Date de Naissance (composée du jour, mois, année)

```
type
 DateN = enregistrement
    j: entier
    m: entier
    a: entier
fin
 Personne = enregistrement
    nom: chaine de caractères
    prenom: chaine de caractères
   date_naissance: DateN
fin
```


Déclaration d'un enregistrement

Exemple :

Déclarer une structure Personne qui contient les champs : nom , prenom et Date de Naissance (composée du jour, mois, année)

```
Personne=dict(
        nom=str,
        prenom=str,
        date_naissance=dict( j=int, mois=int, annee=int)
)
```

```
Personne= {
        "nom":str,
        "prenom": str,
        "date_naissance": {" j":int, "mois":int, "annee:int}
}
```

Déclaration des variables structurées

Une fois le type de l'enregistrement déclaré, il est possible de déclarer des variables enregistrement portant le type déclaré. La déclaration se fait de la même manière que la déclaration d'une variable de type prédéfini.

```
Variables nom_variable : nom_enregistrement
```

```
nom_variable : nom_enregistrement
```

17

Déclaration des variables structurées

Exemple:

```
Type Personne = enregistrement
    nom: chaine de caractères
    prenom: chaine de caractères
    dnais: DateN
Fin
```

```
Type Complexe = Enregistrement
        r : réel
        im : réel
Fin
```

❑ la déclaration des variables de type structurée Personne et Complexe:

```
Variables p : Personne
          c : Complexe
```

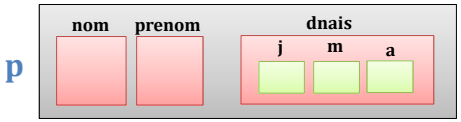
```
p : Personne      ou    p =Personne
c : Complexe      ou    c = Complexe
```

18

Manipulation des enregistrements

La manipulation d'un enregistrement se fait via ses **champs**. Les enregistrements sont composés de plusieurs zones destinées à stocker les valeurs de chaque champ.

Ainsi, la variable p de type *personne* déclarée précédemment peut être représentée comme suit :



19

Manipulation des enregistrements

Accès au champ d'un enregistrement

❑ Les champs d'un enregistrement sont **accessibles** à travers leur **nom**, grâce à l'opérateur ' . ' Un tel champ est défini par le nom de l'enregistrement ainsi que par son nom propre.

❑ Variable.Champ représente la valeur mémorisée dans le champ de l'enregistrement.

Exemple :

```
Variables p : Personne
    p. nom
    p. prenom
    p. dnais.j
    p. dnais.m
    p. dnais.a
```

❑ Pour accéder au nom de la variable p, on utilise l'expression : p.nom

❑ la lecture d'une telle expression se fait de droit à gauche : le nom du personne p.

❑ Pour accéder à l'année de naissance d'une personne p, il faut utiliser deux fois l'opérateur ' . ' :

p.dnais.a : l'année de la date de naissance de la personne p.

20

Manipulation des enregistrements

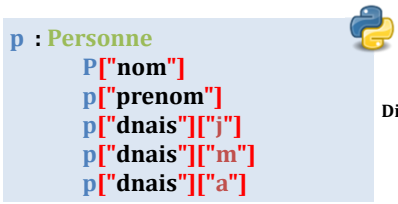
Accès au champ d'un enregistrement

Remarques:

- **Personne.pernom** : représente le champ **prenom** de l'enregistrement **personne**.
- Le nom d'un champ est toujours précédé du nom de l'enregistrement auquel il appartient.
- On ne peut pas trouver un nom de champ tout seul, sans indication de l'enregistrement.
- Le champ d'une variable enregistrement peut être lui-même un enregistrement.
- Il est possible aussi qu'un champ de type structuré soit de type tableau.
- Les champs d'un enregistrement, tout comme les éléments d'un tableau, sont des variables à qui on peut faire subir les mêmes opérations (affectation, saisie, affichage,…).

Manipulation des enregistrements

Accès au champ d'un enregistrement



Dictionnaire

Classes

Manipulation des enregistrements

Affectation

- **L'affectation** de valeurs aux différents champs d'une variable enregistrement se fait comme suit :

```
Variable . champ ← valeur
```

Exemple :

```
Variables  p : Personne
p. Nom ← "Ahmadi"
p. Prenom ← "Mouad"
p. dnais.j ← 24
p. dnais.m ← 3
p. dnais.a ← 1990
```


Manipulation des enregistrements

Affectation

- **L'affectation** de valeurs aux différents champs d'une variable enregistrement en python se fait comme suit :

```
Variable ["champ "]= valeur
```

Exemple :

```
p : Personne
        P["nom"]="Ahmadi"
        p[prenom"]="Mouad"
        p["dnais"]["j"]=24
        p["dnais"]["m"]=3
        p["dnais"]["a"]=1990
```

Manipulation des enregistrements

Lecture

❑ **La lecture** des valeurs saisies par l'utilisateur, des différents champs d'une variable enregistrement se fait comme suit :

```
Lire (Variable . champ )
```

Exemple :

```
Variables   p  : Personne
Lire ( p. Nom )
Lire ( p. Prenom )
Lire ( p. dnais.j )
Lire ( p. dnais.m )
Lire ( p. dnais.a )
```


Manipulation des enregistrements

Lecture

❑ **La lecture** des valeurs saisies par l'utilisateur, des différents champs d'une variable enregistrement en python se fait comme suit :

```
Variable ["champ "]= input( )
```

Exemple :

```
p : Personne
        P["nom"]=input(" saisir nom :")
        p[prenom"]=input(" saisir prénom :")
        p["dnais"]["j"]=int(input("saisir jour de naissance" )
        p["dnais"]["m"]=int(input("saisir mois de naissance" )
        p["dnais"]["a"]=int(input("saisir année de naissance"  )
```


Manipulation des enregistrements

Ecriture

❑ L'affichage des valeurs des différents champs d'une variable enregistrement se fait comme suit :

```
Ecrire (Variable . champ )
```

Exemple :

```
Variables   p  : Personne

Ecrire ( p. Nom )
Ecrire ( p. Prenom )
Ecrire ( p. dnais.j )
Ecrire ( p. dnais.m )
Ecrire ( p. dnais.a )
```


Manipulation des enregistrements

Ecriture

❑ L'affichage des valeurs des différents champs d'une variable enregistrement en python se fait comme suit :

```
print (Variable ["champ"] )
```

Exemple :

```
p  : Personne
print( p["nom"] )
print( p[prenom"] )
print( p["dnais"]["j"] )
print( p["dnais"]["m"] )
print( p["dnais"]["a"] )
```

Manipulation des enregistrements

Remarque:

- Il est possible d'affecter une variable enregistrement dans une autre à condition qu'ils aient la même structure.
- Tous les champs de la variable enregistrement à affecter seront recopies dans les champs de l'autre.

Exemple :

```
Variables  p,q  : Personne
    p. Nom ← "Ahmadi"
    p. Prenom ← "Mouad"
    p. dnais.j ← 24
    p. dnais.m ← 3
    p. dnais.a ← 1990
   p ← q
```

29

Passage d'un enregistrement en paramètre

Il est possible de passer tout un enregistrement en **paramètre** d'une **fonction** ou d'une **procédure** (on n'est pas obligé de passer tous les champs uns à uns, ce qui permet de diminuer le nombre de paramètres à passer), exactement comme pour les tableaux.

Exemple :

```
Procédure Proc_afficher (a : article)
Début
    Afficher (a.ref, a.nom, a.prix)
Fin
```

30

Exercice d'application

- Soit la structure Etudiant constituée par :
  - code : Entier
  - nom : Chaîne
  - prénom : Chaîne
  - genre : Caractère
  - moyenne: Réel

1. Ecrire une procédure qui permet de remplir les champs de cet enregistrement.
2. Ecrire une procédure qui affiche ces champs.
3. Ecrire l'algorithme principal
4. Ecrire le programme Python de ce problème.

Exercice d'application

```
Algorithme FicheEtudiant
type etudiant=enregistrement
            code : Entier
            nom : Chaîne
            prenom : Chaîne
            genre : Caractère
            moyenne: Réel
fin
Procédure saisireet(var  e:etudiant)
début
  ecrire("Saisir code:")
  lire(e.code)
  ecrire("Saisir nom:")
  lire(e.nom)
  ecrire("Saisir prenom:")
  lire(e.prenom)
  ecrire("Saisir genre F ou M:")
  lire(e.genre)
  ecrire("Saisir moyenne:")
  lire(e.moyenne)
fin
```

```
Procedure afficheret(e:etudiant)
début
  ecrire("Code:",e.code)
  ecrire("Nom:",e.nom)
  ecrire("Prenom",e.prenom)
  ecrire("Genre:",e.genre)
  ecrire("Moyenne:",e.moyenne)
fin
Variables  e:etudiant
début
   saisir(e)
   afficher(e)
 fin
```

Exercice d'application

```
Etudiant =dict(
   code = int, nom = str, prenom=str, genre=str,moyenne=float )
def saisir(e:Etudiant):
   e["code"]=int(input("Entrer le numéro de l'élève:"))
   e["nom"]=input("Entrer son nom : ")
   e["prenom"]=input("Entrer son prénom : ")
   e["genre"]=input("Entrer son genre F ou M : ")[0]
   e["moyenne"]=float(input("Entrer sa moyenne : "))
def afficher(e:Etudiant):
   print("Code : ", e["code"])
   print("Nom : ", e["nom"])
   print("Prénom : ", e["prenom"])
   print("Genre : ", e["genre"])
   print("Moyenne : ", e["moyenne"])
e=Etudiant
print("Affichage:")
saisir(e)
afficher(e)
```


Activité

Un établissement scolaire organise les informations concernant ses classes dans une liste identique à la suivante :

N°	code	Nom	Prénom	Moyenne	Observation
1	E5022	Alim	Ahmed	12	Néant
2	E1478	Marzi	Loubna	14	Redoublante
...	...	...	...	...	...
...	...	...	...	...	...
50	J4578	Razi	Amine	10	Dispensé du sport

Le directeur de l'établissement veut créer un programme permettant la saisie et le traitement de ces listes sachant que chaque classe comporte au maximum 30 élèves.

1. Donnez la structure de données nécessaire pour les objets à utiliser.
2. Donnez une déclaration algorithmique de ces objets.

34

Activité

```
Algorithme Etablissement
Type Eleve= Enregistrement
    Numero : entier
    Code : Chaine de caractère
    Nom : Chaine de caractère
    Prenom : Chaine de caractère
    Moyenne : réel
    Obesrvation : Chaine de caractère
Fin
Variables
    E1 : Eleve
Début
        Ecrire("Donner les informations de l'étudiant ")
        Ecrire("Numéro: ")
        Lire(E1.Numero)
        Ecrire(" Code: ")
        Lire(E1.Code)
        Ecrire(" Nom: ")
        Lire(E1.Nom)
        ...
Fin
```

30 élèves???

Activité

Il arrive souvent qu'on veut traiter non pas un seul enregistrement mais plusieurs.

30 élèves???

```
Variables
    E1,...,E30 : Eleve
```

36

Tableaux et enregistrements

- Un tableau ne peut grouper ou contenir que des éléments de même type, et puisque les éléments d'un enregistrement sont de même type qui est celui de l'enregistrement, donc on peut utiliser un tableau ou un vecteur d'enregistrements.

```
Variables
    T : Tableau [30] de Eleve
```

- Chaque élément du tableau est une variable structurée, contenant plusieurs variables de type différent. On accède à une variable structurée par son indice dans le tableau :

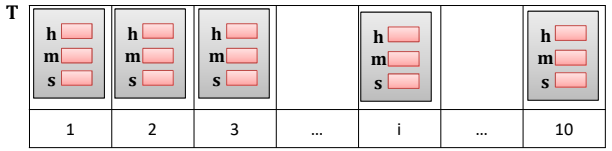
```
T[2]        // représente le deuxième élève
T[2].nom    // représente le nom du deuxième élève
```


Tableaux et enregistrements

Exemple :

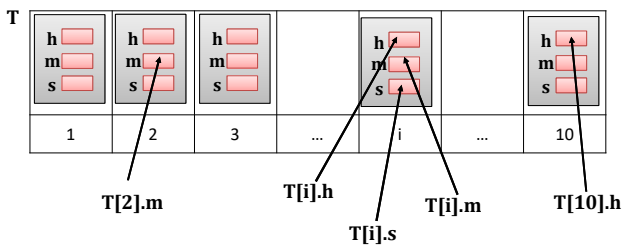
```
Type Temps= Structure
    h : entier
    m : entier
    s : entier
Fin
Variable  T : Tableau [10] de Temps
```

- On vient de déclaré un vecteur de 10 éléments de type '**Temps**' qui peut contenir par exemple les temps de parcourt de 10 athlètes.



Tableaux et enregistrements

Exemple :



Exercice d'application

- Soit la structure Etudiant constituée par :
  - code : Entier
  - nom : Chaîne
  - prénom : Chaîne
  - genre : Caractère
  - moyenne : Réel

Ecrire un algorithme dans lequel vous allez :

1- Définir la structure Etudiant.
2- Saisir les informations de n étudiants donnés (n≤100).
3- Afficher les noms des étudiants ayant une moyenne ≥10.
4- Afficher les informations de l'étudiant qui a la moyenne maximale.
5- Ecrire le programme Python de ce problème.

Exercice d'application

```
Algorithme FicheEtudiant
type etudiant=enregistrement
   ...
fin
Procédure saisiret(var e:etudiant)
...
Procedure afficheret(e:etudiant)
...
Variables  T:Tableau [100] etudiant
           i, imax, n: entier
           max: réel
début
  ecrire("donner le nbr des étudiants :")
  lire(n)
  ecrire("la saisie :")
  pour i de 1 à n faire
     ecrire("Etudiant N°", i ,":")
     saisiret(T[i])
  fin pour
   ecrire(" les étudiants ayant une moyenne
supérieur à 10 sont :" )
```

```
pour i de 1 à n faire
   si T[i].moyenne>=10 alors
        ecrire(T[i].nom)
   fin si
 fin pour
 max <- T[1].moyenne
  imax <- 1
pour i de 2 à n faire
   si T[i].moyenne>=max alors
      max <- T[i].moyenne
      imax <- i
   fin si
fin pour
afficheret(T[imax])
fin
```


Exercice d'application

```
from numpy import array
Etudiant =dict(
   code = int, nom = str, prenom=str, genre=str,moyenne=float)
def saisir():
   e={}
   e["code"]=int(input("Entrer le numéro de l'élève:"))
   e["nom"]=input("Entrer son nom : ")
   e["prenom"]=input("Entrer son prénom : ")
   e["genre"]=input("Entrer son genre F ou M : ")[0]
   e["moyenne"]=float(input("Entrer sa moyenne : "))
   return e
def afficher(e):
   print("Code : ", e["code"])
   print("Nom : ", e["nom"])
   print("Prénom : ", e["prenom"])
   print("Genre : ", e["genre"])
   print("Moyenne : ", e["moyenne"])
n= int(input("Donner le nombre des etudiants:"))
T=array([Etudiant]*n)
```


Exercice d'application

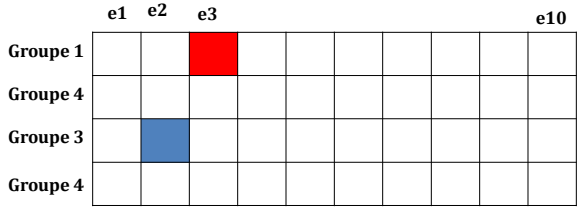
```
...
for i in range(0,n):
   print("Etudiant N°", i+1 ,":")
   T[i]=saisir( )
print (" les étudiants ayant une moyenne supérieur à 10 sont : " )
for i in range(0,n):
if T[i]["moyenne"]>=10:
      print(T[i]["nom"])
maximum=T[0]["moyenne"]
imax=0
for i in range (1,n):
   if T[i]["moyenne"]>=maximum:
      maximum=T[i]["moyenne"]
      imax=i
afficher(T[imax])
```


Tableaux et enregistrements

- On peut utiliser un tableau de deux dimensions ou une **matrice** d'enregistrements.

```
Variables
    groupe : Tableau [4][10] de Eleve
```

groupe[2] [3] *// représente le troisième élève du groupe 2*
groupe[2][3].nom *// représente le nom du troisième élève du groupe 2*
groupe[3] [2] *// représente le deuxième élève du groupe 3*



44

Exemple

```
from numpy import array
def saisiretudiant():
   e={}
   e["nom"]=input("Entrer son nom: ")
   e["moyenne"]=float(input("Entrer sa moyenne: "))
   return e
def afficheretudiant(e):
   print("Nom: ",e["nom"])
   print("Moyenne: ",e["moyenne"])
n=int(input("Donner le nombre des classes:"))
m=int(input("Donner le nombre des etudiants par classe:"))
M=array([[{}]*n]*m)
for i in range(0,n):
   print("Classe ",i+1,":")
   for j in range(0,m):
      print("Eudiant N°:",j+1)
      M[i,j]=saisiretudiant()
print("Affichage:")
for i in range(0,n):
   print("Classe ",i+1,":")
   for j in range(0,m):
      print("Eudiant N°:",j+1)
      afficheretudiant(M[i,j])
```


- Les informations utilisées dans tous les programmes que nous avons déjà écris ne pouvaient provenir que de deux sources : soit elles étaient incluses dans le programme lui-même, soit elles étaient entrées ou saisies en cours de route par l'utilisateur.
- Après la fin du programme, ces informations sont perdues. Si nous exécutons de nouveau le programme, il faut réintroduire les mêmes ou d'autres informations.
- D'autre part, toutes ces informations ont un point commun : elles résident toutes dans la mémoire principale de l'ordinateur. Ceci signifie que l'effacement (volontaire ou non!) de la mémoire provoque la destruction de ces informations, ainsi que celles utilisées dans le programme "PYTHON".
- Il est parfois nécessaire de conserver certaines données après la fin de l'exécution du programme, pour une sauvegarde d'archives ou en prévision d'une utilisation future.

46

Définition

- **Un fichier (file)** est un ensemble structuré de données stocké en général sur un support externe (disquette, disque dur ou optique, ...). Ils servent à stocker des informations de manière permanente, entre deux exécutions d'un programme.

47

Types de fichiers

Le critère important qui différencie les fichiers est la façon dont les informations sont organisées sur ces derniers.
Il existe deux catégories de fichiers : **les fichiers binaires** et **fichiers textes** :

- **Un fichier texte** est formé de caractères ASCII, organisé en lignes, chacune se termine par un caractère de contrôle de fin de ligne. Si chaque ligne contient le même genre d'informations, les lignes sont appelées des enregistrements. Les fichiers texte peuvent être crées avec des éditeurs de texte et affichés de manière lisible à l'écran
  - Par exemple, prenons le cas d'un carnet d'adresses, le fichier est destiné à stoker les coordonnées d'un certain nombre de personnes. Pour chacune, il faudra noter le : nom, prénom, adresse et numéro de téléphone de chaque personne. Dans ce cas, les informations concernant une personne donnée doivent être stockées sur une seule ligne du fichier.

48

Types de fichiers

Le critère important qui différencie les fichiers est la façon dont les informations sont organisées sur ces derniers.
Il existe deux catégories de fichiers : **les fichiers binaires** et **fichiers textes** :

- **Un fichier binaire** contient des données non textuelles. Il n'est pas organisé sous forme d'enregistrement. Les fichiers binaires ne prennent sens que s'ils sont traités par un programme adapté. Par exemple un fichier son, une vidéo, une image, un programme exécutable, etc.
  Dans les fichiers binaires, les données sont écrites à l'image exacte de leur codage en mémoire. Ceci facilite l'accès à ce type de fichier et le rend rapide.

Types d'accès aux fichiers

Le type d'accès est la technique que la machine doit suivre pour aller chercher les informations contenues dans un fichier.
On distingue trois types d'accès aux fichiers:

- **L'accès séquentiel:** Cet accès consiste à traiter les informations séquentiellement, c'est à dire dans l'ordre où elles apparaissent dans le fichier. On ne peut donc accéder à une information qu'en ayant au préalable examiné celle qui la précède. Dans le cas d'un fichier texte, cela signifie qu'on lit le fichier ligne par ligne (enregistrement par enregistrement).
- **L'accès direct (ou aléatoire):** Ce type d'accès consiste à se placer directement sur l'information souhaitée sans parcourir celles qui la précèdent, en précisant la position de l'élément recherché. L'indication d'un numéro permet donc un accès direct et rapide à l'information ainsi référencée.
- **L'accès indexé :** Ce type d'accès combine la rapidité de l'accès direct et la simplicité de l'accès séquentiel. Il est particulièrement adapté au traitement des gros fichiers, comme les bases de données.

Traitement séquentiel des fichiers texte

Ouvrir et fermer un fichier

Ouvrir un fichier texte

Lorsqu'on désire accéder à un fichier, il est nécessaire avant tout accès, d'ouvrir le fichier.

Syntaxe :

Ouvrir ("Nom_du_Fichier" ,Num_canal , "Mode")

Nom_du_Fichier : c'est le nom physique du fichier.
Num_canal : c'est le nom logique du fichier. Pour ouvrir un fichier, il faut lui allouer un numéro du canal valide et disponible.
Mode : le mode d'ouverture du fichier conditionne le travail qui peut être effectué sur ses enregistrements. Il existe trois modes d'ouverture du fichier texte :

- **Lecture :** permet d'ouvrir le fichier en lecture seul
- **Ecriture :** indique son accès en écriture. Dans ce mode, un nouveau fichier est toujours créé. Si le fichier existe déjà, il est réinitialisé à vide et son contenu précédent est perdu.
- **Ajout :** permet d'ajouter des données à un fichier séquentiel existant en conservant le contenu précédent.

Traitement séquentiel des fichiers texte

Ouvrir et fermer un fichier

Ouvrir un fichier texte

Exemple :

OUVRIR ("fiche.txt" , 4 , "Lecture")
OUVRIR ("fiche.txt" , 4 , " Ecriture")

Remarques:
1- On peut donner le chemin du fichier.
Exemple: OUVRIR ("C:\Documents\fich1.txt" ,1, "LECTURE")
2- Le N° de canal doit être unique. Ainsi, si plusieurs fichiers doivent être manipulés par le même programme, choisissez des références différentes.
3- Dans un fichier ouvert pour ajout, les enregistrements seront stockés à la fin du fichier.
4- Dans un fichier ouvert pour Ecriture, les enregistrements seront écrit au début du fichier.

Traitement séquentiel des fichiers texte

Ouvrir et fermer un fichier

Fermer un fichier texte

Une fois qu'on a terminé avec un fichier, il ne faut pas oublier de le fermer.
On libère ainsi le canal qu'il occupait.

Syntaxe :

Fermer(**Nom_du_Fichier**)
Ou bien
Fermer (Num_canal)

Note :
Lorsqu'un fichier doit subir plusieurs interventions nécessitant plusieurs ouvertures, il sera nécessaire de fermer le fichier avant de le re-ouvrir.

Traitement séquentiel des fichiers texte

Lire et écrire dans un fichier

Lecture d'un fichier

Syntaxe :

LireFichier (Num_canal,**nomVariable**)

L'instruction **LireFichier** récupère dans la variable spécifiée l'enregistrement suivant dans le fichier ("suivant", par rapport au dernier enregistrement lu): C'est en cela que le fichier est dit séquentiel.
Lire un fichier séquentiel de bout en bout suppose de programmer une boucle. Si l'on veut stocker au fur et à mesure en mémoire vive les informations lues dans le fichier, on a recours à des tableaux.

Exemple:

```
Variables  ch : chaine
Début
   OUVRIR ("fiche.txt",4,"Lecture")
   LireFichier  (4, ch)
   ....
Fin
```


Traitement séquentiel des fichiers texte

Lire et écrire dans un fichier

Ecriture dans un fichier

Syntaxe :

EcrireFichier (Num_canal,**nomVariable**)

- numCanal : numéro désignant le fichier
- nomVariable : nom de la variable contenant la valeur à écrire dans le fichier.

Exemple:

```
Variables  ch : chaine
Début
   OUVRIR ("fiche.txt",4,"Ecriture")
   ch ← "Bonjour"
   EcrireFichier  (4, ch)
   ....
Fin
```


Traitement séquentiel des fichiers texte

Fin de fichiers

Comme on sait rarement à l'avance combien d'enregistrements comporte le fichier, on utilise alors la fonction EOF (acronyme pour End Of File). Cette fonction renvoie la valeur Vrai si on a atteint la fin du fichier.

Syntaxe :

EOF (Num_canal)

Exemple:

```
Variables  ch : chaine
Début
   OUVRIR ("fiche.txt",4, "Lecure")
   Tantque Non EOF(4)
     LireFichier(4, ch)
   FinTantque
....
```

Traitement séquentiel des fichiers texte

Insérer/ Modifier/ Supprimer un enregistrement :

Vu qu'un fichier ne peut être ouvert que dans l'un des modes d'ouverture déjà cités, donc pour modifier son contenu il faut passer par un fichier intermédiaire où on va copier le contenu du fichier original en le modifiant et puis on supprime ce fichier et on renomme le fichier intermédiaire.

- ❑ Pour supprimer un fichier on utilise une fonction supprimer :
  supprimer (NomFichier)
- ❑ Pour renommer un fichier on utilise une fonction renommer :
  renommer(AncienNom, NouvNom)

Les fonctions (Modules) applicables sur les fichiers Texte :

Algorithmique	Rôle	Python
Ouvrir("Chemin\ Nom_physique", Nom_logique , "Mode") **Exp:** **Ouvrir**("fich.txt", F , "wt")	**Ouverture d'un fichier** • Mode d'ouverture : ○ "**r**" : Lecture ○ "**w**" : Ecriture (création) ○ "**a**" : Ecriture à la fin du fichier	**F= open**("fich.txt","wt")
Lire (Nom_logique, ch) **Exp: Lire** (F, ch)	Lecture de la totalité d'un fichier	var= f.**read**()
Lire_ligne (Nom_logique, ch) **Exp:** **Lire_ligne** (F, ch)	Lecture d'une ligne depuis un fichier texte	var = f.**readline**()
Ecrire (Nom_logique, ch) **Exp: Ecrire** (F, ch)	Ecriture dans un fichier texte	f.**write**(Objet+"\n")
Fin_fichier (Nom_logique) **Exp:** **Fin_fichier** (F)	Retourne **Vrai** si le pointeur est à la fin du fichier sinon elle retourne **Faux**	ok=True while **ok** : var= f.readline() if var == '' : ok=False
Fermer (Nom_logique) **Exp: Fermer** (F)	Fermeture d'un fichier	f. .**close**()

Exercice

1. Ecrire un programme en algorithme et en python qui per permet de lire le nom, le prénom, la classe et la note de cinq étudiants et les enregistrer dans un fichier texte nommé notes.txt dans le répertoire courant.
2. Ecrire un programme python qui permet d'afficher le contenu de fichier notes.txt.

Solution

```
procédure remplir(@f :texte ; n :entier)
début
      ouvrir("notes.txt",f,"Ecriture")
      pour i de 1 à n faire
          écrire("eleve",i+1,":")
          écrire ("le nom:")
          lire(nom)
          écrire ("le prenom:")
          lire(prénom)
          écrire ("la classe:")
          lire(classe)
          écrire ("la note :")
          lire(note)
      fin pour
écrire (f, nom+«  "+prenom+" "+classe+" " +note)
      fermer(f)
fin
```

```
Procédure afficher(f :texte):
début
      ouvrir("notes.txt",f,"Lecture")
      écrire("nom prenom classe note")
      lire(f,h)
      ecrire(h)
      fermer(f)
fin
```

Solution

```
def remplir(n):
    f=open("notes.txt","wt")
    for i in range(n):
        print("eleve",i+1,":")
        nom=input("le nom:")
        prenom=input("le prenom:")
        classe=input("la classe:")
        note=float(input("la note de:"))
        f.write(nom+" "+prenom+" "+classe+" "+str(note)+"\n")
        f.close()
        return f
n=5
f=remplir(n)
```

```
def afficher():
    f=open("notes.txt","rt")
    print("nom prenom classe note")
    h=f.read ()
    print(h)
    f.close()
afficher()
```


Solution

```
def afficher2():
    f=open("notes.txt","rt")
    print("nom prenom classe note")
    h=f.readline()
    while h!="":
        print(h)
        h=f.readline()
    f.close()
afficher()
```

```
def afficher3():
    f=open("notes.txt","rt")
    print("nom prenom classe note")
    for h in f:
        print(h)
    f.close()
afficher()
```

62