

Faculté des Sciences

كلية العلوم

Département d'Informatique

Filière MIP - Semestre 2

2024-2025

Module : Informatique II ( Algorithmique II / Python )

Chapitre 4

Les enregistrements  
et les fichiers

Pr. Khadija Louzaoui

Les enregistrements

Chapitre 4

Types personnalisés

Types personnalisés

❑ Les types de données fondamentales:

Entier, réel, caractère, chaîne 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

3

Chapitre 4

Types personnalisés

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= chaîne de caractères[50]

Variables

ch : nom {ch est une chaîne de 50 caractères}

T : Tableau[100] de nom {T est un Tableau de 100 chaînes de 50 caractères chacun}

4

Pr. Khadija LOUZAOU

1

Chapitre 4

Les enregistrements

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.

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

5

Chapitre 4

Les enregistrements

Activité

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

Num	1	2						30
Code	1	2						30
Nom	1	2						30
Prénom	1	2						30
Moyenne	1	2						30
Observation	1	2						30

6

Chapitre 4

Les enregistrements

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 chaîne de caractères

**Variables**

- Numero : Tableau[ 30] de entier
- Code : Tableau[30] de Tab
- Nom : Tableau[30] de Tab
- Prénom : Tableau[30] de Tab
- Moyenne : Tableau[ 30] de réel
- Observation : Tableau[ 30] de Tab

7

Chapitre 4

Les enregistrements

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** ou **DICTIONNAIRE** en Python) qui permet de réaliser cette tâche.

8

Chapitre 4

Les enregistrements

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 « <b>Date</b> » est composée du jour, mois, et année.	<b>Date</b>	<div>j</div> <div>m</div> <div>a</div>
La structure « <b>Temps</b> » est composée du heure, minute, et second.	<b>Temps</b>	<div>h</div> <div>m</div> <div>s</div>
La structure « <b>Etudiant</b> » est composée du nom, prénom, âge, et moyenne.	<b>Etudiant</b>	<div>nom</div> <div>prénom</div> <div>âge</div> <div>moy</div>

9

Chapitre 4

Les enregistrements

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

Chapitre 4

Les enregistrements

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

FinEnregistrement

Type Nom\_structure = Structure

Nom Champ 1 : Type Champ 1

Nom Champ 2 : Type Champ 2

...

Nom Champ n : Type Champ n

FinStructure

11

Chapitre 4

Les enregistrements

Déclaration d'un enregistrement

Exemple 1:

Déclarer une structure Etudiant qui contient les champs : code, nom, prenom et age.

Type Etudiant= enregistrement

code: entier

nom: chaîne de caractères

prenom: chaîne de caractères

age: entier

Fin

12

Chapitre 4

Les enregistrements

Déclaration d'un enregistrement

Exemple 2:

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: chaîne de caractères  
prenom: chaîne de caractères  
dnais: DateN  
Fin

13

Chapitre 4

Les enregistrements

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

Exemples:

Type Personne = enregistrement  
nom: chaîne de caractères  
prenom: chaîne 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

14

Chapitre 4

Les enregistrements

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.

Exemples:

1. La variable C de type Complexe déclarée précédemment peut être représentée comme suit :

C

r

im

2. La variable p de type personne déclarée précédemment peut être représentée comme suit :

p

nom

prenom

dnais

j

m

a

15

Chapitre 4

Les enregistrements

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.

Exemples :

Variables C : Complexe  
c.r  
c.im

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.

16

Pr. Khadija LOUZAOU

4

Chapitre 4

Les enregistrements

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,...).

17

Chapitre 4

Les enregistrements

Manipulation des enregistrements

Affectation

❑

L'affectation

de valeurs aux différents champs d'une variable enregistrement se fait comme suit :

Variable . champ

←

valeur

Exemples :

Variables c : Complexe

c.r

←

4

c.im

←

-7

Variables p : Personne

p.Nom

←

"Ahmadi"

p.Prenom

←

"Mouad"

p.dnais.j

←

24

p.dnais.m

←

3

p.dnais.a

←

1990

18

Chapitre 4

Les enregistrements

Manipulation des enregistrements

Affectation

❑

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

q

←

p

19

Chapitre 4

Les enregistrements

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 )

Exemples :

Variables c : Complexe

Lire ( c.r )

Lire ( c.im )

Variables p : Personne

Lire ( p.Nom )

Lire ( p.Prenom )

Lire ( p.dnais.j )

Lire ( p.dnais.m )

Lire ( p.dnais.a )

20

Chapitre 4

Les enregistrements

Manipulation des enregistrements

Ecriture

❑ L'affichage

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

Ecrire (Variable . champ )

Exemples :

Variables c : Complexe

Ecrire ( c . r )  
Ecrire ( c . im )

Variables p : Personne

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

21

Chapitre 4

Les enregistrements

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

afficher\_personne (p : **personne**)  
Début  
Ecrire ( p . Nom )  
Ecrire ( p . Prenom )  
Ecrire ( p . dnais.j )  
Ecrire ( p . dnais.m )  
Ecrire ( p . dnais.a )  
Fin

22

Chapitre 4

Les enregistrements

Exercice d'application

❑ Soit l'enregistrement Etudiant constituée par :

code : Entier  
nom : Chaîne  
prénom : Chaîne  
genre : Caractère  
moyenne: Réel

1. Déclarer l'enregistrement.

2. Ecrire une procédure qui permet de remplir les champs de cet enregistrement.

3. Ecrire une procédure qui affiche ces champs.

4. Ecrire l'algorithme principal.

Chapitre 4

Les enregistrements

Exercice d'application ( Solution )

1. Déclaration de l'enregistrement

type Etudiant=enregistrement

code : Entier  
nom : Chaîne  
prenom : Chaîne  
genre : Caractère  
moyenne: Réel

FinEnregistrement

Pr. Khadija LOUZAOU

6

## 28

Chapitre 4

Les enregistrements

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 Etudiant

❑ 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 :

Exemple :

T[2] // représente le deuxième étudiant

T[2].nom // représente le nom du deuxième étudiant

29

Chapitre 4

Les enregistrements

Tableaux et enregistrements

Exemple :

Type Temps= Enregistrement

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.

T

h

m

s

1

h

m

s

2

h

m

s

3

...

h

m

s

i

...

h

m

s

10

30

Chapitre 4

Les enregistrements

Tableaux et enregistrements

Exemple :

T

h

m

s

1

h

m

s

2

h

m

s

3

...

h

m

s

i

...

h

m

s

10

T[2].m

T[i].h

T[i].s

T[i].m

T[10].h

31

Chapitre 4

Les enregistrements en Python

Tableaux et enregistrements

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

Variables

groupe : Tableau [4][10] de Etudiant

groupe[2] [3] // représente le troisième étudiant du groupe 2

groupe[2][3].nom // représente le nom du troisième étudiant du groupe 2

groupe[3] [2] // représente le deuxième étudiant du groupe 3

e1

e2

e3

e10

Groupe 1

Groupe 4

Groupe 3

Groupe 4

32

Pr. Khadija LOUZAOU

8



Chapitre 4

Les enregistrements

Exercice d'application

❑

Soit l'enregistrement Etudiant constituée par :

code : Entier

nom : Chaîne

prénom : Chaîne

genre : Caractère

moyenne: Réel

1. Déclarer l'enregistrement.

2. Ecrire une procédure qui permet de remplir les champs de cet enregistrement.

3. Ecrire une procédure qui affiche ces champs.

4. Ecrire un algorithme principal dans lequel vous allez :

a- Saisir les informations de n étudiants donnés (n≤100).

b- Afficher les noms des étudiants ayant une moyenne ≥10.

c- Afficher les informations de l'étudiant qui a la moyenne maximale.

Chapitre 4

Les enregistrements

Exercice d'application ( Solution )

4. Algorithme principal

a - Saisir les informations de n étudiants donnés (n≤100).

Algorithme Principal

-----1-----

-----2-----

-----3-----

Variables

T:Tableau [100] Etudiant

i, imax, n: entier

max: réel

début

Ecrire("Entrez le nombre d'étudiants (max 100) : ")

Lire(n)

Ecrire("la saisie :")

Pour i de 1 à n faire

Ecrire("Etudiant N°", i, ":")

saisir\_etudiant(T[i])

Fin pour

Chapitre 4

Les enregistrements

Exercice d'application ( Solution )

4. Algorithme principal

b - Afficher les noms des étudiants ayant une moyenne ≥10.

Algorithme Principal

-----1-----

-----2-----

-----3-----

Variables

T:Tableau [100] Etudiant

i, imax, n: entier

max: réel

début

-----

Ecrire(" Les étudiants ayant une moyenne >= 10 sont :")

Pour i de 1 à n faire

Si T[i].moyenne>=10 alors

Ecrire(T[i].nom)

Fin si

Fin pour

Chapitre 4

Les enregistrements

Exercice d'application ( Solution )

4. Algorithme principal

c - Afficher les informations de l'étudiant qui a la moyenne maximale ( un seul ).

Algorithme Principal

-----1-----

-----2-----

-----3-----

Variables

T:Tableau [100] Etudiant

i, imax, n: entier

max: réel

début

-----

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

Ecrire("l'étudiant ayant la moyenne maximale est :")

afficher\_etudiant(T[imax])

Fin

Chapitre 4Les enregistrements en Python

Les Enregistrements en Python

Un **enregistrement** (ou **record**) est une structure de données qui regroupe plusieurs champs (ou attributs) de types potentiellement différents, liés entre eux.

En Python, on utilise souvent les **dictionnaires** ou les **classes** pour modéliser des enregistrements.


37

Chapitre 4Les enregistrements en Python

Les classes

Une **classe** en Python est un modèle permettant de créer des **objets** qui regroupent des données (**attributs**) et des fonctions (**méthodes**). Elle est utilisée pour structurer des programmes selon le paradigme de la **programmation orientée objet (POO)**.

Exemple d'une classe



```
class Personne:
    def __init__(self, nom, age, profession):
        self.nom = nom
        self.age = age
        self.profession = profession

p = Personne('Ahmed', 30, 'Programmeur')
print(p.nom)

>> Ahmed
```

38

Chapitre 4Les enregistrements en Python

Les classes

**Avantages :**

- Parfait pour les enregistrements **structurés et réutilisables**.
- Permet d'ajouter des **méthodes** (comportements).
- Lisible, clair, extensible.
- Meilleure pour des projets plus grands ou orientés objets.

**Inconvénients :**

- Un peu plus de code à écrire au début.

Pour une structure **claire, robuste et réutilisable** : l'utilisation des classes (**class**) est la meilleure option.


39

Chapitre 4Les enregistrements en Python

Les dictionnaires

Un **dictionnaire** en Python est une structure de données qui associe des **clés** à des **valeurs**, ce qui permet un accès rapide aux informations. Chaque élément est défini sous la forme **clé: valeur**, et l'ordre est préservé depuis Python 3.7.

Exemple:



```
Personne={
    'nom': " Ahmed ",
    'age': 30 ,
    'profession':" Programmeur "
}
```

- 'nom', 'âge', 'moyenne' sont les **clés (champs)**.
- 'Ahmed', 22, 'Programmeur' sont les **valeurs**.

40

Chapitre 4

Les enregistrements en Python

Les dictionnaires

Avantages :

- Facile et rapide à utiliser.
- Les clés sont comme les noms de champs.
- Dynamique (tu peux ajouter/supprimer des champs à tout moment).

Inconvénients :

- Pas de vérification de structure.
- Moins clair s'il y a beaucoup de champs ou de logique métier.

Pour un **petit** script ou des données **simples** : on utilise souvent les **dictionnaires (dict)**.

41

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

En Python, on peut créer un dictionnaire de deux manières :

Avec des accolades {}:

```
personne = {"nom": "Ali", "age": 25 }
print(personne["nom"])

>> Ali
```

Avec le constructeur dict() :

```
personne = dict(nom= "Ali", age= 25 )
print(personne["nom"])

>> Ali
```

- Les **clés** doivent être des **identifiants valides** (sans espaces, ni accents) si on utilise la syntaxe **dict(...)**.
- Pour des clés plus complexes (comme des chaînes avec espaces ou des variables), il vaut mieux utiliser des accolades :

```
personne = dict(nom= "Ali", age= 25 )
personne = {"nom complet": "Ali Amari", "age": 25 }
```

42

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Déclaration d'un 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
```

Exemple:

```
Personne={
    'nom' : " ",
    'age' : 0,
    'profession':" "
}
Personne['nom']=" Ahmed "
Personne['age']= 30
Personne[' profession ']="Programmeur "
Print(personne['nom'])

>> Ahmed
```

43

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Exemple :

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

```
Personne= {
    "nom": " ",
    "prenom": " ",
    "dnais": {" jour":0, "mois":0, "annee":0}
}
```

```
Personne= {
    "nom": str,
    "prenom": str,
    "dnais ": {" jour":int, "mois":int, "annee":int}
}
```

```
Personne=dict(
    nom=" ",
    prenom=" ",
    dnais=dict( jour=0, mois=0, annee=0)
)
```

44

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Déclaration des variables structurées

```
Personne= {  
    "nom": " ",  
    "prenom": " ",  
    "dnais": {" jour":0, "mois":0, "annee":0}  
}
```

Exemple :

```
nom_variable = nom_enregistrement
```

```
p = Personne
```

45

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Accès au champ d'un enregistrement

```
Personne= {  
    "nom": " ",  
    "prenom": " ",  
    "dnais": {" jour":0, "mois":0, "annee":0}  
}
```

```
personne["nom"]  
personne["prenom"]  
personne["dnais"]["jour"]  
personne["dnais"]["mois"]  
personne["dnais"]["annee"]
```

46

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Modification de champ d'un enregistrement

❑ La modification ou l'affectation de valeurs aux différents champs d'une variable enregistrement en python se fait comme suit :

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

Exemple :

```
personne["nom"]="Ahmadi"  
personne["prenom"]="Mouad"  
personne["dnais"]["jour"]=24  
personne["dnais"]["mois"]=3  
personne["dnais"]["annee"]=1990
```

47

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme 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 :

```
personne["nom"]=input(" saisir nom :")  
personne["prenom"]=input(" saisir prénom:")  
personne["dnais"]["jour"]=int(input("saisir jour de naissance:" ))  
personne["dnais"]["mois"]=int(input("saisir mois de naissance:" ))  
personne["dnais"]["annee"]=int(input("saisir année de naissance:" ))
```

48

Pr. Khadija LOUZAOU

12

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Ecriture

L'affichage

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

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

Exemple :

```
print(personne["nom"])
print(personne["prenom"])
print(personne["dnais"]["jour"])
print(personne["dnais"]["mois"])
print(personne["dnais"]["annee"])
```

49

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Ajout d'un nouveau champ

Variable["champ"] = valeur

Exemple :

```
personne["mail"] = valeur
```

Suppression d'un champ

del Variable["champ"]

Exemple :

```
Del personne["prenom"]
```

Parcourir un enregistrement

for clé, valeur in Variable.items():
 print(f"{clé} : {valeur}")

Exemple :

```
for clé, valeur in Personne.items():
    print(f"{clé} : {valeur}")
```

50

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Exemple :

```
# Déclaration de la structure de la personne
Personne = {
    "nom": "",
    "prenom": "",
    "dnais": {
        "jour": 0,
        "mois": 0,
        "annee": 0
    }
}

# Saisie des données
Personne["nom"] = input("Saisir nom : ")
Personne["prenom"] = input("Saisir prénom : ")
Personne["dnais"]["jour"] = int(input("Saisir jour de naissance : "))
Personne["dnais"]["mois"] = int(input("Saisir mois de naissance : "))
Personne["dnais"]["annee"] = int(input("Saisir année de naissance : "))

# Affichage pour vérification
print("\nInformations saisies :")
print(Personne)
Personne["mail"] = input("Saisir Mail : ")
del Personne["prenom"]
print(Personne)
for clé, valeur in Personne.items():
    print(f"{clé} : {valeur}")
```

52

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Liste d'enregistrements :

Pour stocker plusieurs enregistrements, on utilise une liste de dictionnaires :

Exemple :

```
etudiants = [
    {"nom": "Ali", "age": 20},
    {"nom": "Amina", "age": 19},
    {"nom": "Mohammed", "age": 30}
]
```

Accès à un champ particulier :

print(etudiants[1]["nom"]) >> Amina

Parcourir une liste d'enregistrements :

for etu in etudiants:
 print(etu["nom"], "a", etu["age"], "ans")

>> Ali a 20 ans
 Amina a 19 ans
 Mohammed a 30 ans

52

## Chapitre 4

## Les enregistrements en Python

### Utilisation des dictionnaires comme enregistrements

#### Liste d'enregistrements :

#### ❖ Ajout d'enregistrements à la liste

On peut demander à l'utilisateur combien de personnes il souhaite saisir, puis on boucle :

```
liste_personnes = []
n = int(input("Combien de personnes voulez-vous saisir ? "))
for i in range(n):
    print(f"\n Personne {i+1} :")
    personne = {
        "nom": input("Nom : "),
        "prenom": input("Prénom : "),
        "dnais": {
            "jour": int(input("Jour de naissance : ")),
            "mois": int(input("Mois de naissance : ")),
            "annee": int(input("Année de naissance : "))
        }
    }
    liste_personnes.append(personne) #liste_personnes.insert(0,personne)
```

54

## Chapitre 4


## Les enregistrements en Python

### Utilisation des dictionnaires comme enregistrements

#### Liste d'enregistrements :

- Parcours et affichage de la liste d'enregistrements

On peut afficher toutes les personnes comme ceci :



```
print("\n Liste des personnes :")
for p in liste_personnes:
    print(f"{p['prenom']} {p['nom']} né le
           {p['dnais']['jour']}/{p['dnais']['mois']}/{p['dnais']['annee']}")
```

```
>> Liste des personnes :
Amrani ahmed né le 12/4/2008
Milam Sara né le 14/5/1999
Mohammed Ali né le 5/5/2011
```

56

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

On peut écrire des fonctions qui reçoivent des enregistrements en paramètre, pour :

- **afficher**
- **saisir**
- **modifier**
- **analyser** les données qu'ils contiennent
- ....

57

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 1 :

- Fonction pour afficher les informations d'un seul étudiant:

```
def afficher_etudiant(e):  
    print("Nom :", e["nom"])  
    print("Prénom :", e["prenom"])  
    print("age :", e["age"])  
    print("Filière :", e["filiere"])
```

- Appel de la fonction :

```
etudiant = {"nom": "Alim",  
            "prenom": "Nora",  
            "age": 20,  
            "filiere": "MIP"}  
  
etud=etudiant  
afficher_etudiant(etud)
```

```
>> Nom : Alim  
    Prénom : Nora  
    age : 20  
    Filière : MIP
```

58

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 2 : Version 1 (modifier un dictionnaire passé en argument)

- Fonction pour saisir et ajouter les informations d'un seul étudiant:

```
def saisir_etudiant(e):  
    e["nom"]=input("Nom :")  
    e["prenom"]=input("Prénom :")  
    e["age"]=int(input("age :"))  
    e["filiere"]=input("Filière :")
```

- Appel de la fonction :

```
etudiant = {} # création du dictionnaire  
saisir_etudiant(etudiant)  
print(etudiant)
```

```
>> Nom :Alim  
    Prénom :Sara  
    Age :21  
    Filière :MIP  
    {'nom': 'Alim', 'prenom': 'Sara', 'age': 21, 'filiere': 'MIP'}
```

59

Chapitre 4

Les enregistrements en Python

Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 2 : Version 2 (retourne un dictionnaire)

- Fonction pour saisir et ajouter les informations d'un seul étudiant:

```
def saisir_etudiant():  
    etudiant = {  
        "nom": input("Nom :"),  
        "prenom": input("Prénom :"),  
        "age": int(input("Age :")),  
        "filiere": input("Filière :")  
    }  
    return etudiant
```

- Appel de la fonction :

```
e=saisir_etudiant()  
print(e)
```

```
>> Nom :Alim  
    Prénom :Sara  
    Age :21  
    Filière :MIP  
    {'nom': 'Alim', 'prenom': 'Sara', 'age': 21, 'filiere': 'MIP'}
```

60

Chapitre 4

Les enregistrements en Python


Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 2 :

- Foncton pour modifier l'âge d'un étudiant

```
def modifier_age(e, nouvel_age):  
    e["age"] = nouvel_age
```



- Appel de la fonction :

```
etudiant = {"nom": "Alim",  
            "prenom": "Nora",  
            "age": 20,  
            "filiere": "MIP"}  
etud=etudiant  
afficher_etudiant(etud)  
modifier_age(etud,15)  
afficher_etudiant(etud)
```

```
>> Nom : Alim  
    Prénom : Nora  
    age : 20  
    Filière : MIP  
    Nom : Alim  
    Prénom : Nora  
    age : 15  
    Filière : MIP
```

61

Chapitre 4

Les enregistrements en Python


Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 3 :

- Foncton pour saisir et ajouter plusieurs étudiants

```
def saisir_etudiants():  
    liste_etudiants=[]  
    n = int(input("Combien de personnes voulez-vous saisir ? "))  
    for i in range(n):  
        print(f"\n Etudiant {i+1} :")  
        etudiant = {  
            "nom": input("Nom :"),  
            "prenom": input("Prénom :"),  
            "age": int(input("age :")),  
            "filiere": input("Filière :")  
        }  
        liste_etudiants.append(etudiant)  
    return liste_etudiants
```



- Appel de la fonction :

```
liste=saisir_etudiants()
```

62

Chapitre 4

Les enregistrements en Python


Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 3 :

- Exemple d'exécution :

```
>> Combien de personnes voulez-vous saisir ? 2  
  
Etudiant 1 :  
Nom : Ali  
Prénom : Karim  
Age : 20  
Filière : MIP  
  
Etudiant 2 :  
Nom : Sara  
Prénom : Amina  
Age : 22  
Filière : MIP
```



63

Chapitre 4

Les enregistrements en Python


Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 4 : Version 1

- Fonction pour calculer la moyenne d'âge d'un groupe d'étudiants

```
def moyenne_age_groupe(liste_etudiants):  
    total = 0  
    for e in liste_etudiants:  
        total += e["age"]  
    return total / len(liste_etudiants)
```



- Appel de la fonction :

```
etudiants = [  
    {"nom": "Ali", "age": 21},  
    {"nom": "Sami", "age": 19},  
    {"nom": "Nour", "age": 22}  
]  
print("Moyenne d'âge :",  
      moyenne_age_groupe(etudiants))
```

```
>> 20.666...
```

64



Chapitre 4

Les enregistrements en Python


Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 4 : Version 2

- Fonction pour calculer la moyenne d'âge d'un groupe d'étudiants

```
def moyenne_age_groupe(liste_etudiants):  
    return sum(e["age"] for e in liste_etudiants) / len(liste_etudiants)
```



- Appel de la fonction :

```
etudiants = [  
    {"nom": "Ali", "age": 21},  
    {"nom": "Sami", "age": 19},  
    {"nom": "Nour", "age": 22}  
]  
print("Moyenne d'âge:",  
      moyenne_age_groupe(etudiants))
```

```
>> 20.666...
```

65

Chapitre 4

Les enregistrements en Python


Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 5 :

- Fonction pour filtrer les étudiants dont l'âge est supérieur ou égal à une valeur donnée

```
def filtrer_par_age(liste_etudiants, age_min):  
    return [e for e in liste_etudiants if e["age"] >= age_min]
```



- Appel de la fonction :

```
liste_etudiants = [  
    {"nom": "Ali", "age": 19},  
    {"nom": "Sami", "age": 21},  
    {"nom": "Lina", "age": 18},  
    {"nom": "Mira", "age": 22}  
]  
etudiants_20_plus = filtrer_par_age(liste_etudiants, 20)  
print(etudiants_20_plus)  
#for e in filtrer_par_age(liste_etudiants, 20):  
#    print(f"{e['nom']} ({e['age']} ans)")
```

```
>> [{"nom": 'Sami', 'age': 21}, {'nom': 'Mira', 'age': 22}]
```

66

Chapitre 4

Les enregistrements en Python


Utilisation des dictionnaires comme enregistrements

Fonctions appliquées à des enregistrements:

Exemple 6 :

- Fonction pour filtrer les étudiants par d'autres critères

```
def filtrer_filiere_age(liste_etudiants, filiere, age_min):  
    return [e for e in liste_etudiants if e["filiere"] == filiere and e["age"] >= age_min]
```



- Appel de la fonction :

```
Liste_etudiants = [  
    {"nom": "Ali", "prenom": "Nassim", "age": 21, "filiere": "Info"},  
    {"nom": "Lina", "prenom": "Amel", "age": 19, "filiere": "Math"},  
    {"nom": "Yasmine", "prenom": "Karim", "age": 22, "filiere": "Info"},  
    {"nom": "Salim", "prenom": "Rami", "age": 18, "filiere": "Info"},  
    {"nom": "Maya", "prenom": "Zahra", "age": 20, "filiere": "Math"}  
]  
resultat = filtrer_filiere_age(liste_etudiants, "Info", 20)  
for e in resultat:  
    print(f"{e['prenom']} {e['nom']} - {e['age']} ans - {e['filiere']}")
```

```
Nassim Ali - 21 ans - Info  
Karim Yasmine - 22 ans - Info
```

67

Chapitre 4

Les enregistrements en Python

Exercice d'application

❑

 Soit l'enregistrement Etudiant constituée par :

code : Entier  
nom : Chaîne  
prénom : Chaîne  
genre : Caractère  
moyenne: Réel

1. Déclarer l'enregistrement.

2. Ecrire une procédure qui permet de remplir les champs de cet enregistrement.

3. Ecrire une procédure qui affiche ces champs.

4. Ecrire un algorithme principal dans lequel vous allez :

a- Saisir les informations de n étudiants donnés (n≤100).

b- Afficher les noms des étudiants ayant une moyenne ≥10.

c- Afficher les informations de l'étudiant qui a la moyenne maximale.

5. Traduire cet algorithme en python en utilisant les dictionnaires.

Chapitre 4

Les enregistrements en Python


Exercice d'application

5. Traduire cet algorithme en python en utilisant les dictionnaires.

```
etudiant = {
    "code": 0,
    "nom": "",
    "prenom": "",
    "genre": "",
    "moyenne": 0.0
}

def saisir_etudiant():
    return {
        "code": int(input("Code : ")),
        "nom": input("Nom : "),
        "prenom": input("Prénom : "),
        "genre": input("Genre (M/F) : "),
        "moyenne": float(input("Moyenne : "))
    }

def afficher_etudiant(e):
    print(f"Code : {e['code']}, Nom : {e['nom']}, Prénom : {e['prenom']}, Genre : {e['genre']}, Moyenne : {e['moyenne']}")
```




Chapitre 4

Les enregistrements en Python

Exercice d'application

5. Traduire cet algorithme en python en utilisant les dictionnaires.

```
n = int(input("Nombre d'étudiants (n ≤ 100) : "))
while n > 100 or n < 1:
    n = int(input("Veuillez saisir un nombre entre 1 et 100 : "))
# a) Saisie des étudiants
etudiants = []
for i in range(n):
    print(f"nÉtudiant {i + 1} :")
    etudiants.append(saisir_etudiant())
# b) Afficher les noms des étudiants avec moyenne >= 10
print("\nÉtudiants ayant une moyenne ≥ 10 :")
for e in etudiants:
    if e["moyenne"] >= 10:
        print(f"{e['nom']} {e['prenom']}")
# c) Afficher l'étudiant avec la moyenne maximale
etudiant_max = max(etudiants, key=lambda x: x["moyenne"])
print("\nÉtudiant ayant la meilleure moyenne :")
afficher_etudiant(etudiant_max)
```



# Les fichiers

Chapitre 4

Les Fichiers

- ❑ Les informations utilisées dans tous les programmes que nous avons déjà écrits 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.

72

Chapitre 4

Les Fichiers

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.

73

Chapitre 4

Les Fichiers

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 créés 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 stocker 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.

74

Chapitre 4

Les Fichiers

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.

75

Chapitre 4

Les Fichiers

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.

76

Chapitre 4

Les Fichiers

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.

77

Chapitre 4

Les Fichiers

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.

78

Chapitre 4

Les Fichiers

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.

79

Chapitre 4

Les Fichiers

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 : chaîne  
Début  
OUVRIR ("fiche.txt",4,"Lecture")  
**LireFichier** (4, ch)  
....  
Fin

80

Chapitre 4

Les Fichiers

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

81

Chapitre 4

Les Fichiers

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, "Lecture")

Tantque Non EOF(4)

LireFichier(4, ch)

FinTantque

....

82

Chapitre 4

Les Fichiers

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)

83

Chapitre 4

Les Fichiers en Python

Les Fichiers en Python

En Python, un fichier est un espace de stockage sur le disque (mémoire secondaire) qui contient des données. Un fichier peut être de texte (fichier .txt, .csv,.json, etc.) ou binaire (images, vidéos, etc.).

Python permet de :

▪ Créer, ouvrir, lire, écrire, modifier et fermer des fichiers facilement.

▪ Utiliser des fonctions intégrées comme open(), read(), write(), close(), etc.

▪ Gérer les fichiers avec différentes modes d'ouverture :

- 'r' : lecture seule (read)

- 'w' : écriture (écrase le fichier s'il existe)

- 'a' : ajout (append à la fin du fichier)

- 'b' : mode binaire (ex : 'rb', 'wb')

- '+' : lecture et écriture (ex : 'r+', 'w+')

84

Chapitre 4

Les Fichiers en Python

Les Fichiers en Python

Ouverture d'un fichier

```
f = open("exemple.txt", "r") # Ouvre le fichier en lecture
```

Lecture d'un fichier

```
f = open("exemple.txt", "r") # Ouvre le fichier en lecture
contenu = f.read()           # Lit tout le contenu
f.close()
```

Autres méthodes utiles :

```
ligne = f.readline()         # Lit une ligne
lignes = f.readlines()       # Liste de toutes les lignes
```

Ecriture dans un fichier

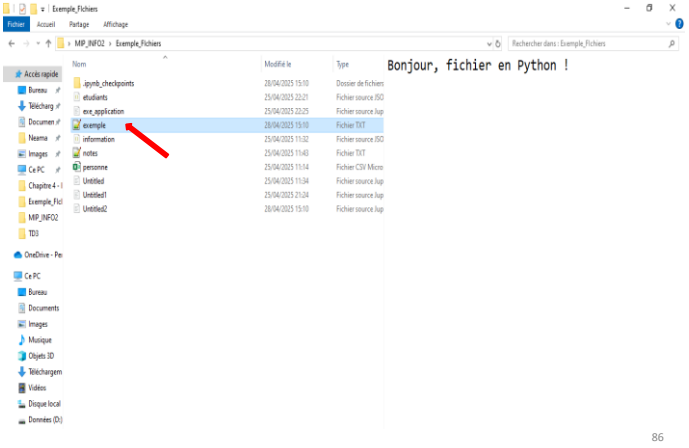
```
# Ouvrir un fichier en mode écriture
fichier = open("exemple.txt", "w")
fichier.write("Bonjour, fichier en Python !")
fichier.close()
```

85

Chapitre 4

Les Fichiers en Python

Les Fichiers en Python



86

Chapitre 4

Les Fichiers en Python

Les Fichiers en Python

Pour ajouter du texte sans écraser :

```
f = open("exemple.txt", "a")
f.write("Ajout d'une nouvelle ligne\n")
f.close()
```

Pour lire et écrire en même temps dans un fichier:

```
# Ouvrir un fichier en mode lecture et écriture
fichier=open("demo.txt", "w+")
# Écriture dans le fichier
fichier.write("Bonjour\nBienvenue en Python\n")
# Remettre le curseur au début pour pouvoir lire
fichier.seek(0) # repositionne au début du fichier
# Lecture du contenu
contenu = fichier.read()
print("Contenu du fichier après écriture :")
print(contenu)
fichier.close()
```

w+ → Ouvre le fichier pour écrire ET lire. ( Il écrase le fichier s'il existe !)

f.seek(0) → Très important : sans seek(0), la lecture commencerait à la fin et ne lirait rien.

87

Chapitre 4

Les Fichiers en Python

Les Fichiers en Python

Le bloc with

Permet de gérer automatiquement la fermeture du fichier :

```
with open("exemple.txt", "r") as f:
    contenu = f.read()
print(contenu)
```

with ... as f :

Le mot-clé with crée un contexte sécurisé :

- Il ouvre le fichier au début du bloc.
- Il ferme automatiquement le fichier à la fin (même s'il y a une erreur).

as f :

donne un nom temporaire (f) à l'objet fichier pour l'utiliser à l'intérieur du bloc.

88

Pr. Khadija LOUZAOU

22

Chapitre 4Les Fichiers en Python

Les Fichiers en Python

Traitement ligne par ligne

```
with open("exemple.txt", "r") as f:
    for ligne in f:
        print(ligne.strip())
```

**ligne.strip()** : Cette méthode supprime les espaces, tabulations ou retours à la ligne (\n) au début et à la fin de la ligne.

Par exemple, si la ligne dans le fichier est "Bonjour\n", alors ligne.strip() devient "Bonjour".

89

Chapitre 4Les Fichiers en Python

Les Fichiers CSV en Python

CSV signifie Comma-Separated Values (valeurs séparées par des virgules). C'est un format simple et largement utilisé pour représenter des tableaux de données (comme des feuilles Excel).

**Ecriture dans un fichier CSV**

```
import csv

with open("personnes.csv", "w", newline='', encoding='utf-8') as f:
    ecrivain = csv.writer(f)
    ecrivain.writerow(["Nom", "Age", "Ville"])
    ecrivain.writerow(["Ahmed", 30, "Rabat"])
    ecrivain.writerow(["Sara", 25, "Tanger"])
```

Remarques :

- `newline=''` évite les lignes vides sur certains systèmes.
- `encoding='utf-8'` est recommandé pour les caractères accentués.

90

Chapitre 4Les Fichiers en Python

Les Fichiers CSV en Python

Lecture dans un fichier CSV

```
import csv

with open("personnes.csv", "r", newline='', encoding='utf-8') as f:
    lecteur = csv.reader(f)
    for ligne in lecteur:
        print(ligne)
```

**Lecture avec DictReader**

```
import csv

with open("personnes.csv", "r", newline='', encoding='utf-8') as f:
    lecteur = csv.DictReader(f)
    for ligne in lecteur:
        print(ligne["Nom"], ligne["Ville"])
```

91

Chapitre 4Les Fichiers en Python

Les Fichiers CSV en Python

**Ecriture avec DictWriter**

```
import csv

personnes = [
    {"Nom": "Ahmed", "Age": 30, "Ville": "Rabat"},
    {"Nom": "Sara", "Age": 25, "Ville": "Tanger"}
]

with open("personnes.csv", "w", newline='', encoding='utf-8') as f:
    champs = ["Nom", "Age", "Ville"]
    ecrivain = csv.DictWriter(f, fieldnames=champs)
    ecrivain.writeheader()
    ecrivain.writerows(personnes)
```

92

Le format JSON en Python

**JSON** (JavaScript Object Notation) est : Un format léger, lisible et standardisé pour représenter des données structurées.

Utilisé pour échanger des données entre un client (ex. navigateur) et un serveur, ou entre fichiers et programmes.

En Python, on utilise le module **json**.

Le format JSON en Python  
Sauvegarde dans un fichier JSON

```
import json
donnees = {
    "nom": "Ali",
    "age": 30,
    "ville": "Rabat"
}
with open("utilisateur.json", "w") as f:
    json.dump(donnees, f, indent=4)
```

**json.dump()** : fonction du module json qui écrit des données au format JSON dans un fichier.

Arguments :

- **donnees** : un objet Python (ex: dictionnaire ou liste de dictionnaires).
- **f** : le fichier ouvert en écriture.
- **indent=4** :
  - Ajoute une indentation de 4 espaces dans le fichier JSON généré.
  - Rends le fichier lisible pour les humains.

Le format JSON en Python  
Lecture du fichier JSON

```
import json
with open("utilisateur.json", "r") as f:
    donnees_lues = json.load(f)
    print(donnees_lues)
```

**json.load()** : Lit le contenu JSON du fichier et le convertit en objet Python.

**f** : désigne le fichier ouvert (utilisateur.json).

**donnees\_lues** : Va contenir les données Python équivalentes.  
Par exemple, si le fichier contient un dictionnaire JSON, il devient un dictionnaire Python.

**print(donnees\_lues)**: Affiche les données récupérées à l'écran.

Tableau comparatif

Format	Description	Structure	Lisibilité	Usage principal
Texte (.txt)	Contient des données brutes ou du texte non structuré	Aucune structure définie	Très lisible	Notes, logs, messages simples
Binaire	Données encodées (non lisibles directement par l'humain)	Dépend du programme qui l'utilise	Peu lisible	Images, vidéos, fichiers compilés
CSV (.csv)	Données tabulaires séparées par des virgules ou points-virgules	Tableaux simples (lignes/colonnes)	Moyennement lisible	Bases de données simples, tableurs (Excel)
JSON (.json)	Format texte structuré, léger, lisible et universel	Hiérarchique (objets, listes, dictionnaires)	Très lisible	Échange de données entre applications, APIs, configurations



Chapitre 4

Les Fichiers en Python

Résumé

- JSON se distingue car il combine :
  - la lisibilité d'un fichier texte
  - la structuration des fichiers CSV
  - peut représenter des données complexes, comme les objets en mémoire.
- Il est plus souple que le CSV
- Plus lisible que le binaire
- Plus structuré qu'un simple fichier texte

Ce qui en fait un format central dans les échanges de données modernes.

97

Chapitre 4

Les Fichiers en Python

Exercice

1. Ecrire un programme en python qui 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.

98

Chapitre 4

Les Fichiers

Solution

```
1 # Ouvrir le fichier en mode écriture
with open("notes.txt", "w") as fichier:
    for i in range(5):
        print(f"Étudiant {i+1} :")
        nom = input("Nom : ")
        prenom = input("Prénom : ")
        classe = input("Classe : ")
        note = input("Note : ")
    # Ecriture des données dans le fichier (une ligne par étudiant)
    ligne = f"{nom},{prenom},{classe},{note}\n"
    fichier.write(ligne)
print("Les données ont été enregistrées dans notes.txt.")

2 try:
    with open("notes.txt", "r") as fichier:
        lignes = fichier.readlines()
        print("\n--- Contenu du fichier notes.txt ---")
        for i, ligne in enumerate(lignes, start=1):
            nom, prenom, classe, note = ligne.strip().split(",")
            print(f"Étudiant {i} : {nom} {prenom}, Classe : {classe}, Note : {note}")
    except FileNotFoundError:
        print("Le fichier notes.txt n'existe pas.")
```

99

Chapitre 4

Les Fichiers

Exercice d'application

❑ Soit l'enregistrement Etudiant constituée par :

- code : Entier
- nom : Chaîne
- prénom : Chaîne
- genre : Caractère
- moyenne: Réel

1. Déclarer l'enregistrement.
2. Ecrire une procédure qui permet de remplir les champs de cet enregistrement.
3. Ecrire une procédure qui affiche ces champs.
4. Ecrire un algorithme principal dans lequel vous allez :
  - a- Saisir les informations de n étudiants donnés (n≤100).
  - b- Afficher les noms des étudiants ayant une moyenne ≥10.
  - c- Afficher les informations de l'étudiant qui a la moyenne maximale.
5. Traduire cet algorithme en python en utilisant les dictionnaires.

100

Chapitre 4Les Fichiers

Exercice d'application

6. Ecrire une fonction qui **enregistre** la liste des étudiants dans un fichier JSON nommé "etudiants.json", où chaque étudiant est représenté comme un dictionnaire.

7. Ecrire une fonction qui **lit** le fichier "etudiants.json" et reconstitue la liste des étudiants sous forme de dictionnaires.

8. Ecrire une fonction qui **recherche** un étudiant dans le fichier "etudiants.json" en fonction de son code, et retourne ses informations si trouvées.

9. Ecrire une fonction qui permet de **mettre à jour** la moyenne d'un étudiant donné (recherche par code) dans le fichier "etudiants.json".

10. Ecrire une fonction qui **supprime** un étudiant du fichier "etudiants.json" (recherche par code), puis enregistre le reste des étudiants dans un nouveau fichier JSON.

Chapitre 4Les Fichiers

Exercice d'application

6. Ecrire une fonction qui **enregistre** la liste des étudiants dans un fichier JSON nommé "etudiants.json", où chaque étudiant est représenté comme un dictionnaire.

```
def enregistrer_etudiants_json(fichier, liste):  
    with open(fichier, "w", encoding="utf-8") as f:  
        json.dump(liste, f, indent=4)
```

7. Ecrire une fonction qui **lit** le fichier "etudiants.json" et reconstitue la liste des étudiants sous forme de dictionnaires.

```
def lire_etudiants_json(fichier):  
    with open(fichier, "r", encoding="utf-8") as f:  
        return json.load(f)
```

Chapitre 4Les Fichiers

Exercice d'application

8. Ecrire une fonction qui **recherche** un étudiant dans le fichier "etudiants.json" en fonction de son code, et retourne ses informations si trouvées.

```
def rechercher_etudiant_json(fichier, code_recherche):  
    liste = lire_etudiants_json(fichier)  
    for e in liste:  
        if e["code"] == code_recherche:  
            return e  
    return None
```

9. Ecrire une fonction qui permet de **mettre à jour** la moyenne d'un étudiant donné (recherche par code) dans le fichier "etudiants.json".

```
def mettre_a_jour_moyenne_json(fichier, code_recherche, nouvelle_moyenne):  
    liste = lire_etudiants_json(fichier)  
    for e in liste:  
        if e["code"] == code_recherche:  
            e["moyenne"] = nouvelle_moyenne  
            break  
    enregistrer_etudiants_json(fichier, liste)
```

Chapitre 4Les Fichiers

Exercice d'application

10. Ecrire une fonction qui **supprime** un étudiant du fichier "etudiants.json" (recherche par code), puis enregistre le reste des étudiants dans un nouveau fichier JSON.

```
def supprimer_etudiant_json(fichier, code_recherche, nouveau_fichier):  
    liste = lire_etudiants_json(fichier)  
    nouvelle_liste = [e for e in liste if e["code"] != code_recherche]  
    enregistrer_etudiants_json(nouveau_fichier, nouvelle_liste)
```