

## C7 Lecture et traitement de données en tables

### Données en tables

- Le traitement et l'analyse de données volumineuses (*big data*) est l'une des activités principales en informatique de nos jours .

### Données en tables

- Le traitement et l'analyse de données volumineuses (*big data*) est l'une des activités principales en informatique de nos jours .
- Ces données sont souvent organisées en **tables**.

### Données en tables

- Le traitement et l'analyse de données volumineuses (*big data*) est l'une des activités principales en informatique de nos jours .
- Ces données sont souvent organisées en **tables**.
- Une ligne de données en table s'appelle un **enregistrement**

### Données en tables

- Le traitement et l'analyse de données volumineuses (*big data*) est l'une des activités principales en informatique de nos jours .
- Ces données sont souvent organisées en **tables**.
- Une ligne de données en table s'appelle un **enregistrement**
- Une colonne s'appelle un **champ**

### Données en tables

- Le traitement et l'analyse de données volumineuses (*big data*) est l'une des activités principales en informatique de nos jours .
- Ces données sont souvent organisées en **tables**.
- Une ligne de données en table s'appelle un **enregistrement**
- Une colonne s'appelle un **champ**
- Les titres des colonnes sont les **descripteurs**

## C7 Lecture et traitement de données en tables

### Données en tables

- Le traitement et l'analyse de données volumineuses (*big data*) est l'une des activités principales en informatique de nos jours .
- Ces données sont souvent organisées en **tables**.
- Une ligne de données en table s'appelle un **enregistrement**
- Une colonne s'appelle un **champ**
- Les titres des colonnes sont les **descripteurs**

### Format csv

Le format de fichier **csv** (*comma separated value*) représente des données en tables. Chaque ligne du fichier est une donnée et sur chaque ligne les champs sont séparés par des virgules (ou parfois un autre caractère comme le point-virgule).

## C7 Lecture et traitement de données en tables

### Exemple

- Des données en table

| Nom      | Prénom | Naissance |
|----------|--------|-----------|
| Pascal   | Blaise | 1623      |
| Lovelace | Ada    | 1815      |
| Boole    | George | 1815      |

- Représentation en fichier csv

```
Nom;Prénom;Naissance
Pascal;Blaise;1623
Lovelace;Ada;1815
Boole;George;1815
```

Le fichier csv à droite sera utilisé par la suite, on l'appelle `exemple.csv` de façon à y faire référence.

## C7 Lecture et traitement de données en tables

### Exemple

- Des données en table

| Nom      | Prénom | Naissance |
|----------|--------|-----------|
| Pascal   | Blaise | 1623      |
| Lovelace | Ada    | 1815      |
| Boole    | George | 1815      |

- Représentation en fichier csv

```
Nom;Prénom;Naissance
Pascal;Blaise;1623
Lovelace;Ada;1815
Boole;George;1815
```

Le fichier csv à droite sera utilisé par la suite, on l'appelle `exemple.csv` de façon à y faire référence.

### Remarques

- La première ligne du fichier csv décrit les champs, il contient les **attributs** (appelés aussi **descripteur**).

## C7 Lecture et traitement de données en tables

### Exemple

- Des données en table

| Nom      | Prénom | Naissance |
|----------|--------|-----------|
| Pascal   | Blaise | 1623      |
| Lovelace | Ada    | 1815      |
| Boole    | George | 1815      |

- Représentation en fichier csv

```
Nom;Prénom;Naissance
Pascal;Blaise;1623
Lovelace;Ada;1815
Boole;George;1815
```

Le fichier csv à droite sera utilisé par la suite, on l'appelle `exemple.csv` de façon à y faire référence.

### Remarques

- La première ligne du fichier csv décrit les champs, il contient les **attributs** (appelés aussi **descripteur**).
- Les données d'un fichier csv sont au format texte, par conséquent même une donnée numérique (comme ici l'année de naissance) est en fait une chaîne de caractères.

## C7 Lecture et traitement de données en tables

### Gestions des fichiers en Python

En python, on peut ouvrir un fichier présent sur l'ordinateur à l'aide de l'instruction `open`. Cette instruction renvoie une variable appelée `descripteur de fichier` et prend un paramètre indiquant le mode d'ouverture du fichier :

### Exemples

### Gestions des fichiers en Python

En python, on peut ouvrir un fichier présent sur l'ordinateur à l'aide de l'instruction `open`. Cette instruction renvoie une variable appelée `descripteur de fichier` et prend un paramètre indiquant le mode d'ouverture du fichier :

- `"r"` (read) pour ouvrir le fichier en lecture. C'est le mode par défaut.

### Exemples

## C7 Lecture et traitement de données en tables

### Gestions des fichiers en Python

En python, on peut ouvrir un fichier présent sur l'ordinateur à l'aide de l'instruction `open`. Cette instruction renvoie une variable appelée `descripteur de fichier` et prend un paramètre indiquant le mode d'ouverture du fichier :

- `"r"` (read) pour ouvrir le fichier en lecture. C'est le mode par défaut.
- `"w"` (write) pour ouvrir le fichier en écriture. Attention, le contenu initial du fichier est alors perdu.

### Exemples

### Gestions des fichiers en Python

En python, on peut ouvrir un fichier présent sur l'ordinateur à l'aide de l'instruction `open`. Cette instruction renvoie une variable appelée `descripteur de fichier` et prend un paramètre indiquant le mode d'ouverture du fichier :

- `"r"` (read) pour ouvrir le fichier en lecture. C'est le mode par défaut.
- `"w"` (write) pour ouvrir le fichier en écriture. Attention, le contenu initial du fichier est alors perdu.
- `"a"` (append) pour ouvrir le fichier en ajout.

### Exemples

## C7 Lecture et traitement de données en tables

### Gestions des fichiers en Python

En python, on peut ouvrir un fichier présent sur l'ordinateur à l'aide de l'instruction `open`. Cette instruction renvoie une variable appelée `descripteur de fichier` et prend un paramètre indiquant le mode d'ouverture du fichier :

- `"r"` (read) pour ouvrir le fichier en lecture. C'est le mode par défaut.
- `"w"` (write) pour ouvrir le fichier en écriture. Attention, le contenu initial du fichier est alors perdu.
- `"a"` (append) pour ouvrir le fichier en ajout.

### Exemples

Ecrire l'instruction permettant de créer le descripteur de fichier `anniv` sur le fichier `anniversaires.txt` en mode ajout.

## C7 Lecture et traitement de données en tables

### Gestions des fichiers en Python

En python, on peut ouvrir un fichier présent sur l'ordinateur à l'aide de l'instruction `open`. Cette instruction renvoie une variable appelée `descripteur de fichier` et prend un paramètre indiquant le mode d'ouverture du fichier :

- `"r"` (read) pour ouvrir le fichier en lecture. C'est le mode par défaut.
- `"w"` (write) pour ouvrir le fichier en écriture. Attention, le contenu initial du fichier est alors perdu.
- `"a"` (append) pour ouvrir le fichier en ajout.

### Exemples

Ecrire l'instruction permettant de créer le descripteur de fichier `anniv` sur le fichier `anniversaires.txt` en mode ajout.

```
anniv = open("anniversaires.txt","a")
```

## C7 Lecture et traitement de données en tables

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier crée à l'aide de l'instruction `open` :

### Exemples

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier créé à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`

### Exemples

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier créé à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`

### Exemples

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier crée à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`
- Ecriture avec de `write`

### Exemples

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier crée à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`
- Ecriture avec de `write`

### Exemples

## C7 Lecture et traitement de données en tables

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier crée à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`
- Ecriture avec de `write`
- Fermeture `close`

### Exemples

## C7 Lecture et traitement de données en tables

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier créé à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`
- Ecriture avec de `write`
- Fermeture `close`

### Exemples

Ouvrir le fichier "truc.txt", lire sa première ligne puis le refermer.

## C7 Lecture et traitement de données en tables

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier crée à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`
- Ecriture avec de `write`
- Fermeture `close`

### Exemples

Ouvrir le fichier "truc.txt", lire sa première ligne puis le refermer.

```
fic = open("truc.txt","r")
```

## C7 Lecture et traitement de données en tables

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier créée à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`
- Ecriture avec de `write`
- Fermeture `close`

### Exemples

Ouvrir le fichier "truc.txt", lire sa première ligne puis le refermer.

```
fic = open("truc.txt","r")  
lig1 = fic.readline()
```

## C7 Lecture et traitement de données en tables

### Opérations sur les descripteurs de fichiers

Les opérations suivantes sont possibles sur un descripteur de fichier créée à l'aide de l'instruction `open` :

- Lecture du contenu complet du fichier avec `read`
- Lecture du contenu ligne par ligne avec `readline`
- Ecriture avec de `write`
- Fermeture `close`

### Exemples

Ouvrir le fichier "truc.txt", lire sa première ligne puis le refermer.

```
fic = open("truc.txt","r")  
lig1 = fic.readline()  
fic.close()
```

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

### Exemples

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

- Un dictionnaire se note entre accolades : { et }

### Exemples

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

- Un dictionnaire se note entre accolades : { et }
- Les paires clés/valeurs sont séparés par des virgules.

### Exemples

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

- Un dictionnaire se note entre accolades : { et }
- Les paires clés/valeurs sont séparés par des virgules.
- Le caractère ":" sépare une clé de la valeur associée.

### Exemples

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

- Un dictionnaire se note entre accolades : { et }
- Les paires clés/valeurs sont séparés par des virgules.
- Le caractère ":" sépare une clé de la valeur associée.

### Exemples

- Un dictionnaire contenant des objets et leurs prix :

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

- Un dictionnaire se note entre accolades : { et }
- Les paires clés/valeurs sont séparés par des virgules.
- Le caractère ":" sépare une clé de la valeur associée.

### Exemples

- Un dictionnaire contenant des objets et leurs prix :  

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16}
```

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

- Un dictionnaire se note entre accolades : { et }
- Les paires clés/valeurs sont séparés par des virgules.
- Le caractère ":" sépare une clé de la valeur associée.

### Exemples

- Un dictionnaire contenant des objets et leurs prix :  
`prix = { "verre":12 , "tasse" : 8, "assiette" : 16}`
- Un dictionnaire traduisant des couleurs du français vers l'anglais

## C7 Lecture et traitement de données en tables

### Les dictionnaires de Python

- Les dictionnaires de Python permettent de stocker des données sous forme de tableau associant une clé à une valeur :

|         |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|
| Valeurs | val1   | val2   | val3   | val4   | val5   |
| ↑       | ↑      | ↑      | ↑      | ↑      | ↑      |
| Clés    | 'cle1' | 'cle2' | 'cle3' | 'cle4' | 'cle5' |

- Un dictionnaire se note entre accolades : { et }
- Les paires clés/valeurs sont séparés par des virgules.
- Le caractère ":" sépare une clé de la valeur associée.

### Exemples

- Un dictionnaire contenant des objets et leurs prix :  
`prix = { "verre":12 , "tasse" : 8, "assiette" : 16}`
- Un dictionnaire traduisant des couleurs du français vers l'anglais  
`couleurs = { "vert":"green" , "bleu" : "blue", "rouge" : "red" }`

### Opérations sur un dictionnaire

- On accède aux éléments d'un dictionnaire avec la syntaxe `nom_dictionnaire[cle]`

### Opérations sur un dictionnaire

- On accède aux éléments d'un dictionnaire avec la syntaxe

`nom_dictionnaire[cle]`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" :  
30 }
```

Par exemple, `prix["verre"]` contient 12

### Opérations sur un dictionnaire

- On accède aux éléments d'un dictionnaire avec la syntaxe

`nom_dictionnaire[cle]`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" :  
30 }
```

Par exemple, `prix["verre"]` contient 12

- On peut ajouter une clé à un dictionnaire existant en effectuant une affectation `nom_dictionnaire[nouvelle_cle]=nouvelle_valeur`

### Opérations sur un dictionnaire

- On accède aux éléments d'un dictionnaire avec la syntaxe

`nom_dictionnaire[cle]`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" :  
30 }
```

Par exemple, `prix["verre"]` contient 12

- On peut ajouter une clé à un dictionnaire existant en effectuant une affectation `nom_dictionnaire[nouvelle_cle]=nouvelle_valeur`

On ajoute un nouvel objet avec son prix :

```
prix["couteau"]=20
```

### Opérations sur un dictionnaire

- On accède aux éléments d'un dictionnaire avec la syntaxe

`nom_dictionnaire[cle]`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" :  
30 }
```

Par exemple, `prix["verre"]` contient 12

- On peut ajouter une clé à un dictionnaire existant en effectuant une affectation `nom_dictionnaire[nouvelle_cle]=nouvelle_valeur`

On ajoute un nouvel objet avec son prix :

```
prix["couteau"]=20
```

- On peut modifier la valeur associée à une clé avec une affectation

`nom_dictionnaire[cle]=nouvelle_valeur`

### Opérations sur un dictionnaire

- On accède aux éléments d'un dictionnaire avec la syntaxe

`nom_dictionnaire[cle]`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" :  
30 }
```

Par exemple, `prix["verre"]` contient 12

- On peut ajouter une clé à un dictionnaire existant en effectuant une affectation `nom_dictionnaire[nouvelle_cle]=nouvelle_valeur`

On ajoute un nouvel objet avec son prix :

```
prix["couteau"]=20
```

- On peut modifier la valeur associée à une clé avec une affectation

`nom_dictionnaire[cle]=nouvelle_valeur`

Le pris d'une tasse passe à 10 :

```
prix["tasse"]=10
```

### Présence dans un dictionnaire

- Attention, essayer d'accéder à une clé qui n'est pas dans un dictionnaire renvoie une erreur !

## C7 Lecture et traitement de données en tables

### Présence dans un dictionnaire

- Attention, essayer d'accéder à une clé qui n'est pas dans un dictionnaire renvoie une erreur !

Il n'y a pas de clé 'fourchette' dans le dictionnaire prix, donc `prix['fourchette']` renvoie une erreur (**KeyError**).

## C7 Lecture et traitement de données en tables

### Présence dans un dictionnaire

- Attention, essayer d'accéder à une clé qui n'est pas dans un dictionnaire renvoie une erreur!  
Il n'y a pas de clé 'fourchette' dans le dictionnaire prix, donc `prix['fourchette']` renvoie une erreur (`KeyError`).
- On teste la présence d'une clé dans un dictionnaire avec `cle in nom_dictionnaire`

### Présence dans un dictionnaire

- Attention, essayer d'accéder à une clé qui n'est pas dans un dictionnaire renvoie une erreur !

Il n'y a pas de clé 'fourchette' dans le dictionnaire prix, donc `prix['fourchette']` renvoie une erreur (`KeyError`).

- On teste la présence d'une clé dans un dictionnaire avec `cle in nom_dictionnaire`

la fourchette n'est pas dans le dictionnaire prix

Le test `fourchette in prix` renvoie `False`

### Présence dans un dictionnaire

- Attention, essayer d'accéder à une clé qui n'est pas dans un dictionnaire renvoie une erreur!  
Il n'y a pas de clé 'fourchette' dans le dictionnaire prix, donc `prix['fourchette']` renvoie une erreur (`KeyError`).
- On teste la présence d'une clé dans un dictionnaire avec `cle in nom_dictionnaire`  
la fourchette n'est pas dans le dictionnaire prix  
Le test `fourchette in prix` renvoie `False`
- On peut supprimer une clé existante dans un dictionnaire avec `del nom_dictionnaire[cle]`

### Présence dans un dictionnaire

- Attention, essayer d'accéder à une clé qui n'est pas dans un dictionnaire renvoie une erreur!  
Il n'y a pas de clé 'fourchette' dans le dictionnaire prix, donc `prix['fourchette']` renvoie une erreur (`KeyError`).
- On teste la présence d'une clé dans un dictionnaire avec `cle in nom_dictionnaire`  
la fourchette n'est pas dans le dictionnaire prix  
Le test `fourchette in prix` renvoie `False`
- On peut supprimer une clé existante dans un dictionnaire avec `del nom_dictionnaire[cle]`  
On supprimer le couteau :  
`del prix["couteau"]`

### Parcours d'un dictionnaire

- Le parcours par clé s'effectue directement avec `for cle in nom_dictionnaire`

### Parcours d'un dictionnaire

- Le parcours par clé s'effectue directement avec `for cle in nom_dictionnaire`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" : 30 }
```

Par exemple, `for objet in prix` permettra à la variable `objet` de prendre successivement les valeurs des clés : "verre", "tasse", "assiette" et "plat".

### Parcours d'un dictionnaire

- Le parcours par clé s'effectue directement avec `for cle in nom_dictionnaire`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" : 30 }
```

Par exemple, `for objet in prix` permettra à la variable `objet` de prendre successivement les valeurs des clés : "verre", "tasse", "assiette" et "plat".

- Le parcours par valeur s'effectue en ajoutant `.values()` au nom du dictionnaire : `for valeur in nom_dictionnaire.values()`

### Parcours d'un dictionnaire

- Le parcours par clé s'effectue directement avec `for cle in nom_dictionnaire`

```
prix = { "verre":12 , "tasse" : 8, "assiette" : 16, "plat" : 30 }
```

Par exemple, `for objet in prix` permettra à la variable `objet` de prendre successivement les valeurs des clés : "verre", "tasse", "assiette" et "plat".

- Le parcours par valeur s'effectue en ajoutant `.values()` au nom du dictionnaire : `for valeur in nom_dictionnaire.values()`

Par exemple, `for p in prix.values()` permettra à la variable `p` de prendre successivement les valeurs du dictionnaire : 12, 8 , 16 et 30.

## C7 Lecture et traitement de données en tables

### Python et les fichiers csv

- Le module `csv` de Python permet de récupérer les informations d'un fichier csv, sous forme de listes de listes ou de dictionnaires

## C7 Lecture et traitement de données en tables

### Python et les fichiers csv

- Le module `csv` de Python permet de récupérer les informations d'un fichier csv, sous forme de listes de listes ou de dictionnaires
- Pour les dictionnaires, ce sont alors les descripteurs qui servent de clés.

## C7 Lecture et traitement de données en tables

### Python et les fichiers csv

- Le module `csv` de Python permet de récupérer les informations d'un fichier csv, sous forme de listes de listes ou de dictionnaires
- Pour les dictionnaires, ce sont alors les descripteurs qui servent de clés.
- Tous les champs (même ceux contenant des nombres) sont récupérés sous forme de chaînes de caractères (type `str` de Python) à la façon de ce qui se passe lors d'un `input`. Faire donc attention lors de calculs ou de comparaisons avec les données de ces champs.

## C7 Lecture et traitement de données en tables

### Exemple

Récupération des éléments du fichier `exemple.csv` ci-dessus dans un dictionnaire :

```
1 import csv
2 fic=open("exemple.csv","r",encoding="utf-8")
3 # Lecture sous forme de dictionnaire
4 donnees = list(csv.DictReader(fic , delimiter=';'))
5 fic.close()
```

## C7 Lecture et traitement de données en tables

### Exemple

Récupération des éléments du fichier `exemple.csv` ci-dessus dans un dictionnaire :

```
1 import csv
2 fic=open("exemple.csv","r",encoding="utf-8")
3 # Lecture sous forme de dictionnaire
4 donnees = list(csv.DictReader(fic , delimiter=';'))
5 fic.close()
```

Après execution, on a par exemple

```
donnees[0]={'Nom' : 'Pascal', 'Prenom' : 'Blaise', 'Naissance' :
'1623'}
```

## C7 Lecture et traitement de données en tables

### Exemple

Récupération des éléments du fichier `exemple.csv` ci-dessus dans un dictionnaire :

```
1 import csv
2 fic=open("exemple.csv","r",encoding="utf-8")
3 # Lecture sous forme de dictionnaire
4 donnees = list(csv.DictReader(fic , delimiter=';'))
5 fic.close()
```

Après execution, on a par exemple

```
donnees[0]={'Nom' : 'Pascal', 'Prenom' : 'Blaise', 'Naissance' : '1623'}
```

C'est à dire que chaque ligne de la table correspond à un dictionnaire

### Traitement des données

- Une fois les données csv lues et récupérées dans une liste de dictionnaires, on peut trier les informations et y faire des recherches.

### Traitement des données

- Une fois les données csv lues et récupérées dans une liste de dictionnaires, on peut trier les informations et y faire des recherches.
- Par exemple pour le fichier csv donné en exemple :

```
Nom;Prénom;Naissance
Pascal;Blaise;1623
Lovelace;Ada;1815
Boole;George;1815
```

### Traitement des données

- Une fois les données csv lues et récupérées dans une liste de dictionnaires, on peut trier les informations et y faire des recherches.
- Par exemple pour le fichier csv donné en exemple :

```
Nom;Prénom;Naissance
Pascal;Blaise;1623
Lovelace;Ada;1815
Boole;George;1815
```

- Si les données sont récupérées dans la liste de dictionnaire `personnages`. On peut afficher les personnes nées en 1815 avec :

```
1     for p in personnages:
2         if p["Naissance"]=="1815":
3             print(p["Nom"],p["Prénom"])
```

### Trier une liste en Python

- La fonction `sorted` de Python permet de trier une liste. Elle renvoie la liste triée. La syntaxe est la suivante : `liste_triee = sorted(liste)`.

### Trier une liste en Python

- La fonction `sorted` de Python permet de trier une liste. Elle renvoie la liste triée. La syntaxe est la suivante : `liste_triee = sorted(liste)`.
- Par exemple :

```
1 notes = [15,11,10,18,9]
2 note_triees=sort(notes)
3 print(notes_triees)
```

affichera : [9,10,11,15,18]

### Trier une liste en Python

- La fonction `sorted` de Python permet de trier une liste. Elle renvoie la liste triée. La syntaxe est la suivante : `liste_triee = sorted(liste)`.
- Par exemple :

```
1 notes = [15,11,10,18,9]
2 note_triees=sort(notes)
3 print(notes_triees)
```

affichera : [9,10,11,15,18]

- On peut obtenir un tri par ordre décroissant en indiquant `reverse=True`  
`liste_triee = sorted(liste,reverse=True)`.

### Trier une liste de dictionnaires

- La fonction `sorted` permet aussi de trier des listes de dictionnaires on indique alors le critère de tri à l'aide de l'option `key`.

### Trier une liste de dictionnaires

- La fonction `sorted` permet aussi de trier des listes de dictionnaires on indique alors le critère de tri à l'aide de l'option `key`.
- Par exemple :

```
1 def note(eleve):  
2     return eleve["Note"]  
3  
4 notes = [{"Prenom": "Albert", "Note": 15}, {"Prenom": "Jim"  
5         , "Note": 10}, {"Prenom": "Sarah", "Note": 19}]  
6 notes.sort(key=note, reverse=True)  
print(notes)
```

affichera : `[{'Prenom': 'Sarah', 'Note': 19}, {'Prenom': 'Albert', 'Note': 15}, {'Prenom': 'Jim', 'Note': 10}]`