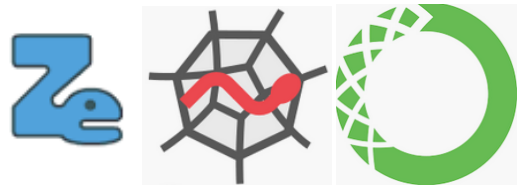



Les données en tables

```
Salle;Adresse;Téléphone
Pathé;Grenoble 21 Boulevard Maréchal Lyautey;08 92 69 66 96
Le Mèlies;Grenoble 28 Allée Henri Frenay;04 76 47 99 31
Le Club;Grenoble Rue du Phalanstère 9 B;04 76 87 46 21
Les 6 Rex;Grenoble 13 Rue Saint-Jacques;04 76 44 06 82
La Nef;Grenoble 1 Rue Emile Augier;08 92 31 62 05
Ciné Club de Grenoble;Grenoble 4 Rue Hector Berlioz;04 76 44 70 38
Juliet Berto;Grenoble Passage Palais de Justice;04 76 00 14 61
```


Résumé :

Les données en tables sont très utilisées pour représenter et traiter de l'information.

La structure qui supporte ces données est constituée de fichiers de textes 'organisés' : les fichiers au format csv. Ils sont l'objet de ce cours.

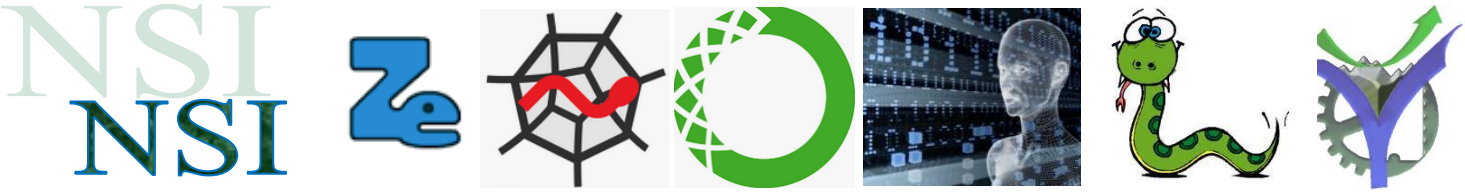
Sommaire :

1 Les données en table 2
1.1 Présentation 2
1.2 Structure de ces tables : 3
1.3 Exploitation des tables quelques exemples 3
a) Sélection d'une ligne répondant à un critère 3
b) Sélection d'une ou plusieurs colonnes répondant à un critère 4
c) Regroupement de tables 4
d) Traitements combinés 5

2 Représentation des données : les fichiers .csv 6
2.1 Le format de fichier csv 6
2.2 Ouverture d'un fichier csv avec python 7
a) Lecture en liste de listes 7
b) Lecture en liste de dictionnaires 8
2.3 Travailler avec les tables (Liste de listes) 9
a) Sélectionner des lignes : sélection σ 9
b) Sélectionner des colonnes : projection π 9
c) Fusion de deux tables : jointure ⋈ 10
2.4 Travailler avec les tables (Liste de dictionnaires) 12
a) Sélectionner des lignes : sélection σ 12
b) Sélectionner des colonnes : projection π 12
c) Fusion de deux tables : jointure ⋈ 13
d) Fusion de deux tables exploitation de la jointure 13
2.5 Enregistrer les résultats : écriture d'un fichier au format .csv 14
a) Création d'un fichier csv à partir d'une liste de listes 15
b) Création d'un fichier csv à partir d'une liste de dictionnaires 15
c) L'utilité de la structure with …. as en python 16

3 Quelques problèmes dans l'utilisation des formats csv et leurs solutions 17
3.1 Conflit entre les différentes données écrites en textes 17
3.2 Mauvaise gestion des sauts de lignes dans le fichier csv généré 18

4 En guise de conclusion 19
4.1 Alors liste de listes ou listes de dictionnaires ? 19
4.2 Vers l'infini et au-delà …. Les bases de données 19
4.3 Quelques ressources 19



1 Les données en table

1.1 Présentation

Nous utilisons tous les jours des données en table sans le savoir. Par exemple quand nous voulons organiser une sortie cinéma en consultant les programmations dans les différentes salles de la ville ou bien en cherchant à répondre à des questions du genre : « Mais tel acteur il jouait bien dans tel film ? ».
La réponse se trouve dans l'analyse des tables ci-dessous :

La table film :

Titre	Directeur	Acteur
Le crabe tambour	Schoendoerffer	Rochefort
Le crabe tambour	Schoendoerffer	Perrin
...		
Vingt Mille Lieues sous les mers	Fleischer	Douglas
Vingt Mille Lieues sous les mers	Fleischer	Mason
Vingt Mille Lieues sous les mers	Fleischer	Lukas

La table coordonnées

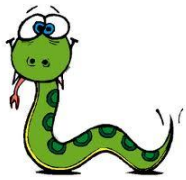
Salle	Adresse	Téléphone
Pathé	Grenoble 21 Boulevard Maréchal Lyautey	08 92 69 66 96
Le Mèlies	Grenoble 28 Allée Henri Frenay	04 76 47 99 31
...		
Les 6 Rex	Grenoble 13 Rue Saint-Jacques	04 76 44 06 82
Ciné Club de Grenoble	Grenoble 4 Rue Hector Berlioz	04 76 44 70 38

La table séance

Salle	Titre	Horaire
Pathé	Mais qui a tué Harry ?	18H30
Pathé	Vingt Mille Lieues sous les mers	19H00
...		
Les 6 Rex	Mais qui a tué Harry ?	18H00
Les 6 Rex	Le souper	22H00

L'ensemble de ces trois tables forme **une base de données**. Les bases de données seront plus particulièrement abordées en terminale avec un langage de requête de type SQL.






1.2 Structure de ces tables :

Nous avons pour chacune de ces tables une première ligne particulière : elle décrit l'organisation de la table, la liste des différents attributs ou colonne y est indiquée. C'est le descripteur :

Titre	Directeur	Acteur
Le crabe tambour	Schoendoerffer	Rochefort
Le crabe tambour	Schoendoerffer	Perrin
...		
Vingt Mille Lieues sous les mers	Fleischer	Douglas
Vingt Mille Lieues sous les mers	Fleischer	Mason
Vingt Mille Lieues sous les mers	Fleischer	Lukas

Descripteur

Items

1.3 Exploitation des tables quelques exemples

A partir des tables ci-dessus nous pouvons répondre à des questions qui exploitent une ou plusieurs tables. Nous exploiteront ces possibilités dans des exemples en python.

Sélection d'une ligne répondant à un critère

C'est répondre à une question du style : « lister tous les films dirigés par Bergman ». Nous allons utiliser un extrait de la table film en exemple, nous verront comment gérer les doublons dans les réponses :

La sélection de lignes selon un critère s'appelle faire une sélection σ dans le langage des bases de données.

Titre	Directeur	Acteur
Mais qui a tué Harry ?	Hitchcock	Gwenn
Mais qui a tué Harry ?	Hitchcock	Forsythe
Mais qui a tué Harry ?	Hitchcock	MacLaine
Mais qui a tué Harry ?	Hitchcock	Hitchcock
Cris et chuchotements	Bergman	Andersson
Cris et chuchotements	Bergman	Sylwan
Cris et chuchotements	Bergman	Thulin
Cris et chuchotements	Bergman	Ullman
Vingt Mille Lieues sous les mers	Fleischer	Douglas
Vingt Mille Lieues sous les mers	Fleischer	Mason
Vingt Mille Lieues sous les mers	Fleischer	Lukas
Vingt Mille Lieues sous les mers	Fleischer	Lorre
Vingt Mille Lieues sous les mers	Fleischer	Wilke
La grande vadrouille	Oury	De Funès
La grande vadrouille	Oury	Bourvil
La grande vadrouille	Oury	Therry-Thomas
Le petit baigneur	Dhéry	Dhéry
Le petit baigneur	Dhéry	Brosset
Le petit baigneur	Dhéry	De Funès
Le petit baigneur	Dhéry	Legras
Le petit baigneur	Dhéry	Galabru
Le petit baigneur	Dhéry	Tornade
La soupe aux choux	Girault	De Funès
La soupe aux choux	Girault	Carmet

Sélection d'une ou plusieurs colonnes répondant à un critère

On peut également réaliser une sélection selon un ou plusieurs attributs d'une table et créer une nouvelle table à partir de cette sélection :

Titre	Directeur	Acteur
Mais qui a tué Harry ?	Hitchcock	Gwenn
Mais qui a tué Harry ?	Hitchcock	Forsythe
Mais qui a tué Harry ?	Hitchcock	MacLaine
Mais qui a tué Harry ?	Hitchcock	Hitchcock
Cris et chuchotements	Bergman	Andersson
Cris et chuchotements	Bergman	Sylwan
Cris et chuchotements	Bergman	Thulin
Cris et chuchotements	Bergman	Ullman
Vingt Mille Lieues sous les mers	Fleischer	Douglas
Vingt Mille Lieues sous les mers	Fleischer	Mason
Vingt Mille Lieues sous les mers	Fleischer	Lukas
Vingt Mille Lieues sous les mers	Fleischer	Lorre
Vingt Mille Lieues sous les mers	Fleischer	Wilke
La grande vadrouille	Oury	De Funès
La grande vadrouille	Oury	Bourvil
La grande vadrouille	Oury	Therry-Thomas
Le petit baigneur	Dhéry	Dhéry
Le petit baigneur	Dhéry	Brosset
Le petit baigneur	Dhéry	
Le petit baigneur	Dhéry	
Le petit baigneur	Dhéry	
Le petit baigneur	Dhéry	
La soupe aux choux	Girault	
La soupe aux choux	Girault	

La sélection de colonnes, ou attributs, s'appelle faire une projection π dans le langage des bases de données.

Directeur	Acteur
Hitchcock	Gwenn
Hitchcock	Forsythe
	MacLaine
	Hitchcock
	Andersson
	Sylwan
	Thulin
	Ullman
	Douglas
	Mason
	Lukas
	Lorre
Fleischer	Wilke
Oury	De Funès
Oury	Bourvil
Oury	Therry-Thomas
Dhéry	Dhéry
Dhéry	Brosset
Dhéry	De Funès
Dhéry	Legras
Dhéry	Galabru
Dhéry	Tornade
Girault	De Funès
Girault	Carmet

Regroupement de tables

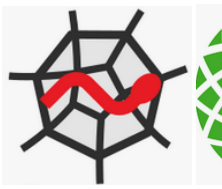
Les informations qui nous intéressent peuvent être réparties dans plusieurs tables. L'opération de fusion portera sur la mise en commun des informations de chacune des tables avec le même attribut en commun :

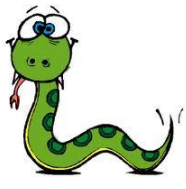
Titre	Directeur	Acteur
Le crabe tambour	Schoendoerffer	Rochefort
Le crabe tambour	Schoendoerffer	Perrin
...		
Vingt Mille Lieues sous les mers	Fleischer	Douglas
Vingt Mille Lieues sous les mers	Fleischer	Mason
Vingt Mille Lieues sous les mers	Fleischer	Lukas

Salle	Titre	Horaire
Pathé	Mais qui a tué Harry ?	18H30
Pathé	Vingt Mille Lieues sous les mers	19H00
...		
Les 6 Rex	Mais qui a tué Harry ?	18H00
Les 6 Rex	Le souper	22H00

Titre	Directeur	Acteur	Salle	Horaire
Le crabe tambour	Schoendoerffer	Rochefort	Le Mélies	21H30
Le crabe tambour	Schoendoerffer			H00
...				...
Vingt Mille Lieues sous les mers	Fleischer			H30
Vingt Mille Lieues sous les r				H00

La combinaison de plusieurs tables en une seule s'appelle faire une jointure $\bowtie$ dans le langage des bases de données.




Traitements combinés

On peut bien sûr combiner toutes ces opérations pour aboutir au résultat recherché. Pour chacun des exemples ci-dessous décrire le raisonnement à réaliser pour obtenir le résultat demandé.

Notre base de données est constituée des trois tables

Film

Titre	Directeur	Acteur
Le crabe tambour	Schoendoerffer	Rochefort
Le crabe tambour	Schoendoerffer	Perrin
...		
Vingt Mille Lieues sous les mers	Fleischer	Douglas
Vingt Mille Lieues sous les mers	Fleischer	Mason
Vingt Mille Lieues sous les mers	Fleischer	Lukas

Coordonnées

Salle	Adresse	Téléphone
Pathé	Grenoble 21 Boulevard Maréchal Lyautey	08 92 69 66 96
Le Mélies	Grenoble 28 Allée Henri Frenay	04 76 47 99 31
...		
Les 6 Rex	Grenoble 13 Rue Saint-Jacques	04 76 44 06 82
Ciné Club de Grenoble	Grenoble 4 Rue Hector Berlioz	04 76 44 70 38

Séances

Salle	Titre	Horaire
Pathé	Mais qui a tué Harry ?	18H30
Pathé	Vingt Mille Lieues sous les mers	19H00
...		
Les 6 Rex	Mais qui a tué Harry ?	18H00
Les 6 Rex	Le souper	22H00

Exemple 1 (travail sur deux tables) :

> Quels sont les adresses des salles qui projettent des films entre 18H30 et 19H15 ?

Exemple 2 (travail sur trois tables):

> Quelles sont les adresses des salles et horaires des séances qui projettent des films de Schoendoerffer ?

2 Représentation des données : les fichiers .csv

Nous allons mettre en œuvre nos bases de données dans des fichiers textes les plus simples possibles : les fichiers textes au format csv.

2.1 Le format de fichier csv

Comma-Separated Values, connu sous le sigle CSV, est un format texte ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules, ou un autre caractère de séparation comme le point virgule, la tabulation.

Ouvert signifie que n'importe quel éditeur de texte 'basique' tel notepad++ est capable d'ouvrir et de modifier ces fichiers. Ils peuvent être créés 'à la main' ou bien à partir de libre Office ou bien d'Excel par exemple.

On retrouve dans ces fichiers la première ligne qui sert de descripteur, puis les lignes suivantes qui représentent les données.

Un exemple de format csv :

Fichier au format .csv	Représentation tabulaire

```
Sexe,Prénom,Année de naissance
M,Alphonse,1932
F,Béatrice,1964
F,Charlotte,1988
```

Sexe	Prénom	Année de naissance
M	Alphonse	1932
F	Béatrice	1964
F	Charlotte	1988

Les caractères pouvant servir de séparateur	
Français	Anglais
virgule	comma
point virgule	semicolon
tabulation	Tab ou bien tabulation
espace	space

2.2 Ouverture d'un fichier csv avec python

La bibliothèque csv est mise à contribution. Par défaut le séparateur utilisé est bien la virgule (comma), la gestion du descripteur est transparente.

Lecture en liste de listes

```
import csv

# On créer la liste vide
table=[]

# Lecture de la table csv, le résultat est dans une
# liste de liste
with open('FILM.CSV', encoding="utf8") as myFile:
    reader = csv.reader(myFile,delimiter=';')
    for row in reader:
        table.append(row)

for x in range(len(table)):
    print(table[x])
```


```
>>> (executing lines 1 to 26 of "Lecture_FILM_liste_de_listes.py")
['Titre', 'Directeur', 'Acteur']
['Mais qui a tué Harry ?', 'Hitchcock', 'Gwenn']
['Mais qui a tué Harry ?', 'Hitchcock', 'Forsythe']
['Mais qui a tué Harry ?', 'Hitchcock', 'MacLaine']
['Mais qui a tué Harry ?', 'Hitchcock', 'Hitchcock']
['Cris et chuchotements', 'Bergman', 'Andersson']
['Cris et chuchotements', 'Bergman', 'Sylwan']
```

Pour accéder à un élément il suffit de donner l'index dans la liste puis d'explorer la sous-liste avec les méthodes connues :

```
>>> table[12]
['Vingt Mille Lieues sous les mers', 'Fleischer', 'Lorre']

>>> table[12][0]
'Vingt Mille Lieues sous les mers'

>>> table[12][1]
'Fleischer'

>>> table[12][2]
'Lorre'
```

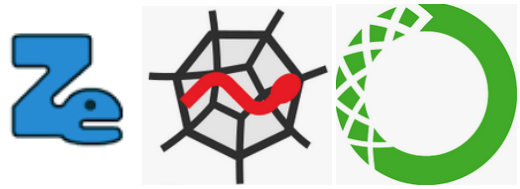
Le descripteur est enregistré en première ligne, soit avec l'index 0 :

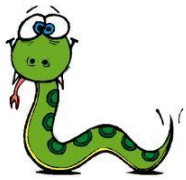
```
>>> table[0]
['Titre', 'Directeur', 'Acteur']
```

Le nombre d'éléments est donné par :

```
>>> len(table)
62
```

Soit un descripteur et 61 enregistrements.






Lecture en liste de dictionnaires

```
import csv

# On créer la liste vide
table=[]

# Lecture de la table csv, le résultat est dans une
# liste de dictionnaires
with open('FILM.CSV', encoding="utf8") as myFile:
    reader = csv.DictReader(myFile,delimiter=';')
    for row in reader:
        table.append(row)

for x in range(len(table)):
    print(table[x])
```

```
>>> (executing lines 1 to 25 of "Lecture_FILM_liste_de_dictionnaires.py")
{'Directeur': 'Hitchcock', 'Acteur': 'Gwenn', 'Titre': 'Mais qui a tué Harry ?'}
{'Directeur': 'Hitchcock', 'Acteur': 'Forsythe', 'Titre': 'Mais qui a tué Harry ?'}
{'Directeur': 'Hitchcock', 'Acteur': 'MacLaine', 'Titre': 'Mais qui a tué Harry ?'}
{'Directeur': 'Hitchcock', 'Acteur': 'Hitchcock', 'Titre': 'Mais qui a tué Harry ?'}
{'Directeur': 'Bergman', 'Acteur': 'Andersson', 'Titre': 'Cris et chuchotements'}
{'Directeur': 'Bergman', 'Acteur': 'Sylwan', 'Titre': 'Cris et chuchotements'}
```

Pour accéder à un élément il suffit de donner l'index dans la liste puis d'explorer le dictionnaire avec les méthodes connues :

```
>>> table[12]
{'Directeur': 'Fleischer', 'Acteur': 'Wilke', 'Titre': 'Vingt Mille Lieues sous les mers'}

>>> table[12]['Directeur']
'Fleischer'

>>> table[12]['Acteur']
'Wilke'

>>> table[12]['Titre']
'Vingt Mille Lieues sous les mers'
```

Le descripteur n'est pas accessible sur une ligne particulière, c'est lui qui a donné les valeurs des différentes clés utilisées et elles sont présentes sur chacune des lignes. On peut en obtenir la liste par la méthode *.keys()* appliquée à n'importe lequel des dictionnaires :

```
>>> table[12]
{'Directeur': 'Fleischer', 'Acteur': 'Wilke', 'Titre': 'Vingt Mille Lieues sous les mers'}

>>> liste_des_cles = table[12].keys()

>>> liste_des_cles
dict_keys(['Directeur', 'Acteur', 'Titre'])
```

Le nombre d'éléments est donné par (il n'y a pas de ligne spécifique pour le descripteur) :

```
>>> len(table)
61
```

2.3 Travailler avec les tables (Liste de listes)

Sélectionner des lignes : sélection σ

On utilise une construction par compréhension de liste :

```
# Sélection de lignes : quels sont les films dirigés par Bergman
# Avec doublons
liste_lignes = [ x for x in table if x[1] == 'Bergman']
```

Comme il y a plusieurs lignes avec 'Bergman' comme directeur puisqu'il y a plusieurs acteurs dans la table Film pour chaque film décrit nous nous retrouvons avec des doublons :

```
Les films dirigés par Bergman :
Cris et chuchotements
Cris et chuchotements
Cris et chuchotements
Cris et chuchotements
```

Pour supprimer les doublons on va utiliser les ensembles de python :

```
# Sélection de lignes : quels sont les films dirigés par Bergman
# sans doublons
liste_lignes_sd = set([ x[0] for x in table if x[1] == 'Bergman'])
```

Les ensembles ne possèdent pas de relation d'ordre aussi on ne peut pas itérer la collection de valeurs avec un index, il faut tester l'appartenance comme ci-dessous :

```
print('Les films dirigés par Bergman sans doublons: ')
for x in liste_lignes_sd:
    print(x)
```

Et voilà le résultat :

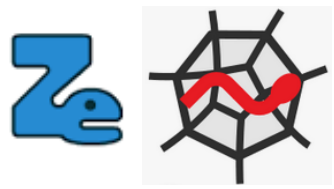
```
Les films dirigés par Bergman sans doublons:
Cris et chuchotements
```

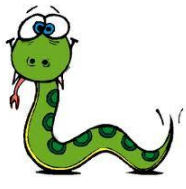
Sélectionner des colonnes : projection π

Effectuons une projection sur la table film en ne conservant que les attributs (colonnes) Directeur et Acteur :

```
# Sélection de colonnes on ignore le descripteur :
liste_colonnes = [[table[x][1],table[x][2]] for x in range (1 , len(table)) ]
```

⇦






Et l'affichage du résultat :

```
print('Table créée par projection sur la table film attribut Directeur, Acteur :')
for x in range(len(liste_colonnes)):
    print(liste_colonnes[x])
```

```
Table créée par projection sur la table film attribut Directeur, Acteur :
['Hitchcock', 'Gwenn']
['Hitchcock', 'Forsythe']
['Hitchcock', 'MacLaine']
```

Fusion de deux tables : jointure

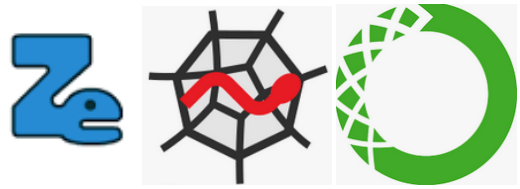
Pour réaliser une mise en commun de deux tables qui possèdent un attribut commun on utilise une fonction jointure telle que ci-dessous :

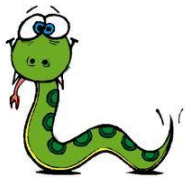
```
# Fusion, jointure de deux tables qui possèdent un attribut commun : attribut_commun
def jointure(table1, table2, attribut_commun):

    # Trouver la position de attribut_commun dans la liste table1
    # dans le descripteur donc l'index [0]
    index1 = table1[0].index(attribut_commun)

    # Trouver la position de attribut_commun dans la liste table2
    # dans le descripteur donc l'index [0]
    index2 = table2[0].index(attribut_commun)


    new_table = []
    for ligne1 in range(0,len(table1)):
        for ligne2 in range(0,len(table2)):
            if table1[ligne1][index1] == table2[ligne2][index2]:
                new_line = table1[ligne1][::]  # deepcopy de la table1
                for x in range(len(table2[0])):
                    if  x != index2:
                        new_line.append(table2[ligne2][x]) # puis ajout des éléments
                new_table.append(new_line)                 # de la table 2
                                                           # sauf l'attribut commun

    return new_table
```

Le codage de l'algorithme utilisé est simple :

On balaye les lignes des deux tables avec deux boucles for imbriquées. Si l'attribut recherché se trouve dans les deux tables alors on recopie entièrement la ligne de la première table et ensuite on ajoute les éléments de la deuxième sauf l'attribut commun qui doit se retrouver au final qu'une seule fois sur chaque ligne.

Le descripteur est traité également il sera présent en première ligne de notre table conformément au travail sur les formats de fichiers csv. Ce qui permet de recréer un fichier .csv si besoin.





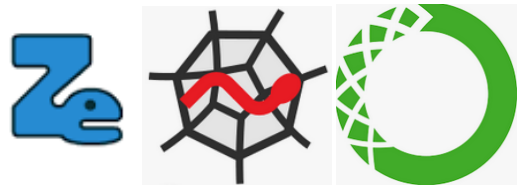

Voilà le code de la lecture des tables et de l'appel de la fonction réalisant la jointure :

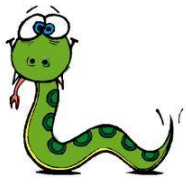
```
# On créer la liste vide
table_coordonnees=[]

# Lecture de la table csv, le résultat est dans une
# liste de listes
with open('COORDONNEES.CSV', encoding="utf8") as myFile:
    reader = csv.reader(myFile,delimiter=';')
    for row in reader:
        table_coordonnees.append(row)

# On créer la liste vide
table_seances=[]

# Lecture de la table csv, le résultat est dans une
# liste de listes
with open('SEANCES.CSV', encoding="utf8") as myFile:
    reader = csv.reader(myFile,delimiter=';')
    for row in reader:
        table_seances.append(row)

table_jointure = jointure(table_coordonnees,table_seances,'Salle')
```

Et l'affichage du résultat :

```
print("Table créée par jointure sur les tables Coordonnees Seances \
avec l'attribut commun Salle")

for x in range(1,len(table_jointure)):
    print(table_jointure[x])
```

```
Table créée par jointure sur les tables Coordonnees Seances avec l'attribut commun Salle
['Pathé', 'Grenoble 21 Boulevard Maréchal Lyautey', '08 92 69 66 96', 'Mais qui a tué Harry ?', '18H30']
['Pathé', 'Grenoble 21 Boulevard Maréchal Lyautey', '08 92 69 66 96', 'Cris et chuchotements', '21H30']
['Pathé', 'Grenoble 21 Boulevard Maréchal Lyautey', '08 92 69 66 96', 'Vingt Mille Lieues sous les mers', '19H00']
['Pathé', 'Grenoble 21 Boulevard Maréchal Lyautey', '08 92 69 66 96', 'Le souper', '18H00']
```






2.4 Travailler avec les tables (Liste de dictionnaires)

Sélectionner des lignes : sélection σ

On utilise une construction par compréhension de liste :

```
# Sélection de lignes : quels sont les films dirigés par Bergman
# avec doublons
liste_lignes = [ x['Titre'] for x in table if x['Directeur'] == 'Bergman']
```

Il suffit ensuite d'afficher la table résultante :

```
Les films dirigés par Bergman avec doublons :
Cris et chuchotements
Cris et chuchotements
Cris et chuchotements
Cris et chuchotements
```

Pour éviter les doublons on va utiliser un ensemble :

```
# Sélection de lignes : quels sont les films dirigés par Bergman
# sans doublons
# utilisation d'un ensemble
liste_lignes_sd = set([ x['Titre'] for x in table if x['Directeur']=='Bergman'])

Les films dirigés par Bergman sans doublons :
Cris et chuchotements
```

Sélectionner des colonnes : projection π

Effectuons une projection sur la table film en ne conservant que les attributs (colonnes) Directeur et Acteur :

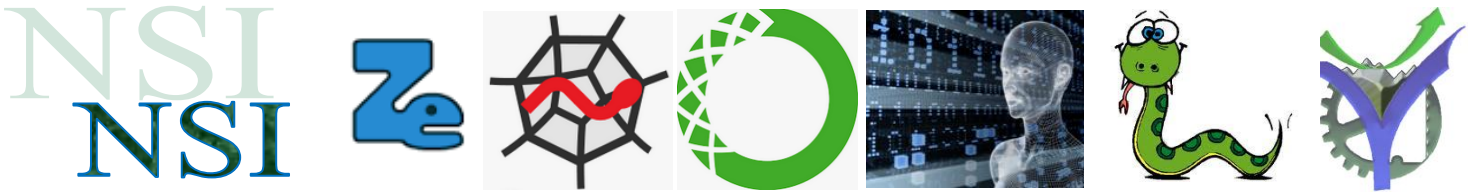
```
# Sélection de colonnes :
liste_colonnes = [{"Directeur":ligne["Directeur"],"Acteur":ligne["Acteur"]} for ligne in table]

print('Table créée par projection sur la table film attribut Directeur, Acteur :')
for x in range(len(liste_colonnes)):
    print(liste_colonnes[x])
```

```
Table créée par projection sur la table film attribut Directeur, Acteur :
{'Directeur': 'Hitchcock', 'Acteur': 'Gwenn'}
{'Directeur': 'Hitchcock', 'Acteur': 'Forsythe'}
{'Directeur': 'Hitchcock', 'Acteur': 'MacLaine'}
```

On peut modifier le code pour préparer l'utilisation d'une fonction dédiée en mettant la liste des attributs recherchés dans une liste dédiée :

```
# Sélection de colonnes solution alternative
# Montre comment utiliser ce principe dans une fonction qui recevrait une liste
# d'attributs
liste_attributs = ['Acteur','Directeur']
liste_colonnes = [{cle:ligne[cle]
                   for cle in ligne
                       if cle in liste_attributs} for ligne in table]
```

⇦



Fusion de deux tables : jointure

Pour réaliser une mise en commun de deux tables qui possèdent une clé commune on utilise une fonction jointure telle que ci-dessous :

```
14  from copy import deepcopy
15
16  # Fusion, jointure de deux tables qui possèdent un attribut commun : cle_commune
17  def jointure(table1, table2, cle_commune):
18      new_table = []
19      for ligne1 in table1:
20          for ligne2 in table2:
21              if ligne1[cle_commune] == ligne2[cle_commune]:
22                  new_line = deepcopy(ligne1)
23                  for cle in ligne2:
24                      new_line[cle] = ligne2[cle]
25                  new_table.append(new_line)
26      return new table
```

Le codage de l'algorithme utilisé est simple :

On balaye les lignes des deux tables avec deux boucles for imbriquées. Si la clé recherchée se trouve dans les deux tables alors on recopie entièrement la ligne de la première table et ensuite on ajoute les éléments de la deuxième.

Pour la copie on utilise deepcopy qui créé une nouvelle copie complète et non pas une nouvelle référence vers le même objet en mémoire.

En supposant les tables Coordonnées et Séances chargées on peut réaliser la jointure :

```
table_jointure = jointure(table_coordonnees,table_seances,'Salle')

print('Table créée par jointure sur les tables Coordonnees Seances avec la clé Salle')
for x in range(len(table_jointure)):
    print(table_jointure[x])
```

```
In [26]: (executing lines 1 to 84 of "Jointure_FILM_liste_de_dictionnaires.py")
Table créée par jointure sur les tables Coordonnees Seances avec la clé Salle
{'Téléphone': '08 92 69 66 96', 'Horaire': '18H30', 'Titre': 'Mais qui a tué Harry ?', 'Adresse': 'Grenoble 21 Boulevard Maréchal Lyautey', 'Salle': 'Pathé'}
{'Téléphone': '08 92 69 66 96', 'Horaire': '21H30', 'Titre': 'Cris et chuchotements', 'Adresse': 'Grenoble 21 Boulevard Maréchal Lyautey', 'Salle': 'Pathé'}
```

Fusion de deux tables exploitation de la jointure

C'est le moment de répondre à la question : Quels sont les adresses des salles qui projettent des films entre 18H30 et 19H15 ?

```
# Sélection de lignes : intervalle 18H30 : 19H15
liste_lignes = [ x for x in table_jointure if x['Horaire'] >= '18H30' and x['Horaire'] <= '19H15' ]
```

On sélectionne les lignes de la table jointure avec une compréhension de liste.

Résultat obtenu : la réponse à la question :

```
Quelles sont les adresses des salles qui projettent des films entre 18H30 et 19H15 ?
Grenoble 21 Boulevard Maréchal Lyautey
Grenoble 21 Boulevard Maréchal Lyautey
Grenoble 28 Allée Henri Frenay
Grenoble Rue du Phalanstére 9 B
Grenoble 13 Rue Saint-Jacques
```

On remarquera les doublons qui sont possibles dans une liste, à cause de la présence de plusieurs films dans le même créneau horaire dans la même salle. Pour l'éviter on utilise des ensembles.

```
# Création de l'ensemble par compréhension
ensemble_lignes = set([ x['Adresse'] for x in table_jointure
                        if x['Horaire'] >= '18H30' and x['Horaire'] <= '19H15' ] )
```

Résultat sans les doublons :

```
Quelles sont les adresses des salles qui projettent des films entre 18H30 et 19H15 ?
Sans les doublons :
Grenoble 21 Boulevard Maréchal Lyautey
Grenoble 28 Allée Henri Frenay
Grenoble Rue du Phalanstére 9 B
Grenoble 13 Rue Saint-Jacques
```

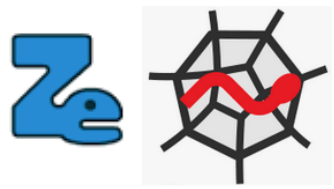
A noter comme les ensembles ne possèdent pas de relation d'ordre on ne peut pas accéder à un élément avec un index. On utilise l'appartenance avec in pour lister les éléments de l'ensemble ensemble_lignes :

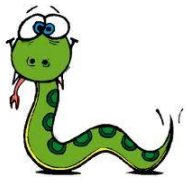
```
# On ne peut pas itérer un ensemble avec un index car il n'y a pas de relation d'ordre
# On test la présence des éléments par in
print('Quelles sont les adresses des salles qui projettent des films entre 18H30 et 19H15 ?')
print('Sans les doublons :')

for x in ensemble_lignes:
    print(x)
```

2.5 Enregistrer les résultats : écriture d'un fichier au format .csv

Le module csv de python dispose de toutes les méthodes nécessaires pour l'écriture de fichiers de données au format csv. Si les données proviennent d'une liste de listes ou bien d'une liste de dictionnaires on procède légèrement différemment.

Nous allons étudier les deux cas sur nos exemples.






Création d'un fichier csv à partir d'une liste de listes

Le codage est très compact :

```
# Création d'un fichier .csv résultat de la jointure
nomFichier = "JOINTURE_TABLE_FROM_LISTE.csv"

with open(nomFichier,'w',encoding="utf8",newline='') as csvfile:
    writer = csv.writer(csvfile, delimiter=';')
    writer.writerows(table_jointure)
```

La première ligne de notre table contient le descripteur donc il suffit de recopier toutes les lignes dans le fichier csv. Le descripteur sera bien placé en première position.

Voilà le début de la table :

```
>>> table_jointure
[['Salle', 'Adresse', 'Téléphone', 'Titre', 'Horaire'], ['Pathé', 'Grenoble 21 B
oulevard Maréchal Lyautey', '08 92 69 66 96', 'Mais qui a tué Harry ?', '18H30']
, ['Pathé', 'Grenoble 21 Boulevard Maréchal Lyautey', '08 92 69 66 96', 'Cris et
 chuchotements', '21H30'], ['Pathé', 'Grenoble 21 Boulevard Maréchal Lyautey', '
08 92 69 66 96', 'Vingt Mille Lieues sous les mers', '19H00'], ['Pathé', 'Grenob
```

On voit bien le descripteur au début. Suivit des enregistrements.

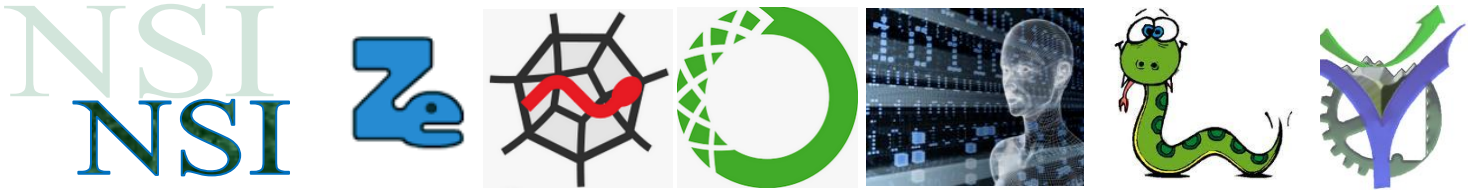
Création d'un fichier csv à partir d'une liste de dictionnaires

Le codage est similaire au codage précédent. Par contre il faut traiter spécifiquement le header à partir d'une liste d'attributs issus des clés du dictionnaire, nous prenons la première ligne de notre liste, mais de ce point de vue elles sont toutes identiques et contiennent toutes les différentes clés.
Il faut ensuite itérer toute la table pour extraire les valeurs.

```
# Création d'un fichier .csv résultat de la jointure
nomFichier = "JOINTURE_TABLE_FROM_DICO.csv"

with open(nomFichier,'w',encoding="utf8",newline='') as csvfile:
    writer = csv.DictWriter(csvfile, delimiter=';', \
                            fieldnames=table_jointure[0].keys())
    writer.writeheader()
    for row in table_jointure:
        writer.writerow(row)
```

```
>>> table_jointure
[{'Téléphone': '08 92 69 66 96', 'Adresse': 'Grenoble 21 Boulevard Maréchal Lyau
tey', 'Horaire': '18H30', 'Salle': 'Pathé', 'Titre': 'Mais qui a tué Harry ?'},
{'Téléphone': '08 92 69 66 96', 'Adresse': 'Grenoble 21 Boulevard Maréchal Lyaut
ey', 'Horaire': '21H30', 'Salle': 'Pathé', 'Titre': 'Cris et chuchotements'}, {'
Téléphone': '08 92 69 66 96', 'Adresse': 'Grenoble 21 Boulevard Maréchal Lyautey
```

⇦



L'utilité de la structure with …. as en python

Il faut toujours faire attention avec les ressources délivrées par le système d'exploitation (OS Operating System) à bien les libérer après utilisation. C'est le cas des fichiers, en effet une fois ouverts ils doivent être fermés correctement. Dans le cas contraire les données peuvent être corrompues, le système de gestion de fichier perturbé avec des demandes de réparation par exemple sur les clés USB enlevées sans précautions etc ….

Illustrons les procédures possibles en python[1]

```
# file handling

# 1) without using with statement
file = open('file_path', 'w')
file.write('hello world !')
file.close()
```

```
# 2) without using with statement
file = open('file_path', 'w')
try:
    file.write('hello world')
finally:
    file.close()
```

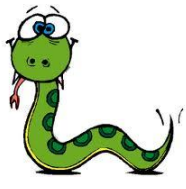
```
# using with statement
with open('file_path', 'w') as file:
    file.write('hello world !')
```

Les deux approches correctes permettant de s'affranchir des conséquences de l'apparition d'une exception pendant l'exécution de la fonction file.write sur la gestion du fichier par l'OS sont encadrées en vert.

La structure with …. as est la plus compacte et est donc utilisée dans nos exemples.

[1] https://www.geeksforgeeks.org/with-statement-in-python/





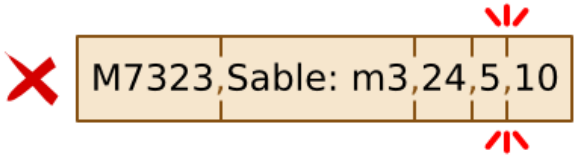

3 Quelques problèmes dans l'utilisation des formats csv et leurs solutions

Ce cours sur les données en table et les fichiers csv ne peut être exhaustif. Pour le compléter il pourra être utile en cas de besoin de consulter les liens mis en ressources et la documentation officielle de python.
Pour ce paragraphe voir ici[2].

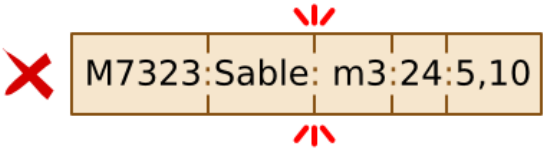
Il peut y avoir des variantes dans l'utilisation des fichiers csv, variantes qui compliquent le traitement automatique.

3.1 Conflit entre les différentes données écrites en textes

Le format CSV utilise traditionnellement la virgule pour séparer les différents champs d'un enregistrement.



Un conflit survient quand la virgule est aussi utilisée comme séparateur décimal.



L'utilisation d'un séparateur de champ alternatif comme le *point-virgule* ou le *deux points* résout le conflit possible avec les données numériques. Mais un conflit reste possible avec le contenu des champs de texte.

L'utilisation de guillemets pour délimiter les données permet de résoudre le conflit lié à la présence du séparateur de champ dans celles-ci.

Pour adapter le comportement de l'interpréteur csv on peut utiliser le dialect pour préciser les valeurs que l'on souhaite. Un résumé des options :

Les attributs d'un `Dialect`

Attribut	Valeur par défaut	Signification
`delimiter`	`,`	Séparateur de champ
`quotechar`	`"`	Délimiteur pour les chaînes
`quoting`	`QUOTE_MINIMAL`	Contrôle l'adjonction de guillemets autour des données
`doublequote`	`True`	Doubler les *quotechar* qui se trouvent dans les données
`escapechar`	`None`	Caractère d'échappement pour protéger les caractères spéciaux dans les données
`lineterminator`	`\r\n`	Séquence de caractères utilisée à la fin de chaque enregistrement
`skipinitialspace`	`False`	Ignorer les espaces entre le délimiteur de chaîne et les données

[2] http://www.chicoree.fr/w/Fichiers_CSV_en_Python

3.2 Mauvaise gestion des sauts de lignes dans le fichier csv généré

Il peut arriver que notre fichier généré ressemble à celui-ci avec des sauts de lignes supplémentaires insérés entre chacune des lignes de données de notre fichier csv :

```
Salle;Adresse;Téléphone;Titre;Horaire

Pathé;Grenoble 21 Boulevard Maréchal Lyautey;08 92 69 66 96;Mais qui a tué Harry ?;18H30

Pathé;Grenoble 21 Boulevard Maréchal Lyautey;08 92 69 66 96;Cris et chuchotements;21H30

Pathé;Grenoble 21 Boulevard Maréchal Lyautey;08 92 69 66 96;Vingt Mille Lieues sous les mers;19H00
```

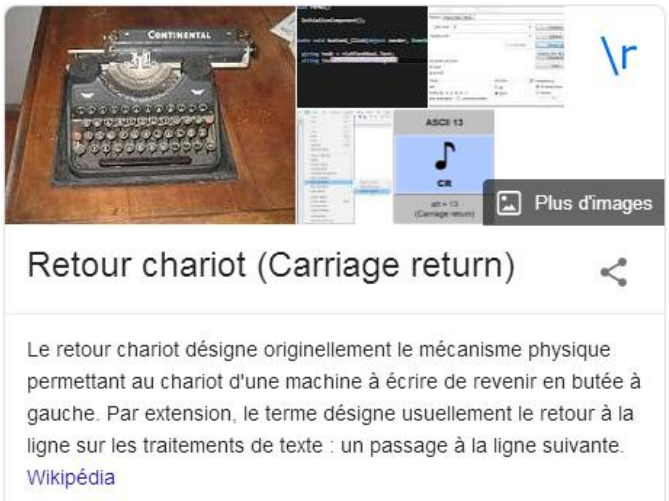
Le problème provient de la différence des fins de lignes des fichiers textes entre les différents systèmes d'exploitation. Linux, MacOS, Windows…

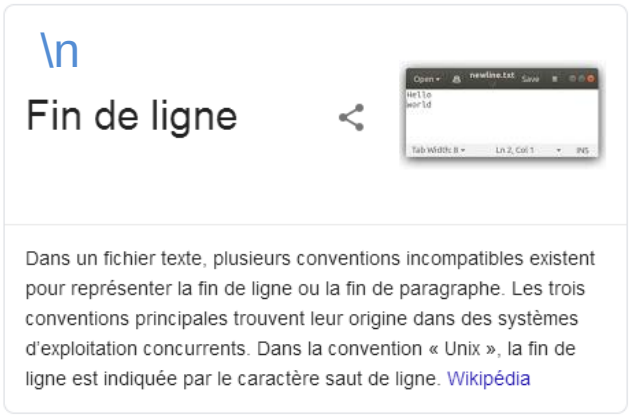
Pour python le séparateur d'enregistrement est donc CR+LF « \r\n »(Carriage Return et Line Feed). En Linux seul LF est utilisé, d'autres OS utilisent seulement CR ….

Bref pour s'en sortir on peut imposer nos propres fins de lignes :

```
newline=''
```

Voir aussi dans le tableau précédent lineterminator.





4 En guise de conclusion

4.1 Alors liste de listes ou listes de dictionnaires ?

A vous de choisir votre préférence. Notons néanmoins que le travail avec les dictionnaires permet d'atteindre les données par les valeurs des clés. Ce qui est plus parlant que des positions d'index dans une liste.

4.2 Vers l'infini et au-delà …. Les bases de données

Après ce chemin dans les tables de données, tables qui structurent l'information entre plusieurs tables chacune représentée par un fichier .csv nous avons découvert que l'extraction d'informations pertinentes de cet ensemble de tables demande un certain travail de compréhension et de codage.

Nous voyons vite les limites de cette façon de faire pour les gros projets contenant beaucoup de données. Ou une organisation qui doit rester évolutive.

C'est pour cela que nous étudierons dans la suite en terminale la définition et l'exploitation directe des bases de données SQL avec des langages d'interrogation dédiés.

4.3 Quelques ressources

Quelques ressources utilisées pour la rédaction de ce cours, que les auteurs soient ici remerciés.

https://stph.scenari-community.org/bdd/0/co/relUC072.html

https://www.fil.univ-lille1.fr/~L1S2API/CoursTP/ensembles_et_dictionnaires.html

http://pauillac.inria.fr/~cheno/taupe/transparents/logique.pdf

http://www.chicoree.fr/w/Fichiers_CSV_en_Python

https://www.geeksforgeeks.org/with-statement-in-python/