

Séquence 5 – Exploiter des données en tableau

Objectifs

1. P-uplets. p-uplets nommés (fonction renvoyant un p-uplet de valeurs)
2. Tableau indexé, tableau, donné en compréhension (tableaux de tableaux pour représenter des matrices : notation a [i] [j], itérer sur les éléments)
3. Dictionnaires, clés et valeurs (EXIF d'une image, méthodes keys(), values() et items())
4. Indexation de tables (Importer une table depuis un fichier texte tabulé ou csv – tableau doublement indexé ou un tableau de p-uplets)
5. Recherche dans une table (lignes vérifiant des critères en logique propositionnelle, doublons)
6. Tri d'une table (suivant une colonne)
7. Fusion de tables

Ce document est la transcription des pages suivantes : https://pixees.fr/informatiquelycee/n_site/nsi_prem_csv.html

1 P-uplets, p-uplets nommées

1.1 Le type list python

Une liste est une séquence mutable/modifiable c'est-à-dire :

- Possibilité d'ajout d'éléments
- Remplacement d'éléments existants
- Tri, inversion

Une liste s'écrit avec des crochets.

```
>>> une_liste = ['un', 2, "trois", 4]
>>> liste_vide = []
>>> sur_plusieurs_lignes = [
    1, 2, 3,
    4, 5, 6
]
>>> list("abcd")
>>> ['a', 'b', 'c', 'd']
>>> une_liste.append(5) # Ajout d'un élément en fin de liste
>>> une_liste
['un', 2, "trois", 4, 5]
>>> toto = une_liste.pop() # Accès au dernier élément (et le
retirer de la liste)
>>> >>> toto
5
>>> une_liste
[un, 2, "trois", 4]
>>> une_liste.insert(0, -1) #Insertion d'un élément à la
position 0
>>> une_liste
[-1, un, 2, "trois", 4]
>>> une_liste[0] = -2 # Remplacement du premier élément
>>> une_liste[2:4] = (0, 3) # Remplacement des éléments 2 et
3
>>> une_liste
[-2, 1, 0, 3, 4]
>>> del une_liste[-2:] # Suppression des deux derniers
éléments

>>> une_liste
[-2, 1, 0]
>>> len(une_liste) # Longueur d'une liste
3
```

ATTENTION ! L' index (ou indice) commence à 0

Une liste est itérable. Cela veut dire que l'on peut utiliser une boucle `for ... in ...` pour parcourir tous ses éléments.

```
>>> une_liste = ['un', 2, "trois", 4]
>>> for i in une_liste :
    print(i)

'un'
2
'trois'
4
```

1.2 Le type tuple python

Un tuple est une séquence ordonnée, non mutable. **On ne peut pas modifier un tuple !**
Un tuple s'écrit avec des parenthèses.

```
>>> un_tuple = (1, "deux", "trois", 4)
>>> un_tuple[0]
1
>>> un_tuple[-1]
4
>>> un_tuple[0] = 'A'
TypeError: 'tuple' object does not support item assignment
```

P. 92 ex. 1

P. 97 ex. 8

P. 100 ex 10 – Partie 1 et partie 2

1.3 Une fonction peut renvoyer un tuple ou une liste

Grâce au tuple, une fonction peut renvoyer plusieurs valeurs.

A faire vous même 1.

- Recopiez, testez et étudiez le code suivant :

```
def add(a, b):
    c = a + b
    return (a, b, c)
mon_tuple = add(5, 8)
print(str(mon_tuple[0]), '+', str(mon_tuple[1]), '=', str(mon_tuple[2]))
print(f"{mon_tuple[0]} + {mon_tuple[1]} = {mon_tuple[2]}")
```

Il faut bien comprendre dans l'exemple ci-dessus que la variable `mon_tuple` référence un tuple (puisque la fonction `add` renvoie un tuple), d'où la "notation entre crochets" utilisée avec `mon_tuple` (`mon_tuple[1]...`)

A faire vous même 2.

Il est possible d'assigner à des variables les valeurs contenues dans un tuple :

```
>>> a, b, c, d = (5, 8, 6, 9)
>>> a
5
>>> b
8
```

- Quelle est la valeur référencée par la variable `a` ? La variable `b` ? La variable `c` ? La variable `d` ? Vérifiez votre réponse à l'aide de la console Python.
- Reprenez la fonction `add` ci-dessus et affectez les 3 valeurs retournées à 3 variables différentes en une seule commande

2 Créer un tableau par compréhension

Nous avons vu qu'il était possible de "remplir" un tableau en renseignant les éléments du tableau les uns après les autres :

```
>>> mon_tab = [5, 8, 6, 9]
```

ou encore à l'aide de la méthode "append" :

```
>>> mon_tab2 = []
>>> mon_tab2.append(55)
>>> mon_tab2.append(66)
>>> mon_tab2.append(77)
```

Il est aussi possible d'obtenir exactement le même résultat en une seule ligne grâce à la compréhension de tableau.

A faire vous même 3.

Quel est le contenu du tableau référencé par la variable `mon_tab` après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
>>> mon_tab = [p for p in range(5, 10)]
```

.....

A faire vous même 4.

Quel est le contenu du tableau référencé par la variable `mon_tab` après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
>>> l = [1, 7, 9, 15, 5, 20, 10, 8]
>>> mon_tab = [p for p in l if p > 10]
```

.....

Les compréhensions de tableau permettent donc de rajouter une condition (if).

A faire vous même 5.

Quel est le contenu du tableau référencé par la variable `mon_tab` après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
>>> ma_liste = [1, 7, 9, 15, 5, 20, 10, 8]
>>> mon_tab = [p**2 for p in ma_liste if p < 10]
```

Rappel : `p**2` permet d'obtenir la valeur de `p` élevée au carré

.....
Comme vous pouvez le remarquer, nous obtenons un tableau (`mon_tab`) qui contient tous les éléments du tableau `ma_liste` élevés au carré à condition que ces éléments de `l` soient inférieurs à 10.

Comme vous pouvez le constater, la compréhension de tableau permet d'obtenir des combinaisons relativement complexes.

P. 93 ex. 3

P. 95 ex. 6

P. 97 ex. 8

3 Listes de listes ou Tableaux

A faire vous même 6.

Quelle est la valeur référencée par la variable `a` après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
m = [['a', 'b', 'c'],
      ['d', 'e', 'f'],
      ['g', 'h', 'i'],
      ['j', 'k', 'l']]
a = m[1][2]
```

.....
Comme vous pouvez le constater, la variable `a` référence bien le caractère situé à la 2e ligne (indice 1) et à la 3e colonne (indice 2), c'est-à-dire 8.

A faire vous même 7.

Quel est le contenu du tableau référencé par la variable `mm` après l'exécution du programme ci-dessous ? (utilisez la console pour vérifier votre réponse)

```
>>> m = [1, 2, 3]
>>> mm = [m, m, m]
>>> m[0] = 100
```

Comme vous pouvez le constater, la modification du tableau référencé par la variable m entraîne la modification du tableau référencé par la variable mm (alors que nous n'avons pas directement modifié le tableau référencé par mm). Il faut donc être très prudent lors de ce genre de manipulation afin d'éviter des modifications non désirées.

Il est possible de parcourir l'ensemble des éléments d'une matrice à l'aide d'une "double boucle for" :

A faire vous même 8.

- Analysez puis testez le code suivant :

```
m = [['a', 'b', 'c'],
      ['d', 'e', 'f'],
      ['g', 'h', 'i'],
      ['j', 'k', 'l']]
nb_colonne = 3
nb_ligne = 4
for i in range(0, nb_ligne):
    for j in range(0, nb_colonne):
        a = m[i][j]
        print(a)
```

Lire P. 88-89 (sauf Dictionnaires en compréhension)

P. 96 ex 7

4 Le type dictionnaire en python

Lire P. 86-87

Un dictionnaire est un objet contenant d'autres objets qui comme c'est le cas dans les listes mais sans que des objets ne soient classés par indice mais par clés.

lien dictionnaire :

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

Un dictionnaire s'écrit avec des accolades.

Prenons par exemple les coordonnées d'une personne (nom, prénom, tel,). Toutes ces informations peuvent être enregistrées dans un dictionnaire .

```
>>> Mes_Coordonnees = {}
>>> Mes_Coordonnees["Nom"] = "DUPONT" # ajoute une clé au dictionnaire
>>> Mes_Coordonnees["Prenom"] = "Thomas"
>>> Mes_Coordonnees["Code Postal"] = 35400
>>> del Mes_Coordonnees["Code Postal"] # supprimer une clé du dictionnaire
>>> print( Mes_Coordonnees)
{'Nom': 'DUPONT', 'Prenom': 'Thomas'}
>>> Mes_Coordonnees.keys() # afficher la liste des clés
dict_keys(['Nom', 'Prenom'])
>>> for key in Mes_Coordonnees : # Itération
...     print ( "la clé est : ",key, " et la valeur de cette clé est : ", Mes_Coordonnees[key])
```

P. 92 ex 2

P. 94 ex 5

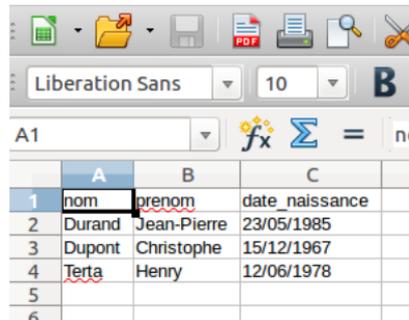
P. 98 ex 9

5 Données structurées – Les fichiers csv

On trouve énormément de données sur internet. Une partie de ces données sont publiques, par exemple le site <https://www.data.gouv.fr/fr/> recense un grand nombre de données publiques. Ces données sont librement réutilisables.

<p>A faire vous même 9.</p> <p>Afin de découvrir ce qu'est "l'open data", allez sur le site https://www.data.gouv.fr/fr/. Lisez la page suivante : https://www.data.gouv.fr/fr/pages/about/a-propos_data-gouv/ A quoi servent les données ouvertes ?</p> <p>.....</p>	<input type="checkbox"/>
<p>A faire vous même 10.</p> <p>Explorez pendant quelques minutes le site https://www.data.gouv.fr/fr/. Recherchez les données "Opérations coordonnées par les CROSS" à l'aide du moteur de recherche proposé par le site Vous pouvez constater que ces données sont au format CSV</p>	<input type="checkbox"/>
<p>5.1 Le format csv</p> <p>Le format CSV est très courant sur internet. Voici ce que nous dit Wikipédia sur le format CSV : <i>Comma-separated values, connu sous le sigle CSV, est un format informatique ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules.</i> Un fichier CSV est un fichier texte, par opposition aux formats dits « binaires ». Chaque ligne du texte correspond à une ligne du tableau et les virgules correspondent aux séparations entre les colonnes. Les portions de texte séparées par une virgule correspondent ainsi aux contenus des cellules du tableau. Voici un exemple du contenu d'un fichier CSV :</p> <pre style="background-color: #e6f2ff; padding: 10px;">nom,prenom,date_naissance Durand,Jean-Pierre,23/05/1985 Dupont,Christophe,15/12/1967 Terta,Henry,12/06/1978</pre> <p>Je pense qu'il est évident pour vous que nous avons ici 3 personnes :</p> <ul style="list-style-type: none"> • Jean-Pierre Durand qui est né le 23/05/1985 • Christophe Dupont qui est né le 15/12/1967 • Henry Terta qui est né le 12/06/1978 <p>nom, prenom et date_naissance sont appelés des descripteurs alors que, par exemple, Durand, Dupont et Terta sont les valeurs du descripteur nom.</p>	<input type="checkbox"/>
<p>A faire vous même 11.</p> <ul style="list-style-type: none"> • Téléchargez le fichier correspondant avec des virgules : http://ninoo.fr/LC21_22/NSI_1ere/seq5_donnees_en_tableau/ident_virgule.csv • Ou avec des point-virgules : http://ninoo.fr/LC21_22/NSI_1ere/seq5_donnees_en_tableau/ident_pointVirgule.csv • Ouvrez le fichier téléchargé avec un éditeur de texte. • Donnez les différentes valeurs du descripteur date_naissance 	<input type="checkbox"/>
<p>ATTENTION :</p> <p>La virgule est un standard pour les données anglo-saxonnes, mais pas pour les données aux normes françaises. En effet, en français, la virgule est le séparateur des chiffres décimaux. Il serait impossible de différencier les virgules des décimaux et les virgules de séparation des informations. C'est pourquoi on utilise un autre séparateur : le point-virgule (;). Dans certains cas cela peut engendrer quelques problèmes, vous devrez donc rester vigilants sur le type de séparateur utilisé. Les tableurs, tels que "Calc" (Libre Office), sont normalement capables de lire les fichiers au format CSV.</p>	<input type="checkbox"/>
<p>A faire vous même 12.</p> <ul style="list-style-type: none"> • Ouvrez ce dernier à l'aide du tableur LibreOffice Calc. <p>Si par hasard votre tableur ne gère pas correctement le fichier avec le séparateur "point-virgule", utiliser le "séparateur virgule". Dans la suite, gardez toujours cet éventuel problème à l'esprit (surtout avec des données "made in France")</p>	<input type="checkbox"/>

Vous devriez obtenir ceci :



	A	B	C
1	nom	prenom	date_naissance
2	Durand	Jean-Pierre	23/05/1985
3	Dupont	Christophe	15/12/1967
4	Terta	Henry	12/06/1978
5			
6			

Vous pouvez constater que les données sont bien "rangées" dans un tableau avec des lignes et des colonnes (voilà pourquoi on parle de données tabulaires).

Il est possible de trouver sur le web des données beaucoup plus intéressantes à traiter que celles contenues dans le fichier "ident_pointVirgule.csv" (ou "ident_virgule.csv"). Par exemple, le site sql.sh, propose un fichier csv contenant des informations sur l'ensemble des communes françaises ou le [fichier sur les prénoms français](#).

Autre exemple : Le [site sur les données du covid](#) qui fournit les données brutes sur les hospitalisation liées au Covid-19. Pas de manipulation, pas de fake news, ce sont les données officielles, authentiques..

A faire vous même 13.

Ouvrez le fichier http://ninoo.fr/LC/1ere_NSI/seq5_donnees_en_tableau/villes_point_virgule.csv à l'aide d'un tableur (c'est une version légèrement modifiée de celle disponible sur le site sql.sh, j'y ai notamment ajouté des entêtes). En cas de problème avec votre tableur, voici une version "séparateur virgule" : http://ninoo.fr/LC/1ere_NSI/seq5_donnees_en_tableau/villes_virgule.csv (attention le séparateur "décimal" est ici le point)

Comme vous pouvez le constater, nous avons 12 colonnes (et 36700 lignes si on ne compte pas l'entête !), voici la signification de ces colonnes :

- dep : numéro de département
- nom : nom de la commune
- cp : code postal
- nb_hab_2010 : nombre d'habitants en 2010
- nb_hab_1999 : nombre d'habitants en 1999
- nb_hab_2012 : nombre d'habitants en 2012 (approximatif)
- dens : densité de la population (habitants par kilomètre carré)
- surf : superficie de la commune en kilomètre carré
- long : longitude
- lat : latitude
- alt_min : altitude minimale de la commune (il manque des données pour certains territoires d'outre-mer)
- alt_max : altitude maximale de la commune (il manque des données pour certains territoires d'outre-mer)

A faire vous même 14.

- En vous aidant du fichier ouvert dans le "À faire vous-même 5", déterminez l'altitude maximale et l'altitude minimale de votre commune.

5.2 Le traitement de données structurées csv avec python

Après avoir découvert le format CSV, nous allons maintenant, à l'aide de Python, apprendre à effectuer des traitements sur ces données.

5.2.1 Enregistrements

Un enregistrement est une structure de données, de types éventuellement différents, auxquelles on accède grâce à un nom.

Exemple :

On peut représenter les notes d'un élève dans différentes disciplines à l'aide d'un enregistrement :

```
{'Nom': 'Joe', 'Anglais': '17', 'Info': '18', 'Maths': '16'}
```

Les champs (ou clés ou attributs) de ces enregistrements sont ici Nom, Anglais, Info et Maths. On leur associe des valeurs, ici 'Joe', '17', '18' et '16'.

En Python, on utilisera les dictionnaires pour représenter les enregistrements conformément au programme.

Voici l'équivalent en tableau et au format csv de l'enregistrement ci-dessus.

	A	B	C	D
1	Nom	Anglais	Info	Maths
2	Joe	17	18	16
3	Zoé	15	17	19
4	Max	19	13	14

Le fichier CSV correspondant est :

```
'Nom','Anglais','Info','Maths'  
'Joe','17','18','16'  
'Zoé','15','17','19'  
'Max','19','13','14'
```

5.2.2 Lecture de fichiers csv

On peut choisir de représenter en Python les fichiers CSV par des listes de dictionnaires dont les clés sont les noms des colonnes.

Avec l'exemple précédent, on définit la liste :

```
Table1 =  
[{'Nom': 'Joe', 'Anglais': '17', 'Info': '18', 'Maths': '16'},  
{'Nom': 'Zoé', 'Anglais': '15', 'Info': '17', 'Maths': '19'},  
{'Nom': 'Max', 'Anglais': '19', 'Info': '13', 'Maths': '14'}]
```

À NOTER

En utilisant le vocabulaire habituel décrivant une feuille de calcul d'un tableur :

- une table est une liste de dictionnaires, ici `Table1` ;
- une ligne est un dictionnaire, ici `Table1[0]` ;
- une cellule est une valeur associée à une clé d'un dictionnaire, ici `Table1[0]['Info']`.

5.2.3 Import d'un fichier CSV

Pour l'import, on crée une liste de dictionnaires (un par ligne de la table).

La première ligne du fichier CSV est considérée comme la ligne des noms des attributs. fichier est une chaîne de caractères donnant le nom du fichier sans son extension. Par exemple depuis `csv('Ma_Base')` pour charger le fichier `Ma_Base.csv`.

```
import csv  
def depuis_csv(fichier) :  
    mon_fichier = open(fichier + '.csv', 'r')  
    lecteur = csv.DictReader(mon_fichier)  
    ma_liste = []  
    for ligne in lecteur :  
        dico_ligne = dict(ligne)  
        ma_liste.append(dico_ligne)  
    return ma_liste
```

La fonction python `dict` est nouvelle. Vous pouvez la tester dans votre console python. Voici un exemple :

```
>>> dict([('one', 1), ('two', 2)])  
{'one': 1, 'two': 2}
```

Voici le code en plus condensé :

```
import csv  
def depuis_csv(fichier) :  
    lecteur = csv.DictReader(open(fichier + '.csv', 'r'))  
    return [dict(ligne) for ligne in lecteur]
```


5.3 Opérations sur les tables

Une fois que l'on dispose des données en table, on a accès à toute la palette de commandes du langage de programmation utilisé.

On peut donc créer des outils de manipulation des tables, comme la recherche et le tri.

5.3.1 Sélection de lignes vérifiant un critère

On cherche à créer une nouvelle table en extrayant d'une table les lignes satisfaisant une condition donnée sous la forme d'une fonction booléenne.

MOT CLÉ - Les opérateurs booléens habituels sont :

`<`, `>`, `<=`, `>=`, `==`, `!=`, `in`, `not`, `and`, `or`, `is...`

Pour illustrer cette recherche, on crée la fonction `select` ci-dessous qui prends en paramètre une table (table, une liste de dictionnaires) et un critère de sélection (critère) sous forme d'une chaîne de caractères contenant un test booléen prenant une ligne en argument. La fonction renvoie une liste construite par compréhension avec un filtre qui ne contient que les lignes de la table qui satisfont le critère donné en argument.

```
def select(table, critère):
    def test(ligne):
        return eval(critère)
    return [ligne for ligne in table if test(ligne)]
```

La fonction python `eval` est nouvelle. Elle prend une chaîne de caractères et retourne un booléen. Vous pouvez la tester dans votre console python. Voici un exemple :

```
>>> eval("2==2")
True
>>> eval("2==3")
False
```

A faire vous même 17.

Reprenons le tableau précédent.

	A	B	C	D
1	Nom	Anglais	Info	Maths
2	Joe	17	18	16
3	Zoé	15	17	19
4	Max	19	13	14

Pour sélectionner les élèves ayant plus de 16 en maths, on utilise la fonction `eval` qui permet d'évaluer l'expression contenue dans la cellule `ligne['Maths']` sous forme d'une chaîne de caractères en un entier.

1. Saisissez la fonction ci-dessus
2. Saisissez la `Table1` ci-dessus
3. Exécutez pour importer fonction et table
4. Tapez la commande suivante :

```
>>> select(Table1, "ligne['Maths'] > 16")
```

Et on obtient bien uniquement la ligne satisfaisant le critère :

```
[{'Nom': 'Zoé', 'Anglais': '15', 'Info': '17', 'Maths': '19'}]
```

5.3.2 Sélection de colonnes

Pour sélectionner un ou plusieurs attributs (colonnes) d'une table (cette opération s'appelle « projection » dans le langage des bases de données), on va créer une nouvelle table qui ne contiendra que ces attributs :

```
def projection(table, liste_attributs):
    ma_liste=[ ]
    for ligne in table :
        mon_dico = {
```

```

for clé in ligne :
    if clé in liste_attributs :
        mon_dico[clé] = ligne[clé]
ma_liste.append(mon_dico)
return ma_liste

```

Voici le code en plus condensé :

```

def projection(table, liste_attributs):
    return [{clé:ligne[clé] \
            for clé in ligne if clé in liste_attributs}\
            for ligne in table]

```

A faire vous même 18.

Dans notre exemple, on veut ne retenir que les notes d'info et le nom des élèves.

1. Saisissez la fonction ci-dessus
2. Saisissez la Table1 ci-dessus
3. Exécutez pour importer fonction et table
4. Tapez la commande suivante :

Pour obtenir la table attendue, on entre :

```
>>> projection(Table1, ['Nom', 'Info'])
```

On obtient :

```

[{'Nom': 'Joe', 'Info': '18'},
 {'Nom': 'Zoé', 'Info': '17'},
 {'Nom': 'Max', 'Info': '13'}]

```

Qui modélise le tableau :

Nom	Info
Joe	18
Zoé	17
Max	13

5.3.3 Tri d'une table selon une colonne

Puisqu'une table est représentée par une liste, on peut la trier en utilisant la fonction de tri `sorted` qui dispose d'un argument `key` permettant de préciser selon quel critère une liste doit être triée (qui doit être une fonction de variables les objets à trier). Un troisième argument, `reverse` (un booléen), permet de préciser si l'on veut le résultat par ordre croissant (par défaut) ou décroissant (en précisant `reverse=True`).

On peut alors créer une fonction `tri` qui trie n'importe quelle table en donnant l'attribut choisi pour le tri et en précisant si l'on veut obtenir le tri dans l'ordre décroissant.

```

def tri(table, attribut, decroit=False):
    def critère(ligne):
        return ligne[attribut]
    return sorted(table, key=critère, reverse=decroit)

```

Exemple :

A faire vous même 19.

Dans notre exemple, on veut ne retenir que les notes d'info et le nom des élèves.

1. Saisissez la fonction ci-dessus
2. Saisissez la Table1 ci-dessus
3. Exécutez pour importer fonction et table
4. Tapez la commande suivante :

Pour trier dans l'ordre décroissant la table Table1 selon les notes de maths, on fera :

```
>>> tri(Table1, 'Maths', True)
```

Qui donne :

	Nom	Maths	Info	Anglais
0	Zoé	19	17	15
1	Joe	16	18	17
2	Max	14	13	19

A faire vous même 20.

2 Opérations sur les tables

→ FICHES 18 et 19

On dispose de la table T représentant les notes d'élèves dans trois matières :

Nom	Maths	Anglais	Informatique
Joe	16	17	18
Zoé	19	15	17
Max	14	19	13

1. Pour sélectionner des colonnes selon un critère donné, laquelle des fonctions définies → FICHE 18 utiliserait-on ?

- a. select b. projection

2. Selon sa définition → FICHE 18, `select(T, "'17' in ligne.values()")` renvoie une table :

- a. vide b. avec une ligne
 c. avec deux lignes d. avec trois lignes

3. Soit U la table suivante :

Nom	Âge	Courriel
Joe	16	joe@info.fr
Zoé	15	zoe@info.fr

Selon sa définition → FICHE 19, `jointure(T, U, 'Nom')` renvoie une table ayant :

- a. 3 lignes b. 2 lignes c. 6 colonnes
 d. 5 colonnes e. 7 colonnes f. 4 colonnes

A faire vous même 21.

3 Déterminer des fonctions basiques

→ FICHE 17

1. Déterminer une fonction qui calcule la cardinalité d'une table, c'est-à-dire son nombre de lignes.

2. Déterminer une fonction qui renvoie la liste des attributs d'une table.

5.4 Jointures de tables

Lorsque l'on traite de grandes quantités de données, celles-ci sont souvent réparties dans plusieurs tables. On est donc souvent amené à regrouper des données dans une nouvelle table. Cette opération s'appelle la jointure de tables.

5.4.1 Fusion de deux tables pour un même attribut

On veut fusionner deux tables selon un attribut commun. On va sélectionner dans chaque table la ligne ayant la même valeur pour l'attribut choisi.

Reprenons le tableau précédent.

	Nom	Maths	Anglais	Info
0	Joe	16	17	18
1	Zoé	19	15	17
2	Max	14	19	13

Définissons une seconde table Table2 donnant l'âge et le courriel de certains élèves :

	Nom	Âge	Courriel
0	Joe	16	joe@info.fr
1	Zoé	15	zoe@info.fr

On voudrait regrouper les données des deux tables. Elles ont l'attribut Nom en commun.

On veut obtenir la table suivante :

	Nom	Âge	Courriel	Maths	Info	Anglais
0	Joe	16	joe@info.fr	16	18	17
1	Zoé	15	zoe@info.fr	19	17	15

On choisit d'exclure la ligne concernant Max car il n'est pas présent dans la seconde table. On effectuera la jointure selon le nom avec la commande :

```
>>> jointure(Table1, Table2, 'Nom')
```

On utilise ici une fonction jointure définie pour l'occasion, comme celle de la page suivante.

5.4.2 La fusion de deux tables pour des attributs différents

Cependant, dans certaines tables, l'attribut commun peut avoir une autre appellation.

Par exemple, la seconde table peut aussi exister sous la forme :

	Name	Age	Email	Maths	CS	English
0	Joe	16	joe@info.fr	16	18	17
1	Zoé	15	zoe@info.fr	19	17	15

Cette fois, on précisera l'attribut de la seconde table :

```
>>> jointure(Table1, Table2, 'Nom', 'Name')
```

5.4.3 Exemple de fonction effectuant une jointure

Voici une proposition de code :

```
1 from copy import deepcopy
2 def jointure(table1, table2, cle1, cle2=None):
3     if cle2 is None:
4         cle2 = cle1
5     new_table = []
6     for ligne1 in table1:
7         for ligne2 in table2:
8             if ligne1[cle1] == ligne2[cle2]:
9                 new_line = deepcopy(ligne1)
10                for cle in ligne2:
11                    if cle != cle2:
12                        new_line[cle] = ligne2[cle]
13                new_table.append(new_line)
14 return new_table
```

Ligne 3 : par défaut, les clés de jointure portent le même nom.

Ligne 5 : la future table créée, vide au départ.

Ligne 8 : on ne considère que les lignes où les cellules de l'attribut choisi sont identiques.

Ligne 9 : on copie entièrement la ligne de table1.

Ligne 10 : on copie la ligne de table2 sans répéter la cellule de jointure.

À NOTER

En terminale, vous découvrirez la gestion des bases de données relationnelles, notamment à l'aide du langage SQL. Dans ce langage, la jointure donnée en exemple s'écrira :

```
SELECT Nom
```

```
FROM Table1 JOIN Table2
ON Table1.Nom = Table2.Nom
```

A faire vous même 22.

8 Sélectionner, trier, joindre

→ FICHES 18 et 19

On dispose de la table BaseAgents :

	NomAgent	VilleAgent
0	Branno	Terminus
1	Darell	Terminus
2	Demerzel	Uco
3	Seldon	Terminus
4	Dornick	Kalgan
5	Hardin	Terminus
6	Trevize	Hesperos
7	Pelorat	Kalgan
8	Riose	Terminus

On a aussi la table BaseGardiens :

	NoCabine	NomAgent
0	1	Branno
1	2	Darell
2	3	Demerzel
3	4	Seldon
4	5	Dornick
5	6	Hardin
6	7	Trevize
7	8	Pelorat
8	9	Riose

1. Renvoyer BaseTerminus, une table extraite de BaseAgents ne contenant que les lignes dont l'attribut VilleAgent vaut « Terminus ».
2. Renvoyer BaseAlpha, une table dérivée de BaseAgents triée selon l'ordre alphabétique du nom des agents.
3. Renvoyer BaseComplete, la table contenant le numéro de cabine, la ville et le nom de l'agent.
4. Renvoyer BaseVille, la table contenant le numéro de cabine et la ville des agents.
5. Renvoyer BaseImpair, la table contenant le nom et la ville des agents ne venant pas de Terminus et dont le numéro de cabine est impair.



À NOTER

On utilisera les fonctions introduites dans les fiches 16 à 18. Les tables seront données sous forme de listes de dictionnaires.