CHAPITRE IV
Traitement base de données

1 Bref rappel lecture de fichier

1.1 Les répertoires

Via la bibliothèque os :

```
>>>import os
>>>os.getcwd()# Pour connaitre le répertoir courant
>>>os.chdr('C:\\users\\nouveau') #pour changer de répertoir courant
```

1.2 Ouvrir un fichier texte en python

Il est utile (surtout pour récupérer des données en grand nombre) de savoir ouvrir un fichier pour le rapatrier sous python. On peut ouvrir un fichier pour le lire (read), ou écrire dedans (write). Pour ça on utilise la commande open et on donne un petit nom au fichier, par exemple pour lire le fichier coucou :

```
>>> nom_fichier=open('coucou.txt','r') # 'r' pour read
```

On peut ensuite extraire une ligne du fichier grâce à la commande readline(), toutes les ligne avec readlines(). Remarque que dans le cas présent le fichier est par défaut dans C :/Pythonxx, si on cherche un fichier ailleurs on indique l'adresse en entier : 'C :/mon dossier/coucou.txt'

```
>>> nom_fichier.readline()#affiche la ligne lu
>>>nom_fichier.readlines()# affiche le fichier dans son ensemble.
```

En fait le fichier est vu comme un tableau de chaines de caractères. Attention à ne pas oublier de fermer le fichier sinon il est garder en mémoire et prend de la place.... Voici un petit programme indiquant la taille (le nombre de caractères) du fichier.

```
>>> nom_fichier=open('coucou.txt','r')
      for L in nom_fichier:
            s=s+len(L)
        nom_fichier.close()
```

A noter que chaque ligne est terminer par le caractère 'n' (qui indique le retour à la ligne) qui est aussi comptabiliser comme un caractère.
Pour écrire des données dans un nouveau fichier on utilise la commande 'w' (write) au lieu de 'r'

```
            >>> fichier2=open('coucouc2.txt',w)
                  for L in nom_fichier:
                  fichier2.write(L)# on recopie ligne par ligne le fichier précédant
                                .
                  fichier2.close()# pour enregistrer
```

Pour compléter (ajouter une ligne) un fichier existant :

```
>>>fichier=open('coucou.txt',a)
fichier.write(\nOn\trigole\tbien\n)#\n début de la ligne et saut de ligne,
              \t tabulation)
```

2 Plus en détail : La gestion des fichiers sous Python.

Il est utile de pouvoir ouvrir, modifier, créer, écrire, réécrire un fichier pour récupérer des données en grand nombre ou pour traiter une image, un son, ...

2.1 Les répertoires.

Les commandes qui suivent doivent être lancées après `import os`.

- savoir quel est le répertoire courant : `os.getcwd()`
- connaître le contenu du répertoire courant : `os.system('truc')` où `truc` est la commande système de l'O.S. [1] sous lequel **Python** est exécuté [2].
- changer de répertoire courant : `os.chdir('chemin_du_repertoire')`

2.2 Les fichiers.

Pour ouvrir un fichier, il faut utiliser la commande `open`, qui renvoie une « interface-fichier » à stocker dans une variable, dont la syntaxe est la suivante :

```
open(fichier,'mode','code')
```

où `mode` et `code` sont optionnels [3]. Si le fichier `fichier` n'est pas dans le répertoire courant, il faut indiquer explicitement le chemin d'accès.

```
f=open('C:\\Users\\eleve\\Documents\\Python\\essai.py') #sous DOS/Windows
```

Les différents `mode` sont :

- `r` : ouverture en lecture seule (**r***ead only*)
- `w` : ouverture en écriture seule (**w***rite only*)
- `r+` : ouverture en mode lecture/écriture
- `x` : ouverture en mode création
- `a` : ouverture en mode **a***ppend* (écriture à la fin du fichier existant)
- `t` : mode **t**ext
- `b` : mode **b**inaire

2.2.1 Lecture.

On suppose ici que notre interface-fichier s'appelle `f`.

- `f.read(n)` :
  - lit le fichier jusqu'au caractère n (le premier caractère est 0 !!) et retourne ces n caractères sous forme de chaîne
  - en cas de répétition de l'instruction, la lecture se poursuit depuis la dernière position précédente
  - si l'argument est omis, tout le fichier est lu
  - s'il n'y a plus rien à lire, retourne `''`
- `f.tell()` : retourne la position courante
- `f.seek(n)` : se place à la position n.

1. *Operating System* ou système d'exploitation
2. ainsi, toutes les commandes systèmes peuvent être effectuées à partir de **Python**, effacement de fichiers, de répertoires, changement de répertoire, etc. : DANGER !!!!
3. valeurs par défaut : `rt` et `UTF-8`

— `f.readline()` :
  — retourne les caractères jusqu'au prochain `\n` qui marque une fin de ligne
  — en cas d'appel ultérieur, retourne la ligne suivante
  — s'il n'y a plus rien à lire, retourne `''`
— `f.readlines()` : retourne toutes les lignes du fichiers, chacune sous forme de chaîne

On peut aussi utiliser la syntaxe :

```
for ligne in f: # o\'u f=open(...)
    instruction
    ....
    ...
```

Remarque. Ne pas ouvrir un fichier binaire en mode texte car il y a un **énorme** risque de confusion dans les caractères de fin de ligne `\n`.

Remarque.
— `\n` est un saut de ligne
— `\t` est une tabulation
— `\b` est un « backspace »
— `\a` est un « bip »
— `\'` est un « ' », mais il ne ferme pas la chaine de caractères
— `\"` est un « " », mais il ne ferme pas la chaine de caractères
— `\\` est un « \ »

2.2.2 Écriture.

La commande `f.write('qqchose')` écrit à partir de la position courante la chaîne de caractère `'qqchose`.

Remarque. Il ne s'agit pas d'une insertion de texte mais d'un remplacement !!

```
>>> f=open('test.py','r+')
>>> f.read()
'ligne2\n'
>>> f.seek(1)
1
>>> f.write('!')
1
>>> f=open('test.py','r+')
>>> f.read()
'l!gne2\n'
>>>
```

2.2.3 Fermeture.

Il est préférable de fermer un fichier lorsqu'il n'est plus manipulé (cela évite de le modifier ou de le supprimer par mégarde) avec la commande : `f.close()`.

Remarque. La commande `f.closed()` retourne un booléen [4] qui indique si le fichier `f` est fermé ou non.

On peut aussi se servir de l'instruction `with ... as ...` qui fermera automatiquement le fichier après l'exécution des instructions.

2.3 Les fichiers images.

On peut représenter une image (noire et blanc ou couleur) grâce à une matrice ou chaque élément a_{ij} représente une case ou un pixel de cordonnée (i, j) qui est de couleur la valeur a_{ij} qui est un nombre entier entre 0 et 255 :

4. voir ?? page ??

```
>>> import PIL.Image
>>> photo=PIL.Image.open('photo.png') # ouvre le fichier photo sous Python
>>> photo.show() # affiche l'image>>> photo.size # indique la taille
>>>photo=PiL.image.new(mode,size,color=0)# creer une image, size=(largeur,hauteur),
                                                    #mode=1 pour un tableau
                                                          de pixel
```

Il faut, pour travailler sur une photo, d'abord la transformer en tableau (matrice). On utilise, pour cela, la bibliothèque `numpy` :

```
>>> import numpy
>>> phototab=numpy.array(photo) # Transforme la photo en tableau
>>> photo2=Image.fromarray(phototab) # Transforme un tableau en image
>>> photo2.save('photo.jpg') # Permet de sauvegarder une image sous un format au choix.
```

En général une image est un quadrillage, dont l'origine se situe en haut à gauche (l'axe des ordonnées est orienté vers le bas). Il existe de nombreuses commandes pour effectuer des opérations élémentaires sur une image (ajouter un pixel, changer de couleurs....).

3 Traitement de base de donnée

Dans cette partie on ne rappelle pas l'algèbre relationnel mais on se contente de se placer du côté de l'utilisateur et... donc de manipuler le langage SQL afin d'effectuer les requêtes sur les bases de données.

Définition 3.0.1. Soit un ensemble de donnée $\mathcal{A}$, ces éléments sont appelés attributs. On associe à cet ensemble une application *dom* définie de $\mathcal{A}$ dans un ensemble $\mathcal{D}$ (les "valeur possible" de l'attribut). L'image $dom(\mathcal{A})$ est appelé le domaine de $\mathcal{A}$. Un schéma relationnel est alors un uplet de $\mathcal{A}^n$ constitué d'éléments distincts. Une relation ou une table associé à un élément $(\mathcal{A}_1, ..., \mathcal{A}_n) \in \mathcal{A}^n$ est un ensemble $dom(\mathcal{A}_1) \times .... \times dom(\mathcal{A}_n)$).

Par exemple si on s'intéresse aux données des classes préparatoires du lycée Decour : $\mathcal{A}$=(La filière, Professeur, Salle,Matière) est un ensemble de donnée et à chaque attribut, comme la filière, on associe alors un domaine par exemple : MPSI, MP,PSI,PSI*,PC*PCSI ECS1 ECS2.... pour la filière et ainsi de suite. Puis on relie ces données pour créer notre table (schéma relationnel). A la filière MPSI on associe sa salle, ses professeurs, ses matières ... remarquons qu'un même professeur peut être affecté à plusieurs classes.

3.1 Requêtes SQL simple

Le mot clé pour lancer une requête SQL est `SELECT`, suivit des commande nécéssaire le symbole `*` est une sorte de neutre.

Opération	Requête SQL	Signification
Projection	`SELECT A1,...An FROM R`	Extrait du schéma relationnel R une partie des attributs Par exemple on extrait certaine colonne de la table de donnée

Selection	`SELECT * FROM R WHERE A=a`
	
	`SELECT * FROM R WHERE (A=a and B=b) OR C!=c`
		Sélectionne les attributs
		de domaine a
		Par exemple les éléments dont les termes d'une colonne vérifie a
		Sélectionne les attributs tel que
		dom(A)=a et dom(B)=b ou dom(C)$\neq$ c

Renommage	`SELECT A1 AS B1.... An AS Bn, C1,....Cn FROM R`	L'attrinut Ai devient Bi
Union	`SELECT * FROM R1 UNION SELECT * FROM R2`	R1$\cup$R2
Intersection	`SELECT * FROM R1 INTERSECT SELECT * FROM R2`	R1$\cap$R2
Privé	`SELECT * FROM R1 EXCEPT * FROM R2`	R1$\backslash$R2

3.2 Clé

Une clé est un attribut qui permet de distinguer deux éléments d'une table, plus précisément :

Définition 3.0.2. Soit $R(S)$ un schéma relationnel de S (une table), on dit que $K \subset S$ est une clé si pour tout élément t et t' de R :

$$t(K) = t(K') \Leftrightarrow t = t'$$

Si K est un attribut de S on parle de clé primaire

Ainsi si deux enregistrement ont même clé , ils sont égaux partout.

Prépa

id_classe	Classe
931	PSI
932	PSI
921	MP
922	MP

En règle général on fabrique une clé sous forme numérique comme un identifiant, pour éviter les clés primaires accidentelles et instables (par exemple les élèves d'une classe peuvent former une clé primaire mais si la classe est modifiée ou qu'un élève suit un enseignement optionnel dans une autre classe...patatra).

3.3 Fonction d'agégation

On peut effectuer des opération simple et classique (bien connu des utilisateur de tableur) moyenne somme max... On parle de fonction d'agégation :

Opération	Requête SQL	Signification
Comptage	`SELECT COUNT(A1) FROM R`	nombre de ligne d'une colonne A1 du tableau R
maximum	`SELECT MAX(A1) FROM R`	max d'une colonne A1 du tableau R
minimum	`SELECT MIN(A1) FROM R`	min d'une colonne A1du tableau R
somme	`SELECT SUM(A1) FROM R`	somme colonne A1 du tableau R
moyenne	`SELECT AVG(A1) FROM R`	moyenne d'une colonne A1du tableau R

On peut aussi effectuer ses opérations pour créer un tableau en regroupant :

Opération	Requête SQL	Signification
Regrouper	`SELECT A1,A2,...,Ap,f(A) FROM R GROUP BY A1,A2,..,Ap`	calcul f(A) regrouper pour chaque attribut Ai
Selectionner	`SELECT A1,A2,...,Ap,f(A) FROM R GROUP BY A1,A2,..,Ap HAVING f(A)>a`	selectionne Ai vérifiant f(A)>a et affiche f(A)

Par exemple, pour obtenir la moyenne par filière : `SELECT filière, AVG(note) FROM R GROUP BY filiere`.
Attention `HAVING` ne fonctionne qu'avec les fonction d'agrégation!!

3.4 Affichage

Opération	Requête SQL	Signification
Ordonner	`SELECT * FROM R ORDER BY A1`	Ordonner les résultat par rapport à A1
Croissant	`SELECT * FROM R ORDER BY A1 ASC`	Ordonner croissant les résultat par rapport à A1
Décroissant	`SELECT * FROM R ORDER BY A1 DSC`	Ordonner décroissantles résultat par rapport à A1
Grouper	`SELECT * FROM R ORDER BY A1,A2...`	Par ordre lexicograpphique par rapport à A1,A2

3.5 Base de donnée relationnelle, opérations complexes

Le but finalement est de combiner plusieurs tables et d'appliquer les différentes requêtes sur ces combinaison (relations) :
A partir de deux tables R et R' on peut constituer une nouvelle table R×R'.

Opération	Requête SQL	Signification
Produit Cartésien	`SELECT * FROM R JOIN R'`	$R \times R'$

En général les tables R et R' ont un lien (par exeple R les étudiants avec leur classes et R' les professeur avec comme attribut leurs classes , leurs salles...), elles sont liées par leur classes, on pourra consruire une JOINTURE entre ses tables qui etablie cette relation :

Opération	Requête SQL	Signification
Produit Cartésien	`SELECT * FROM R JOIN R'`	$R \times R'$
Jointure	`SELECT * FROM R JOIN R' ON A=B`	R×R' tel qu'ils aient même attribut A=B
Jointure multiple	`SELECT * FROM R, R', R'' WEHRE A=B` AND C=D	R×R× R"' tel qu'ils aient même attribut A=B et C=D

L'attribut en commun peut aussi (et c'est assez fréquent êtres la clé primaire). Il est commode dans une table de creer une clé étrangère, qui serait un attribut reliant à une autre table.

3.6 Exemple

On se donne un schéma relationnel, décrivant les classes préparatoire d'un lycée, représenté par la table suivante :
Prépa

Nom	Prénom	Classe	Professeur
Dupont	zu	PSI	Mr Gentil
Durand	Mohamed	PSI*	Mr Méchant
Abdenour	Pierre	PC*	Mr Méchant
Wang	Julie	PSI	Mr Gentil

La requête `SELECT 'Nom' FROM 'Prépa'` retourne :

Nom
Dupontl
Durand
Abdenour
Wang

La requête `SELECT 'Nom','Prénom' FROM 'Prépa'` retourne :

Nom	Prénom
Dupont	zu
Durand	Mohamed
Abdenour	Pierret
Wang	Julie

La requête `SELECT * FROM 'Prépa' where Professeur= 'Mr Gentil'` retourne :

Nom	Prénom	Classe	Professeur
Dupont	zu	PSI	Mr Gentil
Wang	Julie	PSI	Mr Gentil

On se donne une nouvelle table :
Lycée

Nom	Prénom	Classe	Professeur
Alpha	toto	TS1	Mr Rigolo
Beta	titi	TS2	Mr Méchant
Gamma	loulou	TS1	Mr Méchant
Zorro	Pierre	TS2	Mr Rigolo

La requête `SELECT * FROM 'Prépa' UNION Sélect * FROM 'Lycée'` retourne :

Nom	Prénom	Classe	Professeur
Alpha	toto	TS1	Mr Rigolo
Beta	titi	TS2	Mr Méchant
Gamma	loulou	TS1	Mr Méchant
Zorro	Pierre	TS2	Mr Rigolo
Dupont	zu	PSI	Mr Gentil
Durand	Mohamed	PSI*	Mr Méchant
Abdenour	Pierre	PC*	Mr Méchant
Wang	Julie	PSI	Mr Gentil

Si on reprend la Table Lycée : **Lycée**

Nom	Prénom	Classe	Nom Prof
Alpha	toto	TS1	Mr Rigolo
Beta	titi	TS2	Mr Méchant
Gamma	loulou	TS1	Mr Méchant
Zorro	Pierre	TS2	Mr Rigolo

La requête `SELECT * FROM 'Prépa' JOIN 'Lycée' ON 'Nom Prof'='Professeur'` retourne :

Lycée

Nom	Prénom	Classe	Nom Prof	Nom	Prénom	Classe	Professeur
Beta	titi	TS2	Mr Méchant	Durand	Mohamed	PSI*	Mr Méchant
Gamma	loulou	TS1	Mr Méchant	Abdenour	Pierre	PC*	Mr Méchant
Beta	titi	TS2	Mr Méchant	Abdenour	Pierre	PC*	Mr Méchant
Gamma	loulou	TS1	Mr Méchant	Durand	Mohamed	PSI*	Mr Méchant

Une base de données stocke les informations liées aux polices de caractères dans 4 tables ou relations.

Famille décrit les familles de polices, avec `fid` la clé primaire entière et `fnom` leur nom.

Police décrit les polices de caractères disponibles, avec `pid` la clé primaire entière, `pnom` le nom de la police et `fid` de numéro de sa famille.

Caractere décrit les caractères, avec `code` la clé primaire entière, `car` le caractère lui-même, `cnom` le nom du caractère.

Glyphe décrit les glyphes disponibles, avec `gid` la clé primaire entière, `code` le code du caractère correspondant au glyphe, `pid` le numéro de la police à laquelle le glyphe appartient, `groman` un booléen vrai pour du roman et faux pour de l'italique et `gdesc` la description vectorielle du glyphe.

Voici un extrait du contenu de ces tables.

Famille

fid	fnom
1	Humane
2	Garalde
3	Réale
4	Didone
5	Mécane
6	Linéale
...	...

Police

pid	pnom	fid
1	Centaur	1
2	Garamond	2
3	Times New Roman	3
4	Computer Modern	4
...	...	...
21	Triangle	6
...	...	...

Caractere

code	car	cnom
65	A	lettre majuscule latine a
66	B	lettre majuscule latine b
...	...	...
97	a	lettre minuscule latine a
98	b	lettre minuscule latine b
99	c	lettre minuscule latine c
...	...	...

Glyphe

gid	code	pid	groman	gdesc
1	65	20	True	`[ [ [0, 0], [1, 2], [2, 0] ], [ [0.5, 1], [1.5, 1] ] ]`
2	65	20	False	`[ [ [0, 0], [2, 2], [2, 0] ], [ [1, 1], [2, 1] ] ]`
...	...	...	...	...
501	97	21	True	`[ [ [0, 0], [0.5, 1], [1, 0], [0, 0] ] ]`
502	98	21	True	`[ [ [0, 2], [0, 0], [1, 0.5], [0, 1] ] ]`
503	99	21	True	`[ [ [1, 1], [0, 0.5], [1, 0] ] ]`
504	100	21	True	`[ [ [1, 2], [1, 0], [0, 0.5], [1, 1] ] ]`
...	...	...	...	...

Exercice 3.1. A partir de cette base de donnée, extrait sujet Mines 2023 :

1. Proposer une requête SQL sur cette base pour compter le nombre de glyphes en roman.
2. Proposer une requête SQL sur cette base afin d'extraire la description vectorielle du caractère A dans la police nommée Helvetica en italique.
3. Proposer une requête SQL pour extraire les noms des familles qui disposent de polices et leur nombre de police classés par ordre alphabétique..