- Traitement de données en tables -

INDEXATION – RECHERCHE – TRI – FUSION DE TABLES

Plan du chapitre

I. INTRODUCTION

L'organisation tabulaires de données (sous forme de tableaux) est très répandue et très ancienne : les tables les plus anciennes sont des tables de comptes de l'Égypte ancienne.
Aujourd'hui, les données informatiques de plus en plus nombreuses et importantes doivent pouvoir être stockées, gérées, partagées et modifiables à distance. Les recherches sur des systèmes offrant ces possibilités datent du début des années 1960 avec les missions Apollo. Avec l'introduction du modèle relationnel introduit dans les années 1970 par Edgar Franck Codd (employé d'IBM qui travaillait sur un SGBD (Système de Gestion de Base de Données) utilisant le langage SQL (Structured Query Language), les tables de données, stockées dans des bases de données deviennent rapidement le principal moyen de stocker de l'information structurée.

Retour au plan

II. LES DONNÉES STRUCTURÉES AU FORMAT CSV

1. Open data.

Les données ouvertes ou open data sont des données numériques dont l'accès et l'usage sont laissés libres aux usagers. Elles peuvent être d'origine publique ou privée, produites notamment par une collectivité, un service public, un collectif citoyen ou une entreprise.

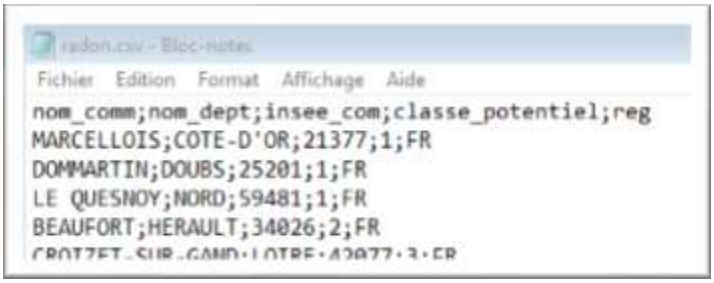
Le gouvernement met en ligne des Open data sur le site data.gouv.fr.
Télécharger sur ce site les données au format *CSV* de l'IRSN (Institut de Radioprotection et de Sureté Nucléaire) concernant les émanations de radon, gaz radioactif produit par la désintégration naturelle de l'uranium présent dans les roches. Cancérigène, il peut présenter un risque pour la santé lorsqu'il s'accumule dans les bâtiments. Le fichier classe le potentiel radon des communes du territoire métropolitain en 3 catégories, numérotée de 1 à 3, par risque croissant.

Retour au plan

2. Format CSV

Le format CSV (Comma-separated values) est très répandu sur internet : c'est un format informatique ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules.
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.

Ouvrir le fichier radon.csv avec le bloc-note de Windows :



La 1ère ligne contient **la liste des champs** (les noms des colonnes) :
"nom_comm","nom-dept", "insee_com" ,"classe_potentiel" et *"reg"* sont appelés **des descripteurs**.
"MARCELLOIS", "COTE D'OR", "25377", "1" et "FR" sont **les valeurs du descripteur**.

Vous constatez qu'ici le séparateur n'est pas la virgule mais le point-virgule (;). La virgule est effectivement un standard pour les données anglo-saxonnes, mais pas pour les données aux normes françaises (où la virgule est déjà utilisée pour séparer les chiffres décimaux). Cela peut engendrer quelques problèmes, il faut donc rester vigilants sur le type de séparateur utilisé.

Ouvrir maintenant le fichier avec le tableur "Calc" (libre Office) ou "Excel" (Windows) et vérifier que les données y sont bien rangées sous forme tabulaire :

	A	B	C	D	E
1	nom_comm	nom_dept	insee_com	asse_potenti	reg
2	MARCELLOIS	COTE-D'OR	21377	1	FR
3	DOMMARTIN	DOUBS	25201	1	FR
4	LE QUESNOY	NORD	59481	1	FR
5	BEAUFORT	HERAULT	34026	2	FR
6	CROIZET-SUR-GAND	LOIRE	42077	3	FR

Retour au plan

III. IMPORTATION DES DONNES

L'objectif est d'extraire les données puis de les enregistrer dans une liste de listes ou éventuellement dans une liste de tuples (p-uplets), c'est-à-dire dans une table.
Pour obtenir une liste de listes, on utilise le script suivant :

```
fichier= open ("radon.csv","r")

Descripteurs = fichier.readline()
listDescripteurs = Descripteurs.rstrip().split(";")

valeurs = fichier.readlines()
table =[]
for i in valeurs:
    liste = i.rstrip().split(";")
    liste[3] =int(liste[3])    # converti les valeurs du descripteur (potentiel radon), indexe 3,  de str en int
    table.append(liste)
fichier.close()
```

En utilisant l'explorateur de variables de Spyder, on peut constater que :

Nom	Type	Taille	Valeur
autresLignes	list	36093	['MARCELLOIS;COTE-D'OR;21377;1;FR ', 'DOMMARTIN;DOUBS;25201;1;FR
champs	str	1	nom_comm;nom_dept;insee_com;classe_potentiel;reg
i	str	1	Le Lamentin;Martinique;97213;1;ANT
liste	list	5	['Le Lamentin', 'Martinique', '97213', '1', 'ANT']
table	list	36093	[['MARCELLOIS', 'COTE-D'OR', '21377', '1', 'FR'], ['DOMMARTIN', 'DOUBS ...

- La variable *champs* est de type *str* et sa valeur correspond à la 1ère ligne de la table, c'est-à-dire aux descripteurs : la méthode *readline()* à la ligne 3 extrait donc la 1ère ligne de la table sous forme d'une chaine de caractères.
- La variable *autresLignes* est de type list et ses éléments dont de type str, leur valeur correspond aux autres lignes de la table : la méthode *readlines()* extrait donc les autres lignes de la table sous forme d'une liste de chaines de caractères.
- La méthode *rstrip()* supprime ici les caractères de fin de ligne *'\n'* situés à droite.
- La méthode *split(';')* converti une chaine de caractères en liste en prenant *";"* comme séparateur.

Retour au plan

IV. RECHERCHE DANS UNE TABLE

La recherche dans une table consiste à obtenir les valeurs de certains descripteurs (champs) avec éventuellement des critères les concernant.

Exercice : A partir du fichier radon.csv, extraire les informations suivantes :

- Liste des communes de France présentant un potentiel radon de 3. La liste nommée *recherche1* comportera tous les champs du fichier csv)
- Liste des communes de l'Aude présentant un potentiel radon de 2 et dont le nom commence par "A". La liste nommée *recherche2* comportera uniquement le champ correspondant au nom de la commune.
- Liste des communes de métropole à risque faible ou moyen (présentant un potentiel radon de 1 ou 2). La liste nommée *recherche3* sera une liste de tuples comprenant respectivement le nom du département, de la commune et du potentiel radon.
- Liste des départements de France dont au moins 1 des communes présente un potentiel radon de 2 ou 3. La liste nommée *recheche4* sera une liste de tuples comprenant le nom du département et le potentiel radon.

Les listes générées pourront être explorées avec l'explorateur de variables.

Remarque : la méthode *index("nomDescripteur")* retourne l'indice du descripteur dans la table, par exemple :

```
14 indexCom = listDescripteurs.index("nom_comm")
```

Le problème des doublons

Parmi les listes précédentes, y-en-a-t-il qui présentent des doublons. Si oui, lesquelles ?

Écrire une fonction nommée sa*nsDoublon(list*), qui prendra en argument une liste et retournera une nouvelle liste sans doublon. Appliquer cette fonction aux listes concernées par les problèmes de doublons.

Retour au plan

V. TRI D'UNE TABLE SUIVANT UNE COLONNE

Le tri dans une table consiste à modifier l'ordre des données pour qu'elles soient présentées dans un ordre croissant ou décroissant selon le choix d'un ou plusieurs critères.

Exercice : Nous avons établi précédemment une liste des communes à risque faible ou moyen nommée recherche3 : liste de tuples comprenant le nom du département, de la commune et le potentiel radon
Trier la liste recherche3 **par ordre alphabétique croissant** des communes.

Pour effectuer un tri il est possible de programmer nous même une fonction de tri (voir chapitre précédent).

Il est aussi possible d'utiliser la fonction *sorted()* de python qui retourne une liste triée par ordre croissant (paramètre *reverse=False*) ou par ordre décroissant (paramètre *reverse=True*) :

```
66 Recheche3Tri = sorted(recherche3, reverse=False)
```

Sur quelle colonne s'est effectuée le tri ? Est-il pertinent par rapport à notre problématique ?
Pour résoudre le problème, une possibilité serait de placer en premier le champ *commune.* Une autre possibilité est d'utiliser le paramètre *key* :

```
66 Recheche3Tri = sorted(recherche3,key=lambda colonne : colonne[1], reverse=False)
```

Ici le tri s'effectue sur la colonne indexée 1, c'est-à-dire la 2ème colonne (qui correspond bien aux communes)

Retour au plan

VI. FUSION DE TABLES

On peut distinguer 2 situations :

- **La concaténation** : on ajoute les enregistrements (c'est-à-dire les lignes de la table) de la table2 dans la table1
- **La jointure** : On ajoute des champs (c'est-à-dire les colonnes de la table) de la table 2 dans la table 1.

Remarque : Il faut bien évidement faire attention à ce que les données soient compatibles entres elles.

1. Concaténation

Si les 2 tables possèdent exactement les mêmes descripteurs, dans le même ordre, et que les valeurs sont de même type, la fusion est facile car il suffit de concaténer les 2 tables.

Il peut être nécessaire de supprimer ensuite d'éventuels doublons (on peut utiliser notre fonction *sansDoublons()* écrite au paragraphe IV). On peut aussi rajouter les éléments de la table 2 à la table 1, un par un, en vérifiant systématiquement qu'ils ne sont pas déjà présents dans la table 1.

S'il manque un champ à une table, il suffit de le créer dans la table en insérant les valeurs *None* dans le champ.
On peut également décider de ne pas prendre les champs qui n'existent pas dans les 2 tables (s'ils ne sont pas utiles). Dans ce cas il faut les supprimer avant de concaténer.

Exercice :

- Établir la liste des communes de France à risque moyen (présentant un potentiel radon de 2). La liste nommée *recherche5* comportera tous les champs du fichier csv.
- Fusionner cette liste avec la liste recherche1 des communes de France à fort risque (présentant un potentiel radon de 3) afin de générer une liste des communes de France à risque (moyen ou fort) nommée *recherche6.*

Retour au plan

2. Jointure

Exercice :

- Établir une liste de tuples nommée *recherche7* de toutes les communes de Métropole comprenant respectivement le nom de la commune, le nom du département et le numéro d'insee.

- Établir une jointure avec la liste *Recherche3* de toutes les communes de métropole à risque faible ou moyen afin d'obtenir une liste de tuples nommée *recherche8* comprenant le nom du département, de la commune, le numéro d'insee et le potentiel radon. Cette liste ne sera plus sous la forme d'une liste de tuples mais d'une liste de listes.

Attention : Vérifier bien que votre liste *recherche8* ne comporte pas plus d'éléments que la liste *recherche3* ! Si c'est le cas, est-ce possible ? Quelle est la cause de ce problème ? y remédier.

Retour au plan

VII. EXPORTATION DES DONNÉES DE LA TABLE TRAITÉE AU FORMAT CSV

Afin d'exporter le fichier au format csv, il faut convertir toutes les valeurs qui ne sont pas déjà au format str.
Écrire le script python permettant cette conversion (2 lignes de code devraient suffire).

Le code suivant permet ensuite l'exportation des données de la table *recherche8* dans le fichier csv nommé *fichierE.csv* :

```
104 descripteurs = ["com","dep","insee","potRadon"]  #listes de descripteurs
105 fichierE =open("fichierE.csv","w")               # ouverture du fichier en ecriture
106 listeDescripteurs = ";".join(descripteurs)+"\n"  # création de la chaine de caractères des descripteurs séparés par ;
107 fichierE.write(listeDescripteurs)                # ecriture dans le fichier
108 for i in recherche8 :
109     valeurs = ";".join(i) + "\n"                 #création de la chaine de caractère d'une ligne de valeurs séparées par ;
110     fichierE.write(valeurs)                      # écriture dans le fichier
111 fichierE.close()                                 # fermeture du fichier
```

Retour au plan