Structure de données

Avant de travailler sur le traitement de structures de données, on va faire une petite présentation et quelques manipulations très simples.

Le traitement des données est devenu un enjeu essentiel. Aujourd'hui, le développement technologique permet de recueillir énormément de données (le fameux big data). Encore faut-il savoir les exploiter.

Un format souvent rencontré sur internet est le CSV (comma separated values) qui présente les données sous forme de table, les données étant séparées par une virgule. Chaque ligne du fichier correspond à une donnée:

Voici un exemple :

```
Classe, effectif, spécialité
101,30,Mathématiques
102,28,Physique
103,32, NSI
104,35,Cinema
105,27,SES
```

La première ligne de ce fichier correspond aux descripteurs : On a les classes (leur numéro), l'effectif correspondant et la spécialité.

Les autres lignes correspondent aux différents éléments de ce fichier : On a ici cinq classes qui sont renseignées.

Ouvrez ce fichier avec un tableur. Attention, le point virgule est souvent utilisé comme séparateur en France.

Une fois que les données ont été récupérées, on peut commencer à effectuer leur traitement. Ici on voudrait que les classes soient rangées par effectif croissant et non par numéro .

Méthode 1: A l'aide du tableur, classer les classes selon le critère retenu.

Méthode 2: On va utiliser le langage python pour travailler sur les données. Si ici on a un fichier minimaliste, on va être amené à traiter des données bien plus conséquentes.

Il va nous falloir :

- Importer les données
- Traiter les données
- Exporter les données traitées

1. **Importation**

```
import csv

def charger_fichier(nom_fichier):
    """
    Permet de charger un fichier CSV
    paramètres: nom_fichier une chaine de caractères contenant le nom du fichier
    resultat: la liste
    """
    classe=[]
    with open(nom_fichier, 'r', newline='', encoding='utf-8' ) as csvfile:

        votre_liste=csv.reader(csvfile,delimiter=";")


        for enreg in votre_liste:
            classe.append(enreg)

    return classe
```

On obtient ici une liste . Si l'on veut travailler avec un dictionnaire , on utilise **DictReader.** On obtient alors des données de type OrderedDict (dictionnaire ordonnée) . Il suffit au moment de l'ajout avec la commande append de spécifier dict(enreg) pour obtenir une liste de dictionnaires.

A noter, lors de l'importation, ***les clés du dictionnaire correspondront aux descripteurs de la première ligne du fichie csv.***

2. Traitement des données

Utiliser les méthodes utilisées pour faire des opérations sur les dictionnaires pour obtenir les classes triées par effectif croissant . (Utiliser la méthode sort ou sorted)

3. Exportation des données

```python
def export_fichier_csv(nom_fichier, liste_enregistrements, separateur=";"):
    """
    Enregistre dans un fichier au format CSV les enregistrements de la liste
    fournie en paramètre.

    Le fichier CSV sera enregistré dans le même dossier que ce module et
    utilisera l'encodage UTF-8.

    Un enregistrement doit être constitué d'un dictionnaire. Tous les
    dictionnaires doivent avoir les mêmes clés, et ces clés seront les entêtes
    des colonnes du fichier CSV.

    Par défaut, le séparateur utilisé est le point-virgule ';'.

    Paramètres
    ----------
    nom_fichier : str
        Une chaîne de caractères représentant le nom du fichier CSV dans lequel
        exporter.
    liste_enregistrement : list[dict]
        Une liste de dictionnaires, chaque dictionnaire représentant un
        enregistrement.
    separateur : str, optionnel
        Une chaine de caractères représentant le séparateur à utiliser entre
        chaque champ d'un enregistrement.

    Valeur de retour
    ----------------
    Aucune

    """

    with open(nom_fichier, 'w', newline='', encoding="utf-8") as csvfile:
        writer = csv.DictWriter(csvfile,
                                fieldnames=liste_enregistrements[0].keys(),
                                delimiter=separateur)

        # Ecriture de la première ligne du fichier CSV avec les noms des champs
        writer.writeheader()

        # Export de tous les enregistrements de la liste
        for enregistrement in liste_enregistrements:
            writer.writerow(enregistrement)
```

Autre version

```python
def sauver_produit(nom_fic, scores):
    """
    Permet de sauvegarder une liste de reference dans un fichier CSV
    paramètres:
       nom_fic une chaine de caractères contenant le nom du fichier
       scores la liste des scores à sauvegarder
    resultat aucun
    """

    # ouverture du fichier CSV
    with open(nom_fic, # nom du fichier
              'w', # ouverture en écriture
              newline='', # évite les problèmes de codage du retour à la ligne
              encoding='utf-8' # permet de forcer la lecture en utf-8
    ) as csvfile: # cvsfile est le fichier que l'on vient d'ouvrir
        # création du lecteur csv indiquant le caractère séparateur
        score_writer=csv.writer(csvfile,delimiter=";")
        # permet d'écrire la ligne d'entête
        score_writer.writerow(["classe","effectif","spécialité"])
        for enreg in scores: # boucle de parcours de la liste des scores
            # enreg est un dictionnaire correspondant à un score champ de l'enregistrement
            score_writer.writerow([enreg["classe"],enreg["effectif"], enreg["spécialité"]])
```