Numérique et sciences informatiques
Classe de première

E Traitement de données en tables
2. Recherche dans un ensemble de données tabulées

Dans cette partie nous supposons que nous disposons d'une variable du type « tableau de dictionnaires » qui représente un ensemble de données tabulées. On peut, par exemple, imaginer que ce tableau a été obtenu par la lecture d'un fichier CSV.

• Rappelons le format de cette variabe :
  * Chacun des dictionnaires de ce tableau contient exactement les mêmes clés qui correspondent aux noms des différents attributs ;
  * Les valeurs associées à une clé donnée sont toutes de même type.

• Le but de cette partie est de décrire les programmes qu'il convient d'écrire pour extraire différentes informations de cette variable, qui peut contenir un très grand nombre de données.

Pour illustrer cela, reprenons la situation décrite dans la partie III du chapitre E1 consacré à la lecture d'un fichier de données tabulées. Par lecture d'un fichier `commandes.csv`, nous avons créé une variable `'commande'` qui est un tableau de dictionnaires. Chaque élément de ce tableau est un dictionnaire dont les clés sont `"Référence"`, `"Désignation"`, `"Prix"`, `"Quantité"` et `"Disponible"`. Les valeurs associés à ces différentes clés sont du type suivant :
  * de type **`str`** pour la clé `"Désignation"`,
  * de type **`int`** pour les clés `"Référence"` et `"Quantité"` ;
  * de type **`float`** pour la clé `"Prix"` ;
  * de type **`bool`** pour la clé `"Disponible"`.

I Recherche et agrégation

I.1 Tester la présence d'un objet vérifiant un certain critère

La première chose que nous pouvons faire c'est déterminer la présence ou l'absence dans le tableau d'objets vérifiant un certain critère.
Par exemple, on pourrait souhaiter savoir si notre variable `'commande'` comporte une ligne pour une certaine référence ou bien si elle comporte des demandes pour un produit indisponible. Il s'agit alors de créer une fonction qui applique une certaine condition à cette variable et renvoit un booléen : **`True`** ou **`False`**.

Un exemple

Voici une telle fonction qui détermine la présence d'une certaine référence dans une variable au format décrit plus haut. :

```
def contientRef(tab, ref) :
    for elem in tab :
        if elem["Référence"] == ref :
            return True
    return False
```

I.2 Nombre d'occurrences des objets vérifiant un certain critère

Au lieu de renvoyer un booléen, on peut aussi souhaiter créer une fonction indiquant le nombre de lignes vérifiant un certain critère. Ceci permettrait également de répondre à la question précédente puisqu'une telle fonction renverrait 0 si aucune ligne ne vérifie le critère.

Un exemple

Voici une fonction pour déterminer combien de dictionnaires de la variable `'commande'` correspond à un prix compris entre une valeur minimalet et une valeur maximale :

```
def nbPrix(tab, pMin, pMax) :
    nb = 0
    for elem in tab :
        if pMin <= elem["Prix"] <= pMax :
            nb += 1
    return nb
```

II Sélection et projection

II.1 Sélection simple

Une opération très courante va consister à former, à partir du tableau de départ, un nouveau tableau « plus petit » qui contiendrait seulement les lignes qui vérifient un certain critère. Dans le vocabulaire du traitement des bases de données, un telle opération s'appelle **une sélection**.

• Pour effectuer une sélection à partir d'un tableau en Python, on peut utiliser deux technique :

* soit la constitution progressive d'un nouveau tableau à l'aide d'une boucle `for` et de la méthode `append` ;
* soit l'utilisation de la « construction par compréhensions ».

Un exemple

Illustrons ces deux méthodes en écrivant une fonction permettant de sélectionner les lignes d'une commande correspondant à un produit disponible :

• Première méthode :

```
def selectDispo1(tab) :
    nouvTab = []
    for elem in tab :
        if elem["Disponible"] :
            nouvTab.append(elem)
    return nouvTab
```

• Deuxième méthode :

```
def selectDispo2(tab) :
    return [elem for elem in tab if elem["Disponible"]]
```

On voit que la deuxième méthode par compréhension est bien plus compacte et élégante.

II.2 Projection simple

Une autre opération courante consiste à garder non pas seulement certaines lignes mais seulement *certaines colonnes du tableau*, c'est-à-dire certains attributs. On obtiendrait dans ce cas le même nombre de lignes que dans le tableau initial mais on ne retiendrait que certains des attributs associés à chaque objet. Dans le vocabulaire du traitement des bases de données, une telle opération s'appelle **une projection**.

Un exemple

Voici par exemple une fonction pour créer une sous-table ne contenant que les attributs "Référence" et "Quantité" en utilisant la méthode par compréhension :

```
def projQuantite(tab) :
    return[{"Référence":elem["Référence"],
"Quantité":elem["Quantité"]} for elem in tab]
```

L'application de cette fonction à la variable 'commande' donnerait le résultat suivant :

```
{'Référence': 18635, 'Quantité': 1}
{'Référence': 17986, 'Quantité': 3}
{'Référence': 20165, 'Quantité': 5}
{'Référence': 31254, 'Quantité': 1}
```

II.3 Sélection et projection combinées

On peut aisément utiliser la méthode par compréhension pour *combiner* une sélection et une projection, c'est-à-dire obtenir un sous-tableau formé de seulement certains attributs (projection) et seulement certaines objets choisis selon un certain critère (sélection).

Un exemple

Pour obtenir seulement la référence et la quantité et seulement pour des produits disponibles, on peut écrire la fonction suivante :

```
def selecProj(tab) :
     return [{"Référence":elem["Référence"],\
     "Quantité":elem["Quantité"]} \
     for elem in tab if elem["Disponible"]]
```

On obtiendrait alors le résultat suivant :

```
{'Référence': 18635, 'Quantité': 1}
{'Référence': 20165, 'Quantité': 5}
```