Aix*Marseille université
Initiative d'excellence
Année 2018-19

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Bloc 1 : TP 1

Imports CSV et requêtes sur des tables

1 Introduction

Dans ce premier TP, nous allons revoir quelques basiques Python que sont les listes et les dictionnaires. Nous allons nous exercer sur des données importées depuis un fichier CSV contenant des relevés MétéoFrance, disponibles en ligne sur cette page. Plus précisément, nous allons nous intéresser aux relevés météo du mois de février 2019, dont le fichier est disponible sur le cours Ametice.

2 Import d'un fichier CSV en Python

La première étape, avant de commencer à travailler sur les données, est d'importer les données depuis le fichier CSV. Pour cela, nous allons utiliser une librairie Python, qui fait cela très bien pour nous (nous reverrons en juin plus en détail comment lire depuis un fichier ou écrire dans un fichier). Il s'agit de la librairie `csv` qui possèdent plusieurs méthodes intéressantes, dont la méthode `csv.DictReader` qu'on peut utiliser ainsi :

```
import csv
file = open('path_to_file/file.csv','r')
reader = csv.DictReader(file, delimiter=';')
```

Après l'exécution de ce code, la structure de données `reader` contient chaque enregistrement (c'est-à-dire chaque ligne) du fichier CSV, sous la forme d'un dictionnaire associant à chaque nom de champ sa valeur sous la forme d'une chaîne de caractères.

Pour ce TP, nous n'avons pas besoin de conserver l'ensemble des champs de chaque enregistrement. Seuls les champs suivants nous intéressent :

- `numer_sta` : l'identifiant de la station météo
- `ff` : la vitesse du vent moyen les dix dernières minutes, en m/s
- `t` : la température en Kelvin
- `u` : l'humidité en pourcentage
- `rr1` : les précipitations dans la dernière heure, en mm

On peut donc récupérer uniquement ces champs en parcourant la structure `reader` chargée précédemment, en traduisant dans le type correct (flottant ou entier) les champs numériques (qui sont sinon tous traités comme des chaînes de caractères) :

```
liste_releves = []
for row in reader:
    liste_releves.append({'numer_sta' : row['numer_sta'],
                          'ff' : float(row['ff']),
                          't' : float(row['t']),
                          'u' : int(row['u']),
                          'rr1' : float(row['rr1'])})
file.close()
```

Question 1. Modifier ce code afin de renvoyer la vitesse du vent en km/h (on appellera `vitesse_vent` le nouveau champ), et la température en degrés Celsius (on appellera `temperature` le nouveau champ), sachant qu'une température de x Kelvin équivaut à $x - 273.15$ degrés Celsius. On renommera également `id` le champ `numer_sta`, `humidite` le champ `u` et `precipitations` le champ `rr1`. Attention, certains enregistrements sont

incomplets, certains de leurs champs étant alors égaux à `mq` : on ne considèrera pas les enregistrements dont l'un des champs `ff`, `t`, `u` ou `rr1` est égal à `mq`. Vous devriez normalement récupérer 12 860 enregistrements complets.

3 Requêtes

3.1 Statistiques

Question 2. Écrire une fonction renvoyant la température minimale relevée.

Question 3. Écrire une fonction renvoyant l'identifiant de la station météo ayant relevé la vitesse maximale de vent.

Question 4. Écrire une fonction renvoyant le taux d'humidité moyen sur l'ensemble des relevés.

Question 5. Écrire une fonction renvoyant le niveau de précipitation moyen relevés par les stations ayant un identifiant compris entre `60000` et `69999` ?

3.2 Recherche d'une station

Question 6. Écrire une fonction filtrant la liste des relevés pour ne renvoyer que les relevés d'une station d'identifiant donné en argument.

Utilisons les fonctions de tri de la librairie standard de Python afin de trier la liste des enregistrements par numéro d'identification croissant.

```
def id(x): return x['id']
liste_releves.sort(key=id)
```

Question 7. Améliorer la fonction de filtre précédente pour qu'elle s'arrête plus rapidement.

Question 8. Modifier finalement la fonction pour qu'elle commence par trouver en temps logarithmique un enregistrement de la station d'identifiant donné en argument, puis qu'elle construise dans un second temps la liste des relevés de cette station.

3.3 Fusion de tables

Comparons les relevés de février 2019 avec ceux de février 2009. Un fichier similaire pour février 2009 est disponible également sur le cours Ametice.

Plus précisément, on cherche à comparer les relevés de température des stations en activité lors de ces deux mois (les identifiants des stations météo n'ont pas changé depuis, mais certains stations ont disparu et d'autres sont apparues), pour chaque relevé. On va donc se servir également d'un champ supplémentaire des fichiers :

— `date` : la date du relevé sous le format `AAAAMMJJhhmmss`

Question 9. Écrire une fonction permettant de fusionner les deux tables extraites des fichiers de 2009 et de 2019, pour conserver dans chaque enregistrement l'identifiant de la station, la date du relevé de laquelle on supprime l'année, et deux champs `t2009` et `t2019` donnant le relevé de température (différent de `mq`) en février 2009 et février 2019, respectivement.

Question 10. Utiliser la table fusionnée pour en déduire lequel des deux mois a été le plus chaud.