SAE15 : Traiter des données
BUT R&T : 1A - S1
TP 2 : Manipuler des données en Python

IUT de Lannion

Ce TP est à faire seul et sera évalué par des points d'avancement. Vous aurez également à déposé votre code commenté sur Moodle en fin de séance. Vous allez concevoir un script Python qui exploitera les données générées lors du TP précédent.

Pour rappel, à l'aide des scripts bash du TP1 et de l'outil *CRON*, vous avez mis en place une stratégie pour surveiller l'évolution de la taille des fichiers d'un répertoire. Ces observations sont centralisées dans un fichier au format *csv* contenant pour chaque fichier observé les données suivantes :

- nomDuFichier,tailleEnKO,timestamp

Si vous n'avez pas su finir le TP1, utilisez les données (*donneesTP2_SAE15.csv*) disponibles sur l'ENT.

1 Chargement des données en Python

1.1 Des données à un dictionnaire

L'objectif de ce premier exercice est de charger en mémoire depuis un script Python les observations réalisées sur la taille des fichiers. Vous trouverez ci-dessous un rappel du code permettant de parcourir en Python des données stockées dans un fichier csv :

```
>>> import csv
>>> with open('data.csv', newline='') as csvfile:
...     datareader = csv.reader(csvfile, delimiter=' ', quotechar='|')
...     for row in datareader:
...         print(', '.join(row))
a1,a2,...,an
b1,b2,...,bn
...,...,...,...
v1,v2,...,vn
```

Votre fichier d'observation décrit l'évolution de la taille de différents fichiers, vous allez tout d'abord regrouper toutes les données qui concernent le même fichier.

Vous allez ainsi créer une fonction de prototype suivant :

```
from typing import Dict
import numpy as np

def csvToDict(f : str) -> Dict[str,np.ndarray]:
      """
      Fonction chargeant les données stockées dans le fichier f et retournant un dictionnaire
      où les clés sont les différents noms de fichiers décrits et les valeurs (taille en ko, timestamp)
      sont stockées dans un ndarray
      """
```

Pour construire l'élaboration de cette fonction, vous aurez besoin des éléments techniques suivants :

- initialiser le tableau *np.ndarray* lorsque vous rencontrer pour la première fois un nom de fichier (*f* avec ses valeurs (*taille1*, *temps1*), ret[f] = np.array([[taille1, temps1]])
  Attention à bien stocker des valeurs numériques dans votre tableau.
- ajouter une nouvelle entrée dans le tableau des valeurs pour un fichier déjà connu (càd. qui possède déjà une clé dans le dictionnaire à construire).
  ret[f] = np.append(ret[f], [[taille, temps]],axis=0)

Testez la bonne exécution de votre fonction en affichant le dictionnaire construit.

2 Statistiques encore

2.1 Observez l'évolution de la taille des fichiers

Les données manipulées décrivent l'évolution de la taille de fichiers selon un intervalle de temps constant (une minute logiquement : 60 secondes). Vous allez construire deux fonctions qui prennent toutes les deux en paramètre un tableau (i.e. *np.ndarray*) et qui retournent respectivement les informations suivantes :

1. l'augmentation moyenne de la taille du fichier entre deux captations,
2. et la plus grande augmentation de taille lors de la période d'évaluation retournée sous forme d'un couple (augmentation, timestamp).

Vous allez maintenant construire une nouvelle fonction qui prend cette fois-ci deux paramètres : un tableau (i.e. *np.ndarray*) de données et une taille limite exprimée en kilo-octets. La fonction retourne la date (i.e. le timestamp) à partir duquel le fichier a atteint la taille limite.

Faites un test depuis votre programme principal en affichant pour chaque fichier observé la date à laquelle il a dépassé les 2MO. Au lieu d'afficher un timestamp au format *EPOCH*, vous utiliserez l'instruction ci-dessous pour obtenir une date plus lisible :

```
from datetime import datetime
ts = int('1636975020')
print(datetime.utcfromtimestamp(ts).strftime('%d-%m-%Y %H:%M:%S'))
```

2.2 Affichage final

Complétez le programme principal de votre script pour afficher pour chaque fichier les informations suivantes :

- le nom du fichier,
- l'augmentation moyenne de sa taille à chaque captation et la date de la plus grande augmentation (avec l'augmentation elle-même),
- la date à laquelle le fichier a dépassé les 2 MO.