Compte rendu SAE 15 TP 2

HARTMANN Matthias, 3A

Sommaire

Partie 1 : Chargement des données en Python

Explication de la consigne :

Dans cette partie nous avions à charger en mémoire depuis un script Python lesobservations réalisées sur la taille des fichier. Pour cela nous récuppérons les données sauvegardées sous la forme d’un fichier csv à l’aide de la fonction python « open » et du module csv

La fonction Python:

```
def csvToDict(path : str) -> Dict[str, np.ndarray]:
      """!
```

```
      @brief Cette fonction génère un dictionnaire qui contient les données
      d'évolutions des fichiers.

      Paramètre(s) :
      @param path : str => le chemin (absolu ou relatif) du fichier qui sera
      utilisé pour obtenir les données
      Retour de la fonction :
      @return Dict[str, np.ndarray] Un dictionnaire ayant pour clé le nom de
      chaque fichier et pour valeur les données d'évolution

      """
      data : Dict[str, np.ndarray] = {}
      with open(path) as file:
      reader = csv.reader(file)
      for row in reader:
            name : str = row[0]
            taille : int = int(row[1])
            temps : int = int(row[2])
            if name in data.keys():
                  data[name] = np.append(data[name], [[taille, temps]], axis=0)
            else :
                  data[name] = np.array([[taille, temps]])
                  return data
```

Exemple de bon fonctionnement :

Affichage suite au lancement du fichier script depuis un terminal :

```
{'/home/toto/Tmp/f1': array([[        40, 1636974960],
       [        52, 1636975020],
       [        56, 1636975080],
       [        64, 1636975140],
       [       132, 1636975200],
       [       132, 1636975260],
       [       132, 1636975320],
       [       196, 1636975380], . . .
```

Partie 2 : Statistiques Encore

Les données manipulées décrivent l'évolution de la taille des fichiers selon un intervalle de temps constant (une minute logiquement). Nous devions construire deux fonctions qui retournes l'augmentation moyenne de la taille du fichier entre deux captations et la plus grande augmentation de la taille lors de la période d'évaluation.

Les fonctions Python :

```
def augmentationMoyenne(data : np.ndarray) -> int:
      """!
```

```
      @brief Cette fonction retourne l'augmentation moyenne de la taille du
      fichier entre deux captations

      Paramètre(s) :
      @param data : np.ndarray => tableau contenant les données d'évolutions
      (taille et temps)
      Retour de la fonction :
      @return int la moyenne
      """
      augment = data[-1,0] - data[0,0]
      return augment / data.shape[0]

def plusGrandeAugmentation(data : np.ndarray) -> Tuple[int, int]:
    """!
    @brief Cette fonction retourne la plus grande augmentation de la taille du
fichier lors de la période d'évaluation

    Paramètre(s) :
        @param data : np.ndarray => tableau contenant les données d'évolutions
(taille et temps)
    Retour de la fonction :
        @return Tuple[int, int] Le couple désignant la plus grande augmentation
et le timestamp

    """
    augmentation : int = 0
    resultat : Tuple[int,int]
    for i in range(1, data.shape[0]):
        augment = data[i,0] - data[i-1,0]
        time : int = data[i, 1]
        if augment > augmentation :
            augmentation = augment
            resultat = (augmentation, time)
    return resultat

def tailleLimite(data : Dict[str, np.ndarray], limite: int) -> int:
    """!
    @brief Cette fonction retourne le timestamp auquel le fichier à dépasser la
taille limite passée en paramètre

    Paramètre(s) :
        @param data : Dict[str, np.ndarray] => tableau contenant les données
d'évolutions (taille et temps)
        @param limite : int => La taille limite en Kilo Octet
    Retour de la fonction :
        @return int le timestamp ou None si la limite n’a pas été dépassée

    """
    time : int = 0
    i : int = 0
    while i < data.shape[0] and data[i, 0] < limite:
        time = data[i,1]
        i += 1
    return None if i == data.shape[0] and data[i-1,0] <limite else time
```

Exemple de bon fonctionnement :

Affichage suite au lancement de la fonction :

```
augmentationMoyenne(donnes["/home/toto/Tmp/f3"])
```

depuis un terminal :

```
15.319148936170214
```

Affichage suite au lancement de la fonction :

```
plusGrandeAugmentation(donnes["/home/toto/Tmp/f3"])
```

depuis un terminal :

```
(64, 1636975140)
```

Affichage suite au lancement de la fonction :

```
if __name__ == "__main__":
        donnes : Dict[str, np.ndarray] = csvToDict("TP_2/donneesTP2_SAE15.csv")
        for i in donnes:
                print(tailleLimite(donnes[i], 2000))
```

depuis le terminal :

```
None
1636976400
None
```

Partie 3 :Affichage Final

Explication de la consigne :

Dans cette dernière partie nous devions produire un affichage « user-friendly » avec l'ensembles des précédentes informations.

Le programme principal:

```
def formatTime(t:int) -> str:
      """!
    @brief Cette fonction renvoie une date EPOCH formatée

    Paramètre(s) :
        @param t : int => le timestamp
    Retour de la fonction :
        @return str la date formatée

    """
    return datetime.utcfromtimestamp(t).strftime('%d-%m-%Y %H:%M:%S')

if __name__ == "__main__":
    donnes : Dict[str, np.ndarray] = csvToDict("TP_2/donneesTP2_SAE15.csv")
    for i in donnes:
        print(f'Le fichier {i} a : \n\t- pour valeur d\'augmentation moyenne :
{augmentationMoyenne(donnes[i])}', end=" ")
        grande : Tuple[int, int] = plusGrandeAugmentation(donnes[i])
        print(f'\n\t- la plus grande augmentation s\'est passée le
{formatTime(grande[1])} ({grande[0]}Ko)', end=" ")
        limite : int or None = tailleLimite(donnes[i], 2000)
```

```
        if limite == None :
            print('Le fichier n\'a jamais dépassé les 2Mo')
        else :
            print(f'le fichier a dépassé les 2Mo le : {formatTime(limite)}')
```

Exemple de bon fonctionnement :

```
Le fichier /home/toto/Tmp/f1 a :
        - pour valeur d'augmentation moyenne : 15.574468085106384
        - la plus grande augmentation s'est passée le 15-11-2021 11:20:00 (68Ko)
Le fichier n'a jamais dépassé les 2Mo
Le fichier /home/toto/Tmp/f2 a :
        - pour valeur d'augmentation moyenne : 28.595744680851062
        - la plus grande augmentation s'est passée le 15-11-2021 11:41:01
(1024Ko) le fichier a dépassé les 2Mo le : 15-11-2021 11:40:00
Le fichier /home/toto/Tmp/f3 a :
        - pour valeur d'augmentation moyenne : 15.319148936170214
        - la plus grande augmentation s'est passée le 15-11-2021 11:19:00 (64Ko)
Le fichier n'a jamais dépassé les 2Mo
```