TD9 : Combinaison de sources

Dans ce TD, vous mobiliserez des données issues de sources diverses (API, fichiers CSV) afin de répondre à LA question du moment : **Qui a gagné la course "Rennes 2 Express" ?**

Préambule

— Dans votre dossier PythonOpenData créez un sous-dossier TD9.

— Lancer l'éditeur Visual Studio Code.

— Dans Visual Studio Code, ouvrir le dossier PythonOpenData/TD9 et créez un fichier td9.py dans lequel vous écrirez votre code.

Pour ce TD, il est conseillé d'importer le module `pprint` qui permet d'afficher de manière claire les dictionnaires.

Rennes 2 Express

Lors de l'émission télévisée Rennes 2 Express, 8 équipes s'affrontent pour tenter de remporter un grand prix ! Les concurrents s'élancent de Marseille le lundi 13 novembre, et ils ont 4 jours pour rallier l'Université Rennes 2, le plus rapidement possible.

Chaque journée de jeu se déroule de la manière suivante :

— Une étape le matin, lors de laquelle les concurrents doivent rallier le plus rapidement possible la ville suivante.

— Une étape l'après-midi, lors de laquelle les concurrents doivent rallier la ville suivante.

Au début et à la fin de chaque étape, une borne est disposée sur le parcours, et les candidats doivent badger pour valider leur temps de parcours de l'étape. À vous de déterminer les grands gagnants de cette course haletante !

La société de production de Rennes 2 Express vous fournit les informations suivantes :

— les informations sur l'édition 2024 de l'émission avec notamment la liste de toutes les étapes (disponible via une API à l'URL `http://my-json-server.typicode.com/alemaitr/python_opendata_l2/rennes2express`) ;

— la position des bornes de début et de fin de parcours, avec leur position GPS (fournie sous la forme d'un fichier `bornes.csv` disponible sur CURSUS) ;

— la liste de tous les enregistrements des équipes sur les bornes, avec l'horaire précis de validation (fournie sous la forme d'un fichier `validation_badge.csv` disponible sur CURSUS).

Téléchargez les fichiers CSV depuis CURSUS et enregistrez-les dans votre dossier `PythonOpenData/TD9`.

Exercice 1 : Les étapes du parcours

Pour commencer, vous allez récupérer les informations relatives aux bornes de début et de fin d'étape du parcours de l'édition 2024 de Rennes 2 Express.

Q.1.1. Visualisez (à l'aide d'un navigateur web) les données fournies par l'API citée plus haut. À quelle clé va-t-il falloir accéder pour récupérer les informations qui vous intéressent ?

Q.1.2. Implémentez une fonction `get_etapes` qui prend en entrée une URL et retourne un dictionnaire ayant :
- pour clés des identifiants d'étapes (aussi appelés `code_etape`) ;
- pour valeur un dictionnaire contenant deux paires clé-valeur : à la clé `borne_depart` sera associée l'identifiant de la borne de départ et à la clé `borne_arrivee` sera associée l'identifiant de la borne d'arrivée (pour l'étape considérée).

Q.1.3. Testez votre fonction. Vous devriez obtenir une sortie du type :

```
{'Etape 1': {'borne_depart': 'B01', 'borne_arrivee': 'B02'},
 'Etape 2': {'borne_depart': 'B03', 'borne_arrivee': 'B04'},
 'Etape 3': {'borne_depart': 'B05', 'borne_arrivee': 'B06'},
 'Etape 4': {'borne_depart': 'B07', 'borne_arrivee': 'B08'},
 'Etape 5': {'borne_depart': 'B09', 'borne_arrivee': 'B10'},
 'Etape 6': {'borne_depart': 'B11', 'borne_arrivee': 'B12'},
 'Etape 7': {'borne_depart': 'B13', 'borne_arrivee': 'B14'},
 'Etape 8': {'borne_depart': 'B15', 'borne_arrivee': 'B16'}}
```

Exercice 2 : Les bornages des équipes

Dans cette deuxième partie, vous allez récupérer les informations de bornage des équipes (identifiants des équipes, identifiants des bornes et dates des bornages) à partir du fichier `validation_badge.csv`.

Q.2.1. Quel est le délimiteur utilisé dans le fichier `validation_badge.csv` ? Quels sont les noms des attributs ? Quelle fonction du module `csv` comptez-vous utiliser pour lire le contenu de ce fichier ?

Q.2.2. Implémentez une fonction `validations_equipes` qui prend en entrée un nom de fichier CSV et retourne un dictionnaire ayant :
- pour clés des identifiants d'équipes ;
- pour valeur un dictionnaire dont les clés sont des identifiants de bornes et pour valeur la date du bornage.

Q.2.3. Testez votre fonction avec le fichier `validation_badge.csv`. Vous devriez obtenir une sortie du type :

```
{'E1': {'B01': '13/11/2024 08:00',
        'B02': '13/11/2024 10:35',
        'B03': '13/11/2024 14:00',
        'B04': '13/11/2024 17:12',
        'B05': '14/11/2024 08:00',
        'B06': '14/11/2024 09:02',
        'B07': '14/11/2024 14:00',
...
```

Q.2.4. Améliorez votre fonction `validations_equipes` pour que les dates soient converties au moment de leur insertion dans le dictionnaire au format `datetime`. Validez votre nouvelle version sur le fichier `validation_badge.csv`.

Exercice 3 : Le calcul des temps des équipes par étape

Q.3.1. Implémentez une fonction `duree_par_etape` qui prend en entrée un dictionnaire décrivant les bornages des équipes tel que celui codé à l'exercice 2 et un dictionnaire décrivant la structure des étapes tel que celui codé à l'exercice 1. Cette fonction devra retourner un dictionnaire ayant :

— pour clés des identifiants d'étapes ;
— pour valeur une liste de tuples de la forme `(equipe_id, duree)` où durée est le temps mis pour l'équipe en question lors de l'étape courante.

Q.3.2. Testez votre fonction en lui fournissant en entrée les dictionnaires retournés par les fonctions codées dans les deux exercices précédents. Vous devriez obtenir une sortie du type :

```
{'Etape 1': [('E1', datetime.timedelta(seconds=9300)),
             ('E2', datetime.timedelta(seconds=5520)),
             ('E3', datetime.timedelta(seconds=5640)),
             ('E4', datetime.timedelta(seconds=6240)),
             ('E5', datetime.timedelta(seconds=6180)),
             ('E6', datetime.timedelta(seconds=7740)),
             ('E7', datetime.timedelta(seconds=5700)),
             ('E8', datetime.timedelta(seconds=8760))],
...
```

Exercice 4 : Les grands vainqueurs

C'est le moment tant attendu, celui de connaître l'équipe vainqueure de l'édition 2024 de Rennes 2 Express.

Pour cela, il vous reste à implémenter une dernière fonction, qui prend en entrée un dictionnaire tel que celui retourné par votre fonction `duree_par_etape` et qui affiche le fameux classement général. Alors, qui est le grand vainqueur ? Mais attention, on ne veut pas ici un identifiant d'équipe, mais bien les noms et prénoms des vainqueurs. À vous de retrouver cette information :)

Autrement dit, votre tâche se décompose en deux étapes :

Q.4.1. Écrire une fonction `duree_totale_par_equipe` qui prend en entrée un dictionnaire tel que celui retourné par votre fonction `duree_par_etape` et qui retourne une structure de données qui stocke, pour chaque équipe, la durée totale du trajet de l'équipe ;

Q.4.2. Écrire une fonction `vainqueurs` qui, à partir du résultat de la fonction précédente, retourne les noms et prénoms des vainqueurs.

Exercice 5 : Pour aller plus loin : le classement par étape

Exercice 6 : Classement par étape

Q.6.1. Améliorez votre fonction `duree_par_etape` pour que les listes associées aux étapes soient triées par ordre de classement. Vous devriez obtenir une sortie du type :

```
{'Etape 1': [('E2', datetime.timedelta(seconds=5520)),
             ('E3', datetime.timedelta(seconds=5640)),
             ('E7', datetime.timedelta(seconds=5700)),
             ('E5', datetime.timedelta(seconds=6180)),
             ('E4', datetime.timedelta(seconds=6240)),
             ('E6', datetime.timedelta(seconds=7740)),
             ('E8', datetime.timedelta(seconds=8760)),
             ('E1', datetime.timedelta(seconds=9300))],
...
```

Q.6.2. Implémentez une fonction qui prend pour entrée un dictionnaire tel que celui retourné par la fonction `duree_par_etape` et qui affiche le classement de chacune des étapes de manière élégante. Quelque chose comme :

```
Classement de l'étape Etape 1
1. Equipe E2 (92 minutes)
2. Equipe E3 (94 minutes)
3. Equipe E7 (95 minutes)
4. Equipe E5 (103 minutes)
5. Equipe E4 (104 minutes)
6. Equipe E6 (129 minutes)
7. Equipe E8 (146 minutes)
8. Equipe E1 (155 minutes)
Classement de l'étape Etape 2
1. Equipe E7 (126 minutes)
2. Equipe E3 (132 minutes)
...
```

Q.6.3. Améliorez votre fonction implémentée à la question précédente pour rajouter, pour chaque étape, la distance à parcourir (obtenue à l'aide de GraphHopper). Vous définirez autant de fonctions qu'il vous semble nécessaire pour rendre votre code le plus lisible possible.

Q.6.4. Supposons qu'au lieu d'additionner les temps de toutes les étapes, on ait plutôt effectué un classement par points selon la règle suivante : le vainqueur d'une étape gagne 5 points, le deuxième 2 points et le 3ème 1 point. Implémentez une fonction qui calcule et affiche l'ordre des équipes dans ce classement par points.