TD8 : Données du réseau STAR

Le réseau de transports en commun de l'agglomération rennaise (la STAR) fournit des données relatives à ses services en libre accès via une API de données. Dans ce TD, vous effectuerez des requêtes à cette API en utilisant le module `requests`.

Préambule

— Dans votre dossier PythonOpenData créez un sous-dossier TD8.

— Lancer l'éditeur Visual Studio Code.

— Dans Visual Studio Code, ouvrir le dossier PythonOpenData/TD8 et créez un fichier td8.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 :

```
from pprint import pprint

[...]
pprint(mon_joli_dictionnaire)
```

Exercice 1 : Accès aux données

Q.1.1. Se rendre sur le site de la STAR, sur l'API qui indique les prochains passages de métro. (`https://data.explore.star.fr/explore/dataset/tco-metro-circulation-passages-tr/information/`)

Q.1.2. Parcourir le tableau pour comprendre les données disponibles. Combien de données sont disponibles avec une précision "Temps réel" ?

Q.1.3. Cliquer sur l'onglet "API" pour accéder aux options de requête.

Q.1.4. En utilisant l'interface d'édition de requêtes de l'API de la STAR, composer une requête pour

— ne conserver que les passages pour lesquels l'attribut `precision` vaut `Temps réel` (valeur à spécifier dans la catégorie *refine*)

— retourner les 100 prochains passages.

Q.1.5. Copier l'URL générée.

Q.1.6. En python, écrire le code permettant d'accéder aux passages de métro renvoyés par cette URL, au format json.

Q.1.7. Combien de passages de métro sont renvoyés ? Comparer le nombre d'enregistrements présents dans `results` et le nombre `total_counts` annoncé en début de requête.

Q.1.8. Stocker la liste des passages (clé `results`) dans une variable `liste_passages_bruts`.

Q.1.9. Il manque quelques passage au dessus de 100 dans la liste `liste_passages_bruts`. La limite de requête est 100, mais il est possible d'indiquer un offset (décalage) sur les résultats. En s'aidant de l'interface d'API, réaliser une requête supplémentaire en Python, pour compléter la liste `liste_passages_bruts` avec le nombre total de métros en temps réel.

Exercice 2 : Etude de tous les passages

Les horaires fournis par la STAR sont fournis avec précision du fuseau horaire (timezone). Curieusement, les horaires ne sont pas homogènes entre l'arrivée et le départ. Ainsi, un passage peut être défini avec les horaires suivants :

depart `2023-11-17T07:47:09+00:00` : il s'agit d'une heure au format ISO, avec précision du fuseau horaire (timezone). Ici, elle est exprimée dans le fuseau horaire avec UTC+0 (`+00:00`), ce qui correspond à 8h47 et 9 secondes en France.

arrivee `2023-11-17 08:46:58+0100` : il s'agit d'une heure *presque* au format ISO puisqu'il manque le caractère T pour séparer la date et l'heure. Ici, elle est exprimée dans le fuseau horaire avec UTC+1 (`+0100`), ce qui correspond à 8h46 et 58 secondes en France.

Q.2.1. Écrire une fonction `extrait_infos_passages()` qui prend en entrée une liste de type `liste_passages_bruts` et qui renvoie une liste de *passages de métros*. Chaque *passage de métro* sera représenté par un dictionnaire disposant des clés suivantes :

- `destination` contenant la destination,
- `nomarret` contenant le nom de l'arret,
- `ligne` contenant le nom court de la ligne.
- `depart` (contenant l'heure de départ au format `datetime`),
- `arrivee` (contenant l'heure de départ au format `datetime`),

Attention : pour certains passages, l'attribut `"depart"` ou `"arrivee"` n'existe pas ou la valeur associée est None : ces passages doivent donc être ignorés.

Note : on choisira probablement une méthode de création de datetime différente pour l'arrivée et le départ. Dans un format, `%z` représente la timezone.

Tester cette fonction.

Q.2.2. Écrire une fonction qui prend en entrée une liste de passages telle que celle créée par la question précédente et calcule le temps moyen passé dans la station pour chaque passage de métro entre le départ et l'arrivée. Tester cette fonction (le résultat attendu est de l'ordre d'une vingtaine de secondes).

Exercice 3 : Prochains métros dans la station

On souhaite renseigner un panneau d'affichage en entrée d'une station de métro, avec l'horaire de chaque prochain passage, pour chaque ligne de métro, dans chaque direction (voir exemple à la fin du sujet).

Q.3.1. Écrire une fonction qui prend en entrée le nom d'une station de métro, la la liste des passages de métro complète, et qui renvoie la liste des passages à cette station.

Q.3.2. Tester cette fonction pour afficher les passages à la station Gares.

Q.3.3. Écrire une fonction qui prend en entrée une liste de passages dans une station, telle que retournée ci-dessus, et qui recherche pour chaque ligne de métro, et chaque direction l'heure du prochain départ à venir.

Les données seront retournées sous la forme d'un dictionnaire dans lequel les clés sont des tuples `(ligne, destination)` et les valeurs la date du premier prochain départ au format datetime.

Note : les objets Datetime de Python qui intègrent la connaissance de la timezone sont dits "aware", par opposition aux dates sans timezone qui sont "naïve". Il n'est pas possible d'effectuer des opérations (comparaisons, différences) entre une date naïve et une date aware. Par exemple, pour comparer une date aware avec la date actuelle, il faudra créer une version aware de datetime.now(), avec :

```
from pytz import timezone
datetime.now(timezone('Europe/Paris'))
```

Q.3.4. Écrire une fonction qui prend en entrée un nom de station et une liste de passages complète, et qui utilise les fonctions précédemment codées, afin d'afficher l'horaire des prochains passages en entrée de la station de métro.

L'affichage pourra être par exemple :

```
****************************************
Bienvenue à la station Gares
****************************************
Ligne a, direction La Poterie : prochain métro à 16:58:39
Ligne a, direction J.F. Kennedy : prochain métro à 16:58:04
Ligne b, direction Saint-Jacques - Gaîté : prochain métro à 16:59:02
Ligne b, direction Cesson - Viasilva : prochain métro à 16:58:09
****************************************
```

Note : Pour forcer l'affichage d'un horaire dans le bon fuseau horaire, on pourra utiliser :

```
madate.astimezone(timezone("Europe/Paris")).strftime(format)
```

Q.3.5. Tester pour plusieurs stations de métro.