API REST d'accès aux données

Planche de TD pour un cours dispensé à l'université de Rennes 2

1 Rappel : organisation de votre code

Pour ce TD, vous créerez un nouveau fichier `td_api_rest.py`. Dans ce fichier, votre code sera organisé de la manière suivante :

```
# Imports
import requests
import json
import datetime

# Fonctions
def [...]

# Tests
[...]
```

Notamment, vous définirez vos fonctions en début de fichier et les appels seront listés en fin de fichier. De cette manière, vous pourrez, d'une question à l'autre, réutiliser les fonctions déjà codées au besoin.

2 Énoncé

1. Se rendre sur le site de la STAR et trouver l'API indiquant les prochains passages de métro rennais. Cliquez sur l'onglet "API" pour accéder aux options de requête. Essayez notamment d'ajouter le *facet* `"depart"` et notez le format de date utilisé.

   Petit point sur :
   UTC, Temps Universel Coordonné, (Coordinated Universal Time) est une échelle de temps adoptée comme base universelle. En France, nous avons une heure d'avance sur ce temps de référence. Pour représenter une date complète utilisable à l'international l'API de la Star utilise le modèle suivant `"jourThoraire+timezone"` en trois parties avec pour séparateurs `"T"` et `"+"` où :

— jour = `"aaaa-mm-jj"` ,
— horaire = `"hh:mn:sc"`,
— timezone `IN {"00:00", "01:00"...}`

Ainsi :

— si timezone vaut `"01:00"` il s'agit de l'heure "en France",
— si timezone vaut `"00:00"`, il y a une heure de retard

2. Écrire une fonction qui prenne en entrée une chaîne de caractères représentant une date et retourne une date en ignorant la "timezone" (fuseau horaire, partie à partir du symbole `"+"` dans la chaîne de caractères). La date devra être de la même forme que dans l'exemple suivant : `"2019-11-23T09:01:52+00:00"`. **Attention :** vous devrez considérer deux cas de figure différents :

   a. si la fin de la chaîne de caractères est `"+00:00"` : il faudra ajouter 1h à la date extraite car le fuseau horaire n'est pas le nôtre ;
   b. si la fin de la chaîne de caractères est `"+01:00"` : la date sera considérée comme correcte.

3. Écrire une fonction qui retourne la liste de tous les passages de métro à venir. Cette fonction fera une requête API, en limitant le nombre de résultats à 100 lignes. La liste retournée par cette fonction contiendra des dictionnaires composés de 3 clés : `"depart"` (contenant l'heure de départ au format `datetime`), `"destination"` et `"nomarret"` et vous ne conserverez que les passages pour lesquels l'attribut `"precision"` vaut `"Temps réel"`. **Attention :** pour certains passages, l'attribut `"depart"` n'existe pas : ces passages doivent donc être ignorés

4. Écrire une fonction qui prenne en entrée une liste de passages tels que ceux retournés par la question précédente et un délai `t` en minutes et qui retourne la liste des passages qui auront lieu dans un délai de `t` minutes après l'instant présent. Tester cette fonction en affichant la liste des prochains passages de métro dans les 10 minutes à venir.

3 Devoir

Cet exercice est à rendre sur CURSUS avant la séance de TD de la semaine prochaine. Le rendu se fera sous la forme d'un unique fichier Python (pas de version `.txt` ou `.pdf`) structuré comme demandé plus haut et contenant le code relatif à cette partie. Ce code devra se suffire à lui-même et devra donc contenir les éventuelles fonctions annexes appelées par ce code.

5. En utilisant le service Open Data de Rennes Métropole, écrivez une fonction qui affiche l'état de la circulation dans la rue de Vezin, et plus précisément, le nombre de points dans cette rue pour lesquels le traffic est fluide (`"freeFlow"`) et le nombre de points pour lesquels le traffic est

chargé (`"congested"` ou `"heavy"`). Votre fonction devra s'assurer que les données récoltées sont récentes (datent de moins de 2h) et afficher un message d'erreur si ce n'est pas le cas.