API web d'accès aux données

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

1 Travail à préparer chez vous avant la séance

— Créez-vous un compte GraphHopper en cliquant ici (vous utiliserez l'URL `http://www.univ-rennes2.fr` pour le champ "site web")
— Créez-vous une clé d'API en vous rendant, connecté avec votre compte, à l'adresse https://graphhopper.com/dashboard/#/api-keys
— Écrivez cette clé d'API dans un fichier JSON nommé `data/credentials.json` au format suivant :

```
{
    "GraphHopper": "..."
}
```

2 Cas des ordinateurs de l'Université de Rennes 2

Sur les ordinateurs de l'Université de Rennes 2, le module `graphh` n'est pas installé par défaut. Pour pouvoir l'utiliser, il va donc falloir commencer par l'installer. Pour cela, vous devrez :

a. Ouvrir le logiciel `Anaconda Prompt` (bienvenue dans le monde merveilleux des lignes de commande :)
b. Entrer la ligne suivante :

```
pip install --user graphh
```

Une fois cela fait, vous pouvez fermer la fenêtre `Anaconda Prompt` et vous devriez pouvoir utiliser ces modules sans soucis (au moins pour la durée de votre session, potentiellement un peu plus que cela si vous restez sur la même machine) dans PyCharm.

3 Énoncé

Le fichier `rando_gps.json` fournit des séries de positions GPS correspondant à des traces GPS de sorties randonnée de M. Toulemonde. On cherchera dans ce TD à écrire un programme calculant les dénivelés cumulés positif et négatif de chacune de ces randonnées. Pour cela, vous utiliserez l'API `GraphHopper` *via* le module Python `graphh`.

3.1 Présentation du module `graphh`

Ce module, dont la documentation est accessible là fournit des fonctions permettant d'interroger l'API GraphHopper.

Pour utiliser cette API, il faut :

a. Charger votre clé d'API lue dans le fichier `data/credentials.json` dans une variable `cle_api`
b. créer un client que l'on initialise en lui fournissant notre clé d'API :

```
gh_client = graphh.GraphHopper(key=cle_api)
```

Enfin, on appelle, pour ce client, la fonction qui nous intéresse. Par exemple, l'instruction :

```
gps_rennes = gh_client.address_to_latlong("Rennes, République")
```

permettra de stocker dans la variable `gps_rennes` une paire de coordonnées GPS correspondant à la position `"Rennes, République"`.

3.2 C'est parti !

1. Écrivez une fonction qui prenne en entrée deux chaînes de caractères décrivant des lieux et une clé d'API GraphHopper et retourne la distance en kilomètres séparant ces lieux.
2. Écrivez une fonction qui prenne en entrée une liste de positions GPS (chacune codée sous la forme d'un dictionnaire comme précisé plus bas) et une clé GraphHopper API et retourne une liste d'altitudes. Vous pourrez utiliser l'exemple suivant pour vos tests :

```
lst_gps = [
  {"lng": -1.426533, "lat": 48.005135},
  {"lng": -1.418127, "lat": 47.986058},
  {"lng": -1.427611, "lat": 47.989871},
  {"lng": -1.430202, "lat": 48.000354}
]
```

3. Écrivez une fonction qui prenne en entrée une liste de positions GPS et une clé GraphHopper API et retourne la somme des dénivelés positifs

(d'une part) et négatifs (d'autre part). Par exemple, si on a une liste de coordonnées GPS pour lesquelles on a obtenu les altitudes suivantes :

```
[38.11, 68.63, 54.60, 36.42]
```

on devrait retourner la paire de valeurs :

```
(30.52, 32.21)
```

5. Écrivez une fonction qui prenne en entrée un nom de fichier JSON (contenant des informations sur diverses randonnées) et une clé GraphHopper API et affiche, pour chaque randonnée, son nom (attribut `"name"`) et la somme de ses dénivelés positifs (d'une part) et négatifs (d'autre part). Pour le fichier `rando_gps.json`, on doit obtenir une sortie du type :

```
TraceGPS Le long de la quincampoix - Pire-sur-Seiche D+:  111.39449691772464 , D-:  111.3871
TraceGPS Issued  Messac - CIRCUIT DU PORT D+:  31.650634765625 , D-:  31.650634765625
TraceGPS Issued  Coemes-Retiers D+:  417.91231536865234 , D-:  417.91231536865223
```