API web d'accès aux données

Aurélie Lemaitre, Romain Tavenard

1 Travail à préparer chez vous avant la séance

1. Préparez la première question de cette planche de TD concernant l'utilisation de l'API Google Maps. Pour cela :

— Créez-vous une clé pour cette API (lien) ;
— Écrivez cette clé d'API dans un fichier JSON au format suivant :

```
{
"API_KEY": "..."
}
```

— Écrivez, pour la position suivante, la requête nécessaire pour obtenir l'altitude du point en question *via* Google Maps Elevation API :
`longitude : -1.426533`
`latitude : 48.005135`
— Effectuez cette requête à l'aide du module `urllib.request` comme vu en cours (votre code devra aller lire votre clé d'API dans le fichier à l'aide du module `json`).

2 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 `Google Maps Elevation` *via* le module Python `googlemaps`.

2.1 Présentation du module `googlemaps`

Ce module, dont la documentation est accessible là, fournit des fonctions permettant d'interroger les différentes API Google Maps (Directions, Elevation, *etc.*).

Pour utiliser cette API, il faut tout d'abord créer un client que l'on initialise en lui fournissant notre clé d'API :

```
gmaps = googlemaps.Client(key=votre_cle_d_api)
```

Ensuite, on appelle, pour ce client, la fonction correspondant à l'API qui nous intéresse. Par exemple, l'instruction :

```
list_routes = gmaps.directions(origin="Rennes",
                               destination="Marseille")
```

permettra de stocker dans la variable `list_routes` une liste d'itinéraires proposés par l'API Directions pour aller de Rennes à Marseille.

Notez que, comme indiqué dans la documentation, les positions géographiques peuvent être passées sous plusieurs formes. On retiendra notamment ici les deux formes les plus utiles :

— une chaîne de caractère définissant un lieu (comme illustré précédemment) ;
— un dictionnaire (ayant deux clés `"lng"` et `"lat"`) identifiant les coordonnées GPS à utiliser.

2.2 Mise en pratique

2. Écrivez une fonction qui prenne en entrée une position GPS et une clé Google Maps Elevation API et retourne l'altitude de la position. La position GPS sera passée sous la forme d'un dictionnaire tel que :

   ```
   coord = {"lng": -1.426533, "lat": 48.005135}
   ```

3. Écrivez une fonction qui prenne en entrée une liste de positions GPS (chacune codée sous la forme d'un dictionnaire tel que précédemment) et une clé Google Maps Elevation 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}
   ]
   ```

4. Écrivez une fonction qui prenne en entrée une liste de positions GPS et une clé Google Maps Elevation 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 Google Maps Elevation 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-:  11
TraceGPS Issued  Messac - CIRCUIT DU PORT D+:  31.650634765625 , D-:  31.650634765625
TraceGPS Issued  Coemes-Retiers D+:  417.91231536865234 , D-:  417.91231536865223
```