# TD1 : API de calcul de trajets

## Travail à préparer avant la séance

— Aller sur le site de graphHopper : https ://graphhopper.com/
— Se créer un compte (sign up). Vous pouvez indiquer "Université Rennes 2" pour le champ Company.
— Vous allez recevoir un mail de confirmation : cliquer sur le lien reçu par mail pour valider votre compte.
— Se connecter avec vos identifiants sur GraphHopper
— Aller dans l'onglet API key
— Créer une clé avec "Add API key". Dans description, vous pouvez indiquer Python L2
— Sauvegarder la chaine textuelle de la clé, de type e7c81e8f-fcxxxx-xxxxx-xxxxx.

## Préambule

1. Créer sur votre disque un dossier PythonOpenData dans lequel vous réaliserez l'ensemble des travaux.
2. Télécharger sur Cursus l'archive contenant les données sources du TD1.
3. Décompresser cette archive sur votre disque. Les fichiers doivent se trouver dans PythonOpenData/TD1.
4. Lancer l'éditeur Visual Studio Code.
5. Dans Visual Studio Code, ouvrir le dossier PythonOpenData/TD1.

## Installation du module graphh

Par défaut, le module graphh n'est pas installé sur votre ordinateur ni sur les machines de l'université. Pour pouvoir l'utiliser, il va donc falloir commencer par l'installer.

Avec Visual Studio Code :

— Ouvrir une fenêtre de terminal par le menu "Affichage - Terminal".
— Taper la commande `pip install graphh`.

Note : cette action est à effectuer une fois pour toutes sur votre machine personnelle, si vous l'utilisez en TD, ou à chaque début de TD sur une machine de l'université si vous travaillez sur un poste de l'université.

## Exercice 1 : Préparation de la clé d'API

1. Dans le fichier credentials.json, insérer votre clé d'API préalablement créée.
2. Dans le fichier td1.py, la fonction fournie `acces_cle_api` permet de lire la clé API dans le fichier credentials.json. Utiliser cette fonction pour afficher votre clé d'API.

## Exercice 2 : Prise en main de GraphHopper

1. Créer un client GraphHopper en utilisant votre clé d'API.
2. Obtenir les coordonnées GPS de "Rennes Beaulieu" ainsi que de "Rennes Villejean".
3. Afficher la distance par la route pour relier les deux campus, de Beaulieu vers Villejean, puis dans le sens inverse.
4. Afficher la durée pour aller de Villejean à Beaulieu en voiture puis en vélo.

## Exercice 3 : Définition de fonctions

1. Créer une fonction `distance_lieux` qui prend en entrée un client GraphHopper, et deux chaines de caractères représentant des lieux, et qui renvoie la distance, en kilomètres, entre ces deux lieux.
2. Tester cette fonction pour calculer la distance de Rennes à Brest.
3. Créer une fonction `duree_lieux` qui prend en entrée un client GraphHopper, et deux chaines de caractères représentant des lieux, et qui renvoie la duree du trajet en voiture, en minutes, entre ces deux lieux.
4. Tester cette fonction pour calculer la duree du trajet de Rennes à Brest.

## Exercice 4 : Voyage à étapes

On définit un voyage comme une liste de lieux (chaines de caractère). Chaque lieu de la liste est une étape du voyage.

Pour toutes les fonctions demandées, penser à chaque fois à utiliser d'autres fonctions existantes !

1. Une variable `voyage1` est définie dans le fichier source. Définir un second voyage de votre choix.
2. Définir une fonction `distances_etapes` qui prend en entrée un voyage, un client GraphHopper, et renvoie la liste des distances de chaque étape. Tester sur les deux voyages.
3. Définir une fonction `distance_totale` qui prend en entrée un voyage et un client GraphHopper, et renvoie la distance totale du trajet en passant par chacune des étapes.