TD2 : Problèmes de trajets, co-voiturages et copains

Préambule

1. Dans votre dossier PythonOpenData, créer un sous-répertoire TD2 dans lequel vous réaliserez l'ensemble des travaux.
2. Télécharger sur Cursus l'archive contenant les données sources du TD2.
3. Lancer l'éditeur Visual Studio Code de puis Anaconda Navigator.
4. Dans Visual Studio Code, ouvrir le dossier PythonOpenData/TD2.
5. Copier le fichier credentials.json du TD1 dans le dossier PythonOpenData/TD2.

Ce TD présente trois problèmes de calculs de trajets. Pour chacun des problèmes, on pourra utiliser des fonctions disponibles dans le fichier source td2.py.

Exercice 1 : Quel co-voiturage ?

Dans cette partie, on suppose que l'on cherche à déterminer, parmi plusieurs choix possibles, un trajet optimal avec les contraintes suivantes :

— les points de départ et d'arrivée sont fixés

— on doit récupérer, sur le trajet, une personne en covoiturage et l'on peut choisir entre plusieurs personnes situées à des positions différentes.

La question peut par exemple être posée sous la forme suivante : *Lors d'un trajet Rennes - Marseille, vaut-il mieux (en termes de temps de trajet) prendre quelqu'un à "Paris 14ème arrondissement", "Lyon 1er arrondissement" ou "Bordeaux" ?*

Écrire une fonction qui prend en entrée :

— un lieu d'origine (sous la forme d'une chaîne de caractères)

— une destination (sous la forme d'une chaîne de caractères)

— une liste d'options sur le trajet (sous la forme d'une liste de chaînes de caractères)

— un client d'API GraphHopper

et retourne l'option correspondant au trajet le plus court (en temps). En cas d'égalité, votre fonction retournera *l'une des options* correspondant au trajet le plus court.

Exercice 2 : Où se retrouver ?

Plusieurs ami(e)s souhaitent se retrouver, tout en minimisant leur distance de trajet. Plus précisément, ils souhaitent que la somme des distances parcourues par eux tous pour se retrouver soit la plus faible possible.

La question peut par exemple être posée sous la forme suivante :

Pour trois amis habitant respectivement "Paris, 12ème arrondissement", "Auxerre" et "Lyon, 1er arrondissement", est-il préférable (selon le critère énoncé ci-dessus) de se rencontrer à Rennes, à Strasbourg ou à Dijon ?

Écrire une fonction qui prend en entrée :

— une liste de positions des amis (chaque position étant représentée par une chaîne de caractères)

— une liste de lieux de rencontres possibles (sous la forme d'une liste de chaînes de caractères)

— un client d'API GraphHopper

et retourne l'option la plus favorable (par rapport au critère énoncé plus haut). En cas d'égalité, votre fonction retournera *l'une des options* les plus favorables.

Exercice 3 : Trouver un partenaire sportif

Ici, on suppose que l'on souhaite faire du sport et que, pour un sport donné, on veut chercher dans une liste de partenaires potentiels celui ou celle qui se trouve le plus proche de nous.

La question peut par exemple être posée sous la forme suivante :

Voici ma liste de contacts :

— *Pauline, joue au tennis et au squash, se trouve "Place du recteur Henri Le Moal, Rennes, France"*

— *Ernest, joue au football et pratique la course à pied, se trouve "Place du Parlement de Bretagne, Rennes, France"*

— *Felix, joue au tennis et au football, se trouve "182 rue de l'Alma, Rennes, France"*

— *Sarah, joue au football, au squash et au tennis, se trouve "88, rue Alphonse Guérin, Rennes, France"*

— *Ingrid, pratique la course à pied, se trouve "3 Mail François Mitterrand, Rennes, France"*

Sachant que je me trouve Place de la République à Rennes, quel est, dans ma liste de contacts, celui ou celle qui se trouve le plus proche de moi pour jouer au tennis ?

Écrire une fonction qui prend en entrée :

— une position (représentée par une chaîne de caractères)

— une liste de contacts (sous la forme d'une liste de dictionnaires telle que `liste_dicos` dans le fichier source)

— une activité physique (représentée par une chaîne de caractères)

— un client d'API GraphHopper

et retourne le nom de la personne dont la localisation est la plus proche de nous et qui pratique le sport voulu. En cas d'égalité, votre fonction retournera le nom *de l'une des personnes* dont la localisation est la plus proche de nous et qui pratique le sport voulu.