TD6 : Données CSV des transports bretons

La région Bretagne fournit des données sur les lignes de transports en communs. Il est possible de télécharger les données au format CSV. Notre objectif est de produire une carte de ces données.

Note : nous utiliserons ici des données brutes, directement téléchargées sur le site web du fournisseur de données. Une grande partie du travail va donc consister à homogénéiser et rendre accessible le format des données.

Préambule

— Créer sur votre disque un dossier TD6 dans PythonOpenData.

— Télécharger sur Cursus l'archive contenant les données sources du TD6, et la décompresser.

— Lancer l'éditeur Visual Studio Code.

— Dans Visual Studio Code, ouvrir le dossier PythonOpenData/TD6.

Exercice 1 : Chargement et analyse des données

Q.1.1. Sur le site web `https://data.bretagne.bzh/explore/dataset/mobibreizh-lignes/export/`, télécharger les données au format CSV du jeu de données « MOBIBREIZH - Lignes », et enregistrer le fichier csv dans le dossier PythonOpenData/TD6.

Q.1.2. Visualiser avec Visual Studio Code le contenu du fichier csv. Quel est le séparateur utilisé ? Quels sont les champs présents dans le fichier ? Noter que certaines lignes sont très longues...

Q.1.3. Trouver une ligne sur laquelle tous les champs ne sont pas renseignés : constater que les coordonnées sont parfois manquantes.

Q.1.4. Donner 2 exemples de valeurs indiquées dans la colonne "color". A quoi ces valeurs correspondent-elles ?

Exercice 2 : Préparation des données

Dans la suite de ce sujet, on nomme « trajet » chaque ligne issue du fichier csv. Tout le code produit devra être inclus dans le fichier td6.py.

Q.2.1. La fonction fournie `taille_max` prend en entrée un nom de fichier texte, et renvoie la longueur maximale d'une ligne dans ce fichier texte. Utiliser cette fonction pour afficher la taille maximale des lignes dans le fichier mobibreizh-lignes.csv.

Q.2.2. Charger le contenu du fichier mobibreizh-lignes.csv dans une liste de dictionnaires, chaque dictionnaire représentant un trajet.

Q.2.3. Afficher le nombre de trajets disponibles.

Q.2.4. Afficher les clés du dictionnaire disponibles pour un trajet.

Q.2.5. Nous allons supprimer les trajets pour lesquels les coordonnées ne sont pas fournies. Ecrire une fonction `elimine_sans_coordonnnees` qui prend en entrée la liste des trajets, et renvoie une liste des trajets dans laquelle on a enlevé les trajets pour lesquels le champ *geo_point_2d* est vide. Combien de trajets sont maintenant disponibles ?

Q.2.6. Ecrire une fonction `liste_couleurs` qui prend en entrée la liste des trajets, en renvoie la liste des valeurs du champ *Color*. Utiliser cette fonction pour afficher la liste des couleurs. Le format des couleurs est-il homogène ?

Q.2.7. Il faut parfois ajouter le symbole `#` devant les couleurs lorsqu'il n'est pas présent. Ecrire une fonction `corrige_couleurs` qui prend en entrée la liste des trajets, et renvoie la liste des trajets dans laquelle les couleurs ont été corrigée, par ajout du symbole `#`, lorsque c'est nécessaire. Utiliser à nouveau la fonction `liste_couleurs` pour s'assurer que les corrections sont effectives.

Exercice 3 : Prise en main du module cartographie

Il est mis à votre disposition un fichier `cartographie.py` contenant plusieurs fonctions codées par les enseignants. Ces fonctions permettent de créer et d'afficher une carte dans un navigateur web.

Voici les fonctions disponibles :

Nom de la fonction	Paramètres d'entrée	Valeur renvoyée
`creer_carte(titre)`	`titre` : titre à afficher au dessus de la carte	la carte créée
`tracer_point(carte, long,lat,label, couleur)`	`carte` : la carte sur laquelle afficher le point `long` : la longitude du point, ex : -1.65 `lat` : la latitude du point, ex : 48.652 `label` : l'étiquette à afficher au survol du point `couleur` (facultatif) : la couleur du point, au format chaine de caractère ex : `"green"` ou hexa ex : `#45A6B9`.	aucun
`tracer_ligne(carte, lst_long,lst_lat, label,couleur)`	`carte` : la carte sur laquelle afficher la ligne `lst_long` : la liste des longitudes de la ligne, ex : [-1.65, -1.67, -1.85] `lst_lat` : la liste des latitudes de la ligne, ex : [48.65, 47.25, 46.23] `label` : l'étiquette à afficher au survol de la ligne `couleur` (facultatif) : la couleur de la ligne, au format chaine de caractère ex : `"green"` ou hexa ex : `#45A6B9`.	aucun
`afficher_carte(carte)`	`carte` : la carte à afficher	aucun

Pour réaliser une carte, il faut d'abord la créer, puis tracer tous les élements nécessaires, et enfin demander l'affichage de cette carte.

Q.3.1. En utilisant les fonctions ci-dessus, créer une carte ayant pour titre "Bretagne", y tracer un point aux coordonnées de la ville de Rennes (-1.6742900,48.1119800), en bleu, et afficher la carte.

Q.3.2. Ajouter un point sur cette carte pour Saint-Malo, aux coordonnées (-2.025674, 48.649337), en jaune.

Q.3.3. Ajouter une point sur cette carte pour Nantes, aux coordonnées (-1.553621, 47.218371), en vert.

Q.3.4. Ajouter une ligne sur cette carte, reliant Nantes, Rennes et Saint-Malo, en couleur `#FF5533`.

Exercice 4 : Cartographie des points des trajets

L'objectif de cette partie est de tracer sur une carte les points de chaque trajet, représentés dans le fichier dans la colonne *'geo_point_2d'*. Note : on trace bien ici des points et pas des lignes.

Q.4.1. Écrire une fonction `ajoute_point_trajet` qui prend en paramètre une carte, un trajet (un dictionnaire), et trace sur la carte le point correspondant de coordonnées *'geo_point_2d'*, en utilisant la couleur *Color*, et en utilisant le *long_name* en tant que légende.

Q.4.2. Écrire une fonction `carte_tous_points_2d` qui prend en entrée la liste des trajets, qui crée une carte, ajoute un point pour chacun des trajets puis affiche la carte.

Exercice 5 : Pour aller plus loin : cartographie des lignes des trajets

L'objectif de cette partie est de tracer sur une carte les lignes de chaque trajet, représentées dans le fichier dans la colonne *'Shape'*. La colonne *'Shape'* du fichier csv contient une structure, de type dictionnaire. Le champs *coordinates* contient plusieurs listes de coordonnées, on se focalise sur la première liste de coordonnées.

Q.5.1. Afficher le champ *Shape* du premier trajet. Quel est son type ?

Q.5.2. Il faut préparer les données pour en extraire les listes de coordonnées. Écrire une fonction `prepare_lst_coord` qui prend en entrée un trajet et qui renvoie le trajet enrichi avec les listes de coordonnées préparées pour ce trajet. Il faudra notamment :

- convertir la chaine de caractère du champ *Shape* en dictionnaire, en utilisant `json.loads(chaine)`
- accéder à la première liste des coordonnées,
- créer et remplir deux listes `lst_long` et `lst_lat`,
- ajouter ces deux listes comme 2 éléments du dictionnaire du trajet.

Q.5.3. Écrire une fonction `ajoute_ligne_trajet`, qui prend en paramètre une carte, un trajet (un dictionnaire), et trace sur la carte la ligne de coordonnées *'lst_long'* et *lst_lat*, en utilisant la couleur *Color*, et en utilisant le *long_name* en tant que légende.

Q.5.4. Écrire une fonction `carte_toutes_lignes` qui prend en entrée la liste des trajets, qui crée une carte, ajoute un point pour chacun des trajets puis affiche la carte.