**Travail pratique 2 –**

**Gestion de données géographiques**

|  |
| --- |
| * **Pondération :** 10% de la note finale du cours * **Échéance de remise :** mardi 21 mai 2023 à 23 :55 * **Où Remettre :** Une fois le travail terminé, remettre **uniquement** votre dossier zippé **NomPrenomx2.zip** sur LÉA dans l’espace intitulé « TP2 » contenant vos fichiers py. * Travail à réaliser en équipe de 2 dans un même groupe de cours (obligatoirement). Veuillez m’écrire sur Léa si vous n’avez pas trouvé de coéquipier dès la réception de cet énoncé. Attention, le professeur peut, s’il le juge nécessaire, poser des questions en regard au code informatique remis pour s’assurer qu’il ou elle est l’auteur authentique du travail remis. * Vous aurez à présenter le programme en exécutant les différentes options de celui-ci devant le professeur. * Tout plagiat sera sévèrement sanctionné, et la peine minimale encourue sera la note zéro attribuée à l’étudiant. * Prenez soin de vous identifier en entête du document en inscrivant vos informations personnelles (Numéro de matricule, date, nom et prénom) en commentaire selon le format ci-dessous :   **'''**  **Votre description du programme**  **@auteur(e)s  X.prenom X.nom et Y.prenom Y.nom**  **@matricules eXXXXXX et eYYYYYY**  **@date  20-05-2024**  **'''** |

# Objectifs du travail pratique

Ce travail pratique consiste à créer un programme en Python qui gère des données géographiques. Le travail devra obligatoirement être réalisé avec git en mode ligne de commande et GitHub comme outil de collaboration.

# Mise en situation

On vous demande de créer un programme en Python qui affiche à l’écran les données (ville, pays, latitude, longitude), à partir d’un fichier de données .csv.

On veut également sauvegarder des données dans un fichier .json. On veut également afficher les données relatives à la distance la plus courte entre deux villes lues du fichier.

## Fichier de données

Le fichier de données possède 4 valeurs, séparées par des virgules, soit

* Le nom de la ville
* Le nom du pays
* La latitude en degrés décimaux.
* La longitude en degrés décimaux.

tel qu’obtenu du site web suivant : <https://www.coordonnees-gps.fr/zone-rayon-100km>

|  |
| --- |
| Ville, Pays, Latitude, Longitude  Paris,France, 48.8534951, 2.3483915  Londres, Angleterre, 51.5073219, -0.1276474  Hong Kong, Chine, 22.2793278, 114.1628131  New York, États-Unis d'Amérique, 40.7127281, -74.0060152  Montréal, Canada, 45.5031824, -73.5698065  Berlin, Allemagne, 52.5170365, 13.3888599 |

## La classe DonneesGeo

Créer une classe nommée DonneesGeo.

Les attributs de cette classe sont les suivantes :

Attributs :

ville : Cette donnée est de type chaine de caractère.

pays : Cette donnée est de type chaine de caractère.

latitude: Cette donnée est conservée en degré de type float.

longitude : Cette donnée est conservée en degré de type float.

Méthodes :

Constructeur

Le constructeur initialise la ville, le pays, la latitude et la longitude.

La méthode \_\_str\_\_ qui retourne la valeur des attributs dans une chaine de caractères.

## La fonction lireDonneesCsv(nomFichier)

Implémenter une fonction lireDonneesCsv qui prend en paramètre le nom d’un fichier csv et lit les données dans ce fichier avec les méthodes de lecture de votre choix. La fonction retourne une liste contenant les informations de données géographique (une liste d’objets DonneesGeo). Utiliser le module csv pour lire le fichier.

## La fonction ecrireDonneesJson(nomFichier, listeObjDonneesGeo)

Implémenter une fonction ecrireDonneesJson qui prend en paramètre le nom d’un fichier json et une liste d’objets DonneesGeo. Cette fonction convertira cette liste d’objets en liste de dictionnaires et sauvegardera cette structure dans le fichier passé en paramètre en format indenté. Utiliser le module json de Python pour écrire le fichier.

## La fonction trouverDistanceMin(nomFichier)

Implémenter une fonction trouverDistanceMin qui prend en paramètre le nom d’un fichier json. Cette fonction lira le fichier .json, permettra de rechercher et d’afficher à l’écran les données des villes parmi celles apparaissant dans le fichier avec la distance minimale avec la phrase suivante :

Distance minimale en kilomètres entre 2 villes : Ville 1 : <ville> <pays> <latitude> <longitude> et Ville 2 : <ville> <pays> <latitude> <longitude> Distance en kilomètres : <distance>

La manière simple de calculer une distance entre deux points consiste à utiliser la formule de Pythagore pour calculer l'hypoténuse d'un triangle (A² + B² = C²). Ceci est connu comme la distance euclidienne. Cependant, cela ne s'applique pas à la géographie car la distance la plus courte entre deux points sur une sphère est différente de la distance entre deux points sur une carte plate. De plus, les lignes de latitude et de longitude ne sont pas équidistantes. La formule de Haversine permet de déterminer la [distance du grand cercle](https://fr.wikipedia.org/wiki/Distance_du_grand_cercle) entre deux points d'une [sphère](https://fr.wikipedia.org/wiki/Sph%C3%A8re), à partir de leurs [longitudes](https://fr.wikipedia.org/wiki/Longitude) et de leurs [latitudes](https://fr.wikipedia.org/wiki/Latitude). Elle se calcule ainsi :



où

r = rayon de la terre (6371 km)

d = distance entre 2 points en kilomètres

Les latitudes : φ1 et φ2 en radians

Les longitudes : λ1 et λ2 en radians

Utilisez les instructions import math, math.radians(degre), math.cos(x), math,sqrt(x), math.pow(x,y), math.sin(x), math.asin(x).

Sauvegardez tous les calculs de distances dans le fichier distances.csv dans le format suivant : ville1, ville2, distance

## Menu

Lorsqu’on lance le programme, un menu doit s’afficher à l’écran et attendre une entrée de l’utilisateur. Le menu doit afficher :

|  |
| --- |
| 1. Lire les données du fichier csv, créer les objets et afficher les données. 2. Sauvegarder les données dans un fichier .json. 3. Lire les données du fichier .json, déterminer et afficher les données associées à la distance minimale entre deux villes et sauvegarder les calculs dans distances.csv.   Entrez un numéro pour choisir une option ou appuyez sur 'q' pour quitter : |

* Si l’utilisateur appuie sur 1, la première option doit être exécutée
* Si l’utilisateur appuie sur 2, la deuxième option doit être exécutée
* Si l’utilisateur appuie sur 3, la troisième option doit être exécutée
* Si l’utilisateur appuie sur **q**, le programme doit se terminer
* Si l’utilisateur appuie sur une autre touche que les 4 précédentes, rien ne doit arriver et le menu doit attendre une autre entrée de l’utilisateur
* Lorsqu’une option est exécutée, le menu doit être réaffiché.
* Note : Valider que l’utilisateur a fait le choix 1 avant de faire le choix 2. Valider que l’utilisateur a fait le choix 2 avant de faire le choix 3.

**Menu option 1**

L’option 1 doit lire le fichier Donnees.csv et les afficher à l’écran.

La fonction doit lire le fichier de données et afficher

* Une ligne de titre
* Attendre que l’utilisateur appuie sur une touche
* Afficher les données du fichier

**Menu option 2**

Cette option doit sauvegarder les données dans un fichier json.

**Menu option 3**

L’option 3 doit afficher les données associées à la distance minimale entre deux villes du fichier et sauvegarder les calculs dans le fichier distances.csv.

Ouvrez Donnees.csv **avec Bloc-Notes ou Notepad** et copiez le contenu ci-dessous :

|  |  |  |
| --- | --- | --- |
| |  | | --- | | Ville, Pays, Latitude, Longitude  Paris, France, 48.8534951, 2.3483915  Londres, Angleterre, 51.5073219, -0.1276474  Hong Kong, Chine, 22.2793278, 114.1628131  Montréal, Canada, 45.5031824, -73.5698065  Berlin, Allemagne, 52.5170365, 13.3888599  Singapour, Singapour, 1.357107, 103.8194992  Tokyo, Japon, 35.6821936, 139.762221  Johannesburg, Afrique du Sud, -26.205, 28.049722  Mexico, Mexique, 19.4326296, -99.1331785  Pékin, Chine, 40.190632, 116.412144  Melbourne, Australie, -37.8142454, 144.9631732  Rio de Janeiro, Brésil, -22.9110137, -43.2093727  Hawaï, États-Unis d'Amérique, 19.5938015, -155.4283701 | | Athènes, Grèce, 37.9755648, 23.7348324  Mumbai, Inde, 18.9733536, 72.8281049  Lagos, Nigéria, 6.4550575, 3.3941795  Buenos Aires, Argentine, -34.6037181, -58.38153  Los Angeles, États-Unis d'Amérique, 34.0536909, -118.242766  Reykjavik, Islande, 64.145981, -21.9422367  Santiago, Chili, -33.4377756, - 70.6504502  Vancouver, Canada, 49.2608724, -123.113952  Manille, Philippines, 14.5906346, 120.9799964  Oslo, Norvège, 59.9133301, 10.7389701  Séoul, Corée du Sud, 37.5666791, 126.9782914 | |

## Diagramme de flux de données

Les trois flèches de gauche à droite :

* Flèche 1 : Lecture du fichier de données .csv. Les informations lues sont utilisées pour créer des objets DonneesGeo en initialisant leurs attributs.
* Flèche 2 : Sauvegarde des informations contenues dans les objets DonneesGeo dans un fichier .json.
* Flèche 3 : Lecture du fichier .json auparavant sauvegardé pour recréer les DonneesGeo, traiter ces données pour déterminer parmi la liste de villes, le binôme ayant la distance la plus courte tout en sauvegardant les calculs de distances entre deux villes dans le fichier distances.csv.

## L’utilisation de git en ligne de commande et de GitHub

Vous devrez utiliser git en ligne de commande. Les commandes git init, git add, git commit, git status, git log, git clone, git push, git pull vous seront utiles pour contribuer au travail pratique. GitHub vous permettra de collaborer avec votre coéquipier.

# Grille de correction

|  |  |
| --- | --- |
| Affichage du menu avec les options et accès aux fonctionnalités | /1 |
| Classe DonneesGeo : constructeur et \_\_str\_\_ | /1 |
| Fichier de données csv présent et lu par le programme | /1 |
| Affichage du fichier de données csv à l’écran | /1 |
| Création de la liste d’objets DonneesGeo | /1 |
| Fichier de données json indenté créé et sauvegardé par le programme | /1 |
| Lecture du fichier json et affichage des données pour la distance minimale entre 2 villes avec sauvegarde des calculs dans le fichier distances.csv | /3 |
| Déclarations de variables, constantes significatives.  Commentaires pertinents | /1 |
| Utilisation de git en ligne de commande | /2 |
| Collaboration en utilisant GitHub | /2 |
| Présentation du programme au professeur en classe | /1 |
| **Bonus** : Qualité (Efficience : qualité du code dépassant les attentes du cours) | /+1 |
| **Correction négative** : Non-respect des critères de remise | /-1 |
| **TOTAL** | **/15** |