1 Bibliothèques à importer

Importer les librairies suivantes

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import  pdist
from scipy.spatial.distance import  squareform
R=6367.445
```

2 Coordonnées Géographiques

1. Placer dans un même dossier votre programme et le fichier `villes_normandes.csv` du TP3
2. Charger en mémoire dans `coordN` les colonnes 11 et 12 de `villes_normandes.csv`
   `coordN=np.loadtxt("villes_normandie.csv", delimiter=";", usecols=(11,12), skiprows=1)`
3. `coordN` est une matrice de type `np.array` dont les éléments sont de type `float`, contenant les coordonées géographiques `(latitude, longitude)` en degrés de chaque ville .
4. Si vous n'avez pas réussi à charger `coordN` dans la question précédente :
   — Charger le tableau en entier `coordN=np.loadtxt("villes_normandie.csv", delimiter=";", skiprows=1)`
   — Projeter sur les colonnes 11 et 12 : `coordN=coordN[: , (11,12)]`
   — convertir en un tableau de float : `coordN=coordN.astype(float)`
5. Afficher le tableau `coordN`, afficher sa dimension `coordN.shape`.
6. Pour écourter les temps de calcul, nous n'allons manipuler qu' une partie de `coordN`. Fixer un nombre de villes `nbV=300` ou `nbV=coordN.shape[0]//10`. Dans toute la suite, nous n'allons manipuler que le tableau restreint `coord=coordN[nbV]`. Afficher `coord`.
7. Les coordonnées géographiques `(lat, lon)` dans `coord` sont en degrés. Les convertir en radian :`coord=coord*(np.pi/180)`

3 Distances géodésiques entre villes

8. Dans `coord`, les coordonnées géographiques de `Caen` sont à la ligne `4` et celles de `Falaise` à la ligne `81`, les calculer : `cgCaen= ...`, `cgFalaise= ....`
9. Définir une fonction `distGeo` qui calcule la distance géodésiue entre deux villes à partir de leurs coordonnées géographiques $cg1 = (lat_1, lon_1)$ et $cg2 = (lat_2, lon_2)$.

   ```
   def distG(cg1, cg2):
        return R * np.arccos(np.sin(cg1[0]) * np.sin(cg2[0])  + np.cos(cg1[0])*np. cos(cg2[0])
                       * np.cos(cg1[1]-cg2[1]) )  }
   ```
10. Calculer la distance géodésique entre `Caen` et `Falaise` (environ $30km$)
11. Calculer le vecteur condensé des distances géodésiques entre villes en appliquant `distG` sur les coordonnées géographiques de chaque paire de villes *distinctes* : `vdistG = pdist(coord, distG)`.
12. Transformer `vdistG` en la matrice carrée des distances géodésiques entre villes `mdistG= squareform(vdistG)`
13. En fait, `vdist` représente la partie triangulaire supérieure de `mdistG`
14. Calculez à partir de `mdistG` la distance géodésique entre `Caen` et `Falaise`.

4 Distance euclidienne entre villes

On veut maintenant calculer les distances euclidiennes entre les paires `(v,w)` de villes quelconques :

$$dist(v, w) = ||v - w|| = \sqrt{(x_v - x_w)^2 + (y_v - y_w)^2 + (z_v - z_w)^2}$$

qui peut être calculé avec les coordonnées cartésiennes `cc_v` de `v` et `cc_w` de `w` par `np.linalg.norm(cc_v - cc_w)`

15. On va d'abord construire la matrice $coord3d = [[x_0, y_0, z_0], \dots, [x_i, y_i, z_i], \dots]$ des coordonnées cartésiennes des villes à partir de $coord = [[lat_1, lon_1], \dots, [lat_i, lon_i], \dots]$ et la relation définissant les coordonnées cartésiennes (x, y, z) d'une ville à partir de ses coordonnées géographiques $(lat, long)$:

$$(x, y, z) = (R * cos(lat) * sin(long), R * cos(lat) * cos(long), R * sin(lat))$$

16. Calculer `coord3d` de la façon suivante :
    —
    — récupérer dans `Lat` la colonne des latitudes et dans `Lon` celle des longitudes. Ex. `Lat = coord[ :, 0]` ; afficher la dimension de `Lat` ; assurez-vous que ce soit de la forme `(nbV,1)` et non pas `(nbV,)` : `Lat=Lat.reshape((nbV, 1))` ou `Lat=Lat.reshape((-1, 1))`. Faire de même avec `Lon`
    — calculer la colonne X des abcisses à partir de `Lat` et `Lon` : `X=R*np.cos(Lat) ...`. Afficher la dimension de `X`.
    — calculer de même les colonnes `X` et `Z`.
    — calculer `coord3d` en concaténant les vecteurs colonnes `X`, `Y` et `Z` : `np.concatenate([X, Y, Z], axis= 1)`.
    — Afficher `coord3d` et sa dimension
    — Calculer les coordonnées cartésiennes `ccCaen` et `ccFalaise` de `Caen` et `Falaise`
    — Définir une fonction `distE` qui calcule la distance euclidienne entre deux villes données par leurs coordonnées cartésiennes : `def distE(cc1, cc2) : return np.linalg.norm(cc1-cc2)`.
    — Calculer la matrice des distances euclidiennes entre toutes les villes en appliquant `distE` sur `coord3d` : `mdistE=squareform(pdist(...))`
    — Calculer la distance euclidienne entre `Caen` et `Falaise` avec `distE`.
17. En utilisant la fonction `plt.scatter` de `matplotlib`, affichez chaque ville sous la forme de point de coordonnée `(x,-y)`. Calculer `plt.scatter(X, -Y)` ou `plt.scatter(coord3d[:,0], -coord3d[:,1])`.
18. Redessiner avec toutes les villes normandes : `nbV=coordN.shape[0]`.

5 Ville les plus proches

19. Que contient la ligne 4 (qui est le numéro de lla ville de `Caen`) de la matrice `mDistE` ?
20. Ecrire une fonction `dist_a_partir_ville` qui, étant donné une ville origine `vo`, retourne le vecteur des distances entre les autres villes et `vo`. : `def dist_a_partir_ville(vo): return ....`
21. Calculer le vecteur `caenDist` des distances des villes par rapport à `Caen`
22. Calculer les 10 villes les plus proches de Caen : calculer les dix premiers élements de `indicesCaen=np.argsort(caenDist)[:10`, où `np.argsort(A)` retourne un vecteur d'indices $[j_0, j_1, j_2, j_3 \dots]$ de même taille que A tel que $[A_{j_1}, A_{j_2}, A_{j_3}, \dots]$ est trié.
23. Calculer les noms des dix villes les plus proches de Caen avec `nom_villes [indicesCaen]` où `nom_villes` est obtenu par `nom_villes_normd=np.loadtxt("villes_normandie.csv", delimiter=';', dtype=npstring_, skiprows=1,usecols=8)`
24. De même, trouver les 10 villes les plus éloignées de Caen.

.