UNIVERSITÉ DE CAEN **L2**
U.F.R. des Sciences **Calcul scientifique**

TP 3 : Les villes de France

---

1 Introduction

L'objectif de ce TP est d'aller plus loin dans l'utilisation de numpy en réalisant un programme qui permette de calculer des distances entre villes.

> Sauf si cela est précisé, vous ne devez pas utiliser de boucle (for, while) ou de branchement conditionnel (if) durant ce TP.

2 Analyse du fichier fourni

Nous allons travailler dans ce TP sur une version modifiée d'un fichier recensant l'ensemble des villes de France dont la version original se trouve à l'adresse suivante : `https://www.data.gouv.fr/fr/datasets/listes-des-communes-geolocalisees-par-regions-departements-circonscriptions-nd/`. Vous avez à votre disposition sur la plateforme ecampus deux fichiers appelé `villes_france.csv` et `villes_normandie.csv` contenant des informations sur respectivement l'ensemble des villes de France et de Normandie. Vous pouvez commencer à regarder leurs contenus au moyen d'un éditeur de texte (par exemple gedit), ou l'importer sous Open Office ou Libre Office. Vous verrez que la première ligne contient les intitulés des colonnes. Les colonnes sont séparées par des " ;".

Vous travaillerez en premier sur le fichier `villes_normandie.csv`. Une fois votre code fonctionnel, vous pouvez tester sur l'ensemble des villes de France (`villes_france.csv`).

Vous allez désormais importer les données contenues dans le fichier `villes_normandie.csv` dans des variables python.

Pour cela, nous allons utiliser la fonction `np.loadtxt` de numpy. Regardez pour cela ce que donne le code suivant :

```
nom_ville = np.loadtxt('villes_normandie.csv', delimiter=';',dtype=np.string_,usecols=8,skiprows=1)
print(nom_ville[4].decode('latin_1'))
```

1. À l'aide de cette fonction faite un tableau numpy que vous nommerez nom_ville contenant toutes les villes de Normandie. Vous pourrez gardez le format numpy des chaines de caractères.
2. En utilisant le tableau précédent et une boucle for, créez un dictionnaire python nommé ville donnant pour un nom de ville son numéro correspondant à son ordre d'apparition dans le tableau nom_ville. Par exemple *Audrieu* aura le numéro 0 et *Caen* le numéro 4. On ignorera les doublons et on ne gardera que le numéro de la dernière occurrence. Vous convertirez les chaines de caractères numpy en chaine de caractères python comme dans l'affichage ci-dessus.
3. Créez une matrice numpy *coord* dans laquelle chaque ligne représente une ville, la première colonne contient la latitude et la seconde colonne la longitude de la ville. Pour cela remplacez dans le code suivant les " ?" avec les bonnes valeurs :

```
coord = np.loadtxt('villes_normandie.csv', delimiter=';', usecols=(?,?),skiprows=1)
```

Sauver le dictionnaire et les matrices dans un fichier de type "pickle", cela sera plus facile à charger ultérieurement.

```python
import pickle

# pour écrire les données sur disque avec pickle
with open('data.pickle', 'wb') as f:
    pickle.dump([ville,nom_ville,coord], f)

# pour lire les données sur disque avec pickle
with open('data.pickle', 'rb') as f:
    [ville,nom_ville,coord] = pickle.load(f)
```

3 Distances géodésiques entre villes

Pour la suite du TP, vous utiliserez le fichier de données calculé dans la partie précédente.

1. Dans un premier temps, vous écrirez une fonction qui permet de calculer la distance géodésique entre deux villes, à partir des numéros des villes. Vous utiliserez pour cela l'équation suivante :

   $$d_g = R\cos^{-1}(\sin(p1_{lat}) * \sin(p2_{lat}) + \cos(p1_{lat}) * \cos(p2_{lat}) * \cos(p1_{lon} - p2_{lon}))$$

   où $R = 6367.445$ est le rayon de la terre (en km), $p1$ et $p2$ les deux points dont on souhaite calculer la distance. Les indices $_{lat\ lon}$ représentent la latitude et la longitude en radian. Attention, dans le ficher de données, les valeurs sont données en degrés.
2. Vérifiez que votre fonction donne un résultat correct en calculant la distance entre Caen et Rouen et en le comparant au résultat donné par le site `https://www.ephemeride.com/atlas/distance/27/` qui permet de calculer les distances à vol d'oiseau entre deux villes.
3. Nous désirons désormais calculer une matrice contenant l'ensemble des distances entre villes. La valeur (i, j) de cette matrice représentera la distance entre la ville i et la ville j.

   Construire cette matrice avec des boucles prendrait beaucoup trop de temps. Nous vous proposons de le faire au moyen des fonctions `pdist` et `squareform` de la librairie `scipy`. Expliquez comment cela est possible en étudiant la documentation de ces fonctions `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html` et `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.squareform.html#scipy.spatial.distance.squareform`. Nous attirons votre attention sur l'argument `metric` de la fonction `pdist` qui peut être soit une chaîne de caractère parmi celles proposées dans la documentation, soit une fonction que vous avez préalablement créé.

   Calculer cette matrice de distance pour les 100 premières villes (matrice 100x100) pour vérifier que tout se passe bien.
4. Mesurez le temps d'exécution du calcul des distances entre ville. Vous pouvez vous reporter au TP 1 et 2 pour mesurer le temps d'exécution d'un code python.
5. Que se passe-t-il quand on veut calculer cette matrice pour l'ensemble des villes de France ?

4 Matrice des distances euclidiennes entre villes

Afin de rendre les calculs plus rapides, nous allons désormais écrire une fonction qui calcule la distance euclidienne entre deux villes.

1. Nous allons dans un premier temps représenter les villes par leurs coordonnées 3D, et c'est à partir de ces coordonnées 3D que l'on pourra calculer les distances euclidiennes.

   Pour calculer les coordonnées 3D, vous pourrez utiliser les équations suivantes :

$$\begin{aligned} x &= R * \cos(p_{lat}) * \sin(p_{lon}) \\ y &= R * \cos(p_{lat}) * \cos(p_{lon}) \\ z &= R * \sin(p_{lat}); \end{aligned}$$

2. Calculez la distance entre Caen et Rouen avec cette nouvelle méthode. Comparez avec la valeur précédente et commentez.
3. En utilisant la fonction `plt.scatter` de `matplotlib`, affichez chaque ville sous la forme de point de coordonnée (x,-y).

5 Villes les plus proches

Écrire un programme qui recherche les 10 villes les plus proches d'une ville donnée.

Vérifiez en demandant les 10 villes les plus proches de Caen, au sens de la distance euclidienne.

Vous devriez trouver 'Caen', 'Hérouville-Saint-Clair', 'Louvigny', 'Mondeville', 'Saint-Germain-la-Blanche-Herbe', 'Epron', 'Colombelles', 'Fleury-sur-Orne', 'Giberville', 'Bretteville-sur-Odon'.

Vous ferez de même avec le calcul des 10 villes les plus éloignées d'une ville donnée.

6 Histogramme des distances des villes

Au moyen de la fonction `np.histogram` calculez l'histogramme des distances entre villes.

Affichez-le avec la librairie matplotlib.

7 Conclusion

1. Testez maintenant le TP avec toutes les villes de France.
2. Nous avons actuellement pas géré les doublons dans la base (par exemple une ville ayant plusieurs circonscription électorale ou des villes différentes ayant le même nom), en utilisant le code INSEE des villes faites en sorte de mieux gérer ces doublons. Si nécessaire, vous pouvez utiliser des boucles for.