TRAVAIL À RÉALISER POUR L'ÉTAPE 1 (AVANT LE 15 AVRIL)

A. Importer les données dans un dictionnaire **donneesbus** : la clé est le nom de l'arrêt et la valeur de la clé contiendra une liste avec la latitude, la longitude et la liste des arrêts voisins.

B. Créer une liste **noms_arrets** contenant les noms des arrêts

C. Créer les fonctions suivantes qui pourrons (ou non !) se révéler utiles pour les futures manipulations.
   - **nom(ind)** : permet d'associer le nom du sommet à son indice **ind** dans **noms_arrets**
   - **indice_som(nom_som)** : permet d'associer l'indice de l'arrêt à son nom **nom_som**
   - **latitude(nom_som)** : renvoie la latitude d'un arrêt nommé **nom_som**
   - **longitude(nom_som)** : renvoie la longitude d'un arrêt nommé **nom_som**
   - **voisin(nom_som) :** renvoie la liste des voisins d'un arrêt nommé **nom_som**

D. Le réseau de bus peut, bien sûr, être modélisé par un graphe.
   Représenter ce graphe sous Python :
   - La liste d'adjacence par un dictionnaire ***dic_bus***
   - La matrice d'adjacence par une liste de liste ***mat_bus***

E. Distance.
   Créer deux fonctions :
   - ***distarrets(arret1,arret2)*** renvoyant la distance à vol d'oiseau entre les arrêts *arret1* et *arret2.*
   - ***distarc(arret1,arret2)*** renvoyant la distance à vol d'oiseau entre les les arrêts *arret1* et *arret2* quand l'arc ***(arret1,arret2)*** existe, et sinon la valeur qui vous paraît la plus cohérente.

   UNE FONCTION UTILE : DISTANCEGPS

```
from math import sin, cos, acos, pi
import numpy as np
##########################################################
# calcul de la distance entre deux points A et B dont   #
# on connait la lattitude et la longitude               #
##########################################################
def distanceGPS(latA,latB,longA,longB):
    # Conversions des latitudes en radians
    ltA=latA/180*pi
    ltB=latB/180*pi
    loA=longA/180*pi
    loB=longB/180*pi
    # Rayon de la terre en mètres (sphère IAG-GRS80)
    RT = 6378137
    # angle en radians entre les 2 points
    S = acos(round(sin(ltA)*sin(ltB) + cos(ltA)*cos(ltB)*cos(abs(loB-loA)),14))
    # distance entre les 2 points, comptée sur un arc de grand cercle
    return S*RT
```

F. Le réseau de bus peut aussi être modélisé par un graphe ***pondéré***, qui pourra être utilisé par les algorithmes de plus courts chemins.
   Représenter ce graphe sous Python par une matrice des poids (liste de listes), ***poids_bus*** prenant en compte à la fois l'existence des arcs et leur poids.
   Sachant que notre objectif est de chercher les chemins de poids minimum, on affectera aux arcs qui n'existent pas dans le graphe une valeur adaptée à cette future recherche…