LOCALISATION, CARTOGRAPHIE ET MOBILITE

I- Généralités sur la cartographie.

1) Représentation des distances et du relief.

Les **cartes sont des représentations réduites de données réelles.**

Elles sont soumises à une **échelle** qui fait état de la réduction effectuée.

Exemple : les cartes IGN à 1:25000 sont des représentions qui prennent pour échelle 1cm sur la carte pou 25000cm dans la réalité, soit 250m.

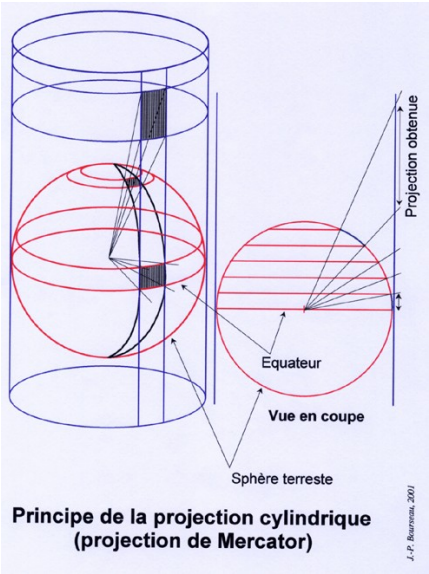
Le relief peut être représenté grâce aux **courbes de niveaux.**

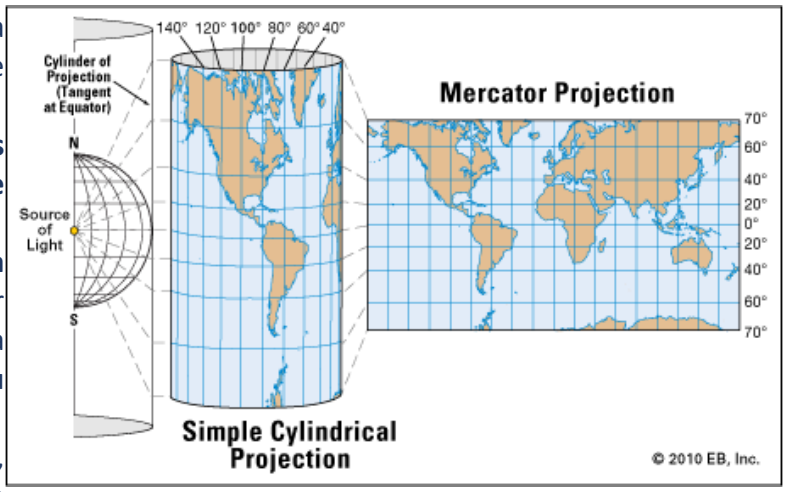
Cf :

https://www.geoportail.gouv.fr/donnees/carte-ign

2) Projections cartographiques.

Exemple de la projection de Mercator



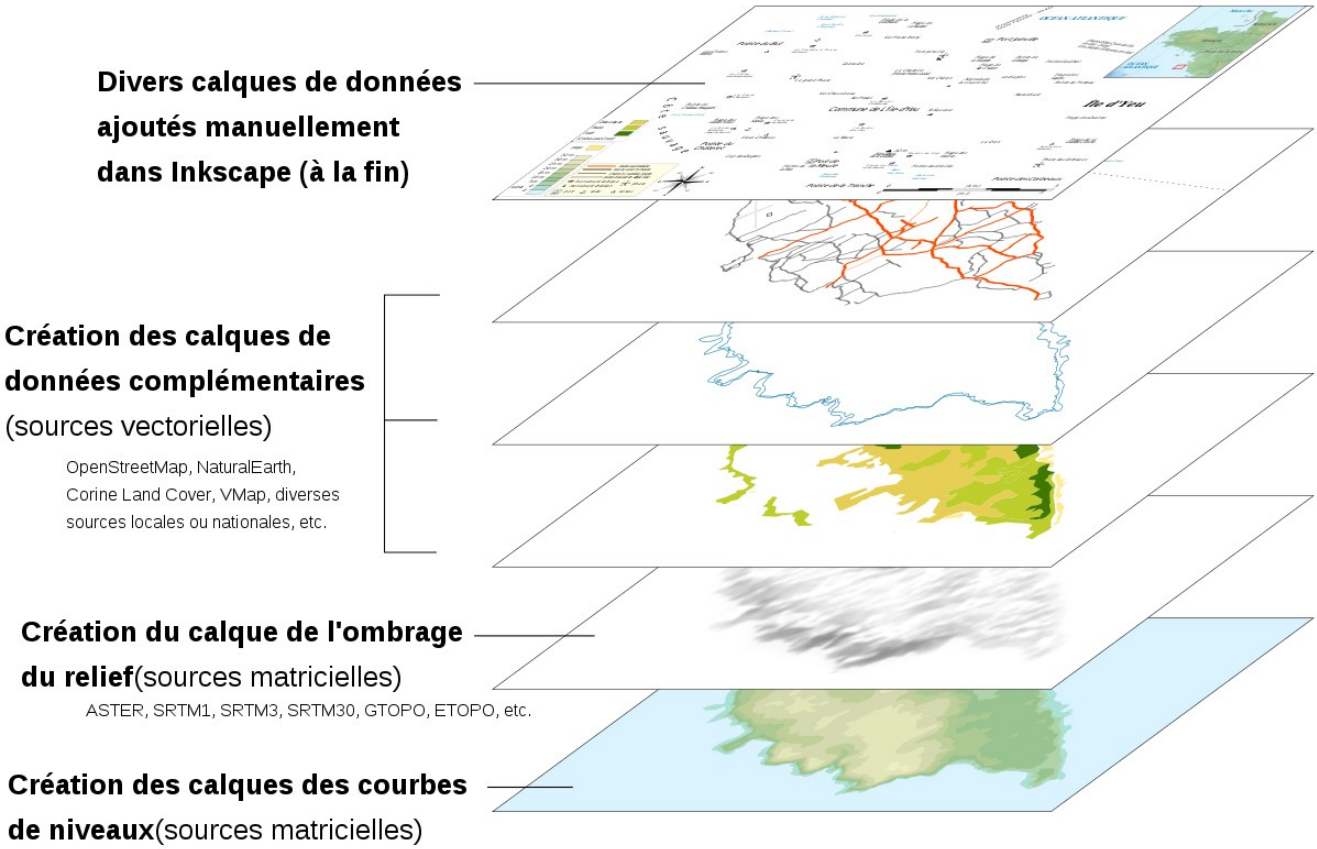


Avec la "Projection de Mercator" (1569), **les distances ne correspondent aucunement à la réalité** (par exemple la taille du Groenland, très au Nord, dépasse celle de l'Amérique du Sud sur l'équateur alors que cette dernière est 8 fois lus étendue.). Cependant, les rapports angulaires entre les lieux restent exacts (conformes). Alors que les architectes et les géomètres préfèrent des cartes « équidistantes » (1 cm sur la carte égale x cm en réalité), les navigateurs préfèrent celle de Mercator.

3) Les cartes numériques

Les cartes numériques sont des assemblages de calques fournis par divers « producteurs » mais basés sur la même matrice. Ils sont donc associables à souhaits et apportent diverses informations selon nos intérêts.

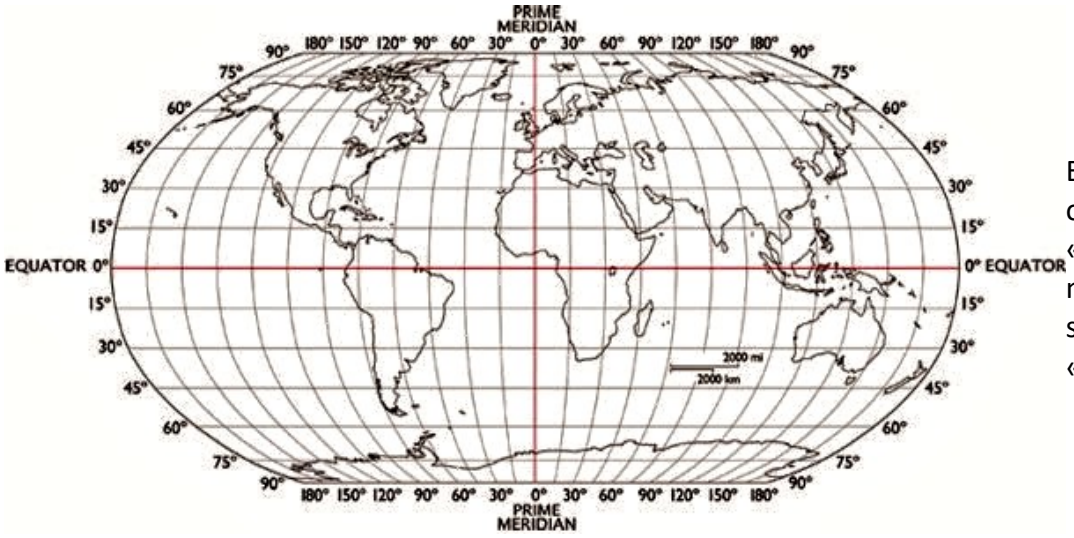
système de calques.cf géoportail. **https://www.geoportail.gouv.fr/**



II- Géolocalisation.

1) Coordonnées géographiques

On divise le globe en lignes parallèle au méridien de Greenwich reliant les pôles et à l'équateur.
La façon la plus usitée pour exprimer une position est selon la forme D.M.S.(degré, minute, seconde)



Entre chaque d'un degré, on peut subdiviser en 60 « minutes », chaque minute peut elle être subdiviser en 60 « secondes ».

La tour Eiffel est donc situé à : 48°51'29''N 2°17'38''E

°pour degré, ' pour minute et ''pour seconde. Les références sont le méridien de Greenwich pour la longitude(la T.E. se trouve donc à 'Est de ce méridien) et l'équateur pour les latitudes(la T.E. se trouve au Nord de cette ligne.).

Il existe aussi une façon décimale(D.D. : Degré décimal) de représenter les coordonnées :

La tour Eiffel est donc situé à : 48,858053N 2,2944991E

Exo :
1- Créer une fonction permettant de transformer les coordonnées DD en DMS et vis versa… : ddtodms et dmstodd
2- Créer une fonction permettant de donner un distance à vol d'oiseau entre des points de surface du globe : distance_vol_oiseau … :

formule :
d=*ACOS(SIN(**latA**)*SIN(**latB**)+COS(**latA**)*COS(**latB**)*COS(**longB-longA**))*R*.

Avec les latitudes et longitudes en radians ; conversion :
angle en radians = (angle en degrés/180)*Pi
R= rayon moyen de la Terre à l'équateur= 6378137m
pensez à importer les modules nécessaires : from math import sin, cos, acos

Corrections :
1-

```
def ddtodms(dd) :
    d=int(dd)
    x=(dd-d)*60
```

```
        m=int(x)
        s=(x-m)*60
        return d,m,s
def dmstodd(d,m,s) :
        dd=d+m/60+s/3600
        return dd
```

2-

```
from math import sin, cos, acos

def  distance_vol_oiseau(latA, longA, latB, longB) :
        R= 6378137
        latA=latA/(180*Pi)
        longA=longA/(180*Pi)
        latB=latB/(180*Pi)
        longB=longB/(180*Pi)
        LongArc=acos(sin(latA)*sin(latB) + cos(latA)*cos(latB)*cos(abs(longB-longA)))
        return LongArc*R
```

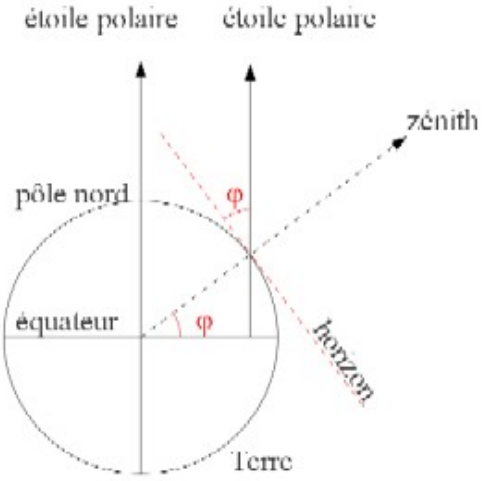
2) géolocalisation.

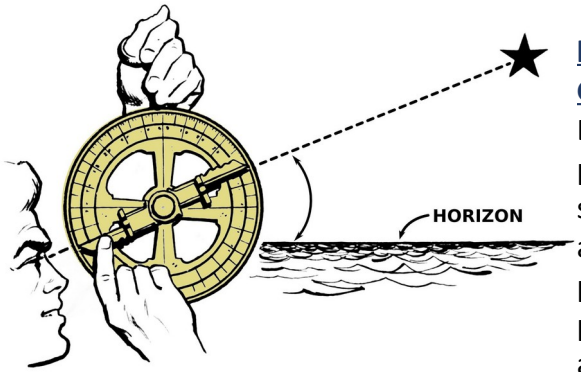
a- Principe et technique historiques :

Technique historique :

Grâce à un astrolab :

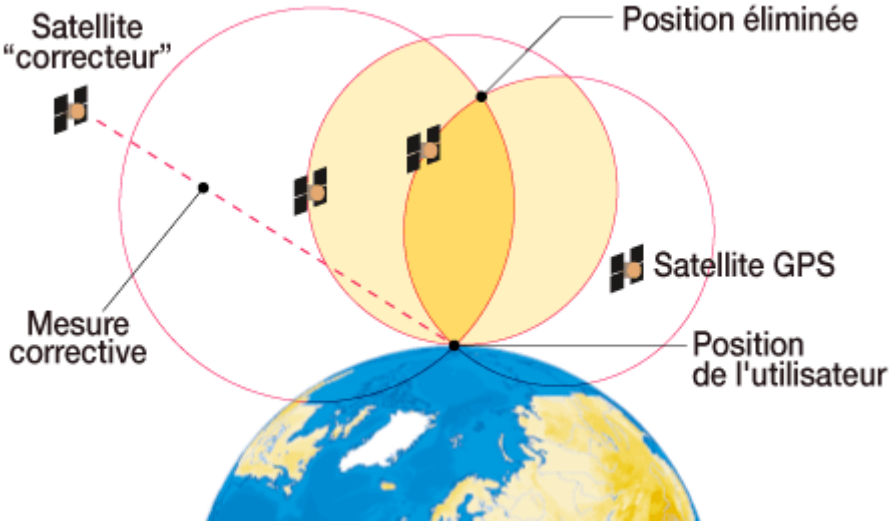
On peut trouver sa latitude en observant à midi la hauteur maximale du Soleil, altitude spécifique à chaque jour de l'année pour une latitude donnée. En bref, si l'on connaît la date, on peut connaître sa latitude en consultant un almanach. Ce qui vaut pour le Soleil vaut tout autant pour d'autres astres et étoiles dont on peut mesurer l'altitude. Pour mesurer la latitude, on compte à partir de l'équateur 90° jusqu'au pôle Nord, et 90° jusqu'au pôle Sud.





b- Principe actuelle : les systèmes GPS et Galiléo

Les 24 satellites nécessaires au positionnement GPS des objets connectés sont en orbite autour de la Terre, à une altitude de 25 000 km environ. A chaque point du globe, 4 satellites sont visibles en permanence : c'est le nombre minimum pour assurer un positionnement précis. Le GPS a une précision de 10 à 15 m seulement, car les ondes sont ralenties par l'atmosphère et parfois réfléchies sur les immeubles.



1. Les satellites de la constellation, équipés d'une horloge atomique d'une extrême précision, émettent des signaux indiquant l'heure de départ du satellite.
2. Le récepteur au sol, intégré par exemple dans un navigateur, possède en mémoire **les coordonnées précises des orbites de tous les satellites**. Il reçoit le signal d'un satellite et chronomètre le temps qu'il a mis pour arriver. Multiplié par la vitesse du signal (celle de la lumière), il obtient sa position par rapport à lui.
3. En croisant les données de trois satellites, il ne reste que deux possibilités. Le récepteur élimine celle qui n'est pas sur Terre.
4. Mais si les satellites sont équipés d'horloges atomiques, ce n'est pas le cas des récepteurs standards. Or un décalage d'une seconde peut entraîner une erreur de 300 000 km ! Pour confirmer sa position, il faut **un quatrième satellite qui affine les mesures précédentes.**

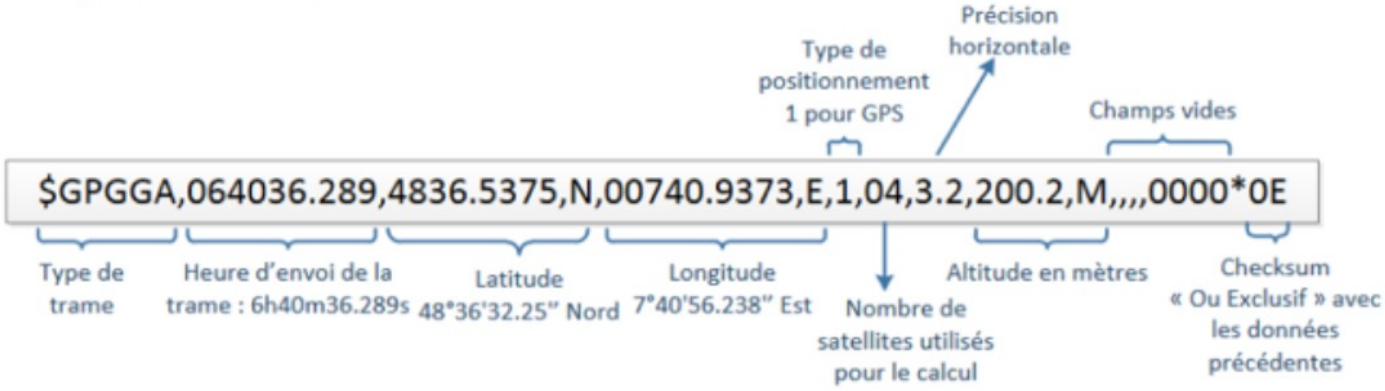
Pour des points mobiles, on a compris, une évaluation à partir d'une suite de positions fixes n'est en effet pas satisfaisante. Grâce à l'effet Doppler ; si l'utilisateur s'éloigne du satellite, la longueur d'onde du signal émis par celui-ci augmente. Si il s'en rapproche, elle diminue.

Le système Galileo comprendra 30 satellites (6 de plus que le GPS), placés sur 3 orbites circulaires. Il offrira 10 bandes de fréquence, contre une seule actuellement pour le GPS. Sa résolution sera meilleure (1 m environ).

3) Le protocole NEMEA0183 :

Les **récepteurs GPS** reçoivent les **informations** sous forme **normalisée** et facilement exploitable. Le protocole NEMEA fournit ainsi des **trames :**

Exemple de trame GGA :



Une autre trame très courante pour les bateaux est la RMC, qui donne l'heure, la latitude, la longitude, la date, ainsi que la vitesse et la route sur le fond mais pas l'altitude. Par exemple :

$GPRMC,053740.000,A,2503.6319,N,12136.0099,E,2.69,79.65,100106,,,A*53

type de trame

heure de réception

coordonnées

vitesse en nœuds.

Exercice :

1- Donner les coordonnées et l'altitude associées à la trame :

$GPGGA,174553.258,423600.8900,N,084821.8700,E,1,04,3.1,0.0,M,,,,0000*0E

2- Construire une fonction python NEMEA_DD qui donne les coordonnées en format DD si l'on rentre une trame $GPGGA. Facultatif, c'est chiant !!!

Corrections :

1-

42°36'00,89''N 8°48'21,87E 0m d'altitude

2-

```
def NEMEA_DD(trame) :
    champsTrame=trame.split(',')
    lat=float(champsTrame[2])
    deg=lat/100
    min=lat-deg*100
    lat=lat+min
    Orlat=champsTrame[3]
    long=float(champsTrame[4])
    deg=long/100
    min=long-deg*100
    long=long+min
    Orlong=champsTrame[5]
```

return lat, orLat, long, OrLong

III- travail numérique, évaluation simple des distances/ des trajets.

Le package folium de python : créateur de cartes.

Exercice 3 : Création de cartes et de trajets, choix des trajets les plus courts.

1- Créer une carte grâce au module folium : la zone importe peu, je veux 4 points sur la carte reliés par des lignes.

Indispensable :

→ installation de ***folium***.

Pip **install** folium

→ début de programme.

Import folium as fol

Attribution de la carte à une variable(c'est un tableau, une image matricielle!).

Code :

Macarte=**fol.Map**(**location**=[lat,long], **zoom_start**=10)

location : centre la carte, coordonnées en DD. → indispensable

zoom_start : permet d'afficher avec un zoom→ facultatif

tiles : indique une des 200 cartes de fond appelée, par défaut, openstreetmap→ facultatif

divers instructions du module folium :

variable**.save**('nomde fichier.html') → crée un fichier de type html qui affiche la carte.

Exemple : Macarte.**save**('MaCarte.html')

fol.Marker([lat,long], popup='nomdupoint')add_to(Macarte) → place des icônes à des endroits précis dur la carte nommée *Macarte* et affiche *nomdupoint* quand on clique dessus.

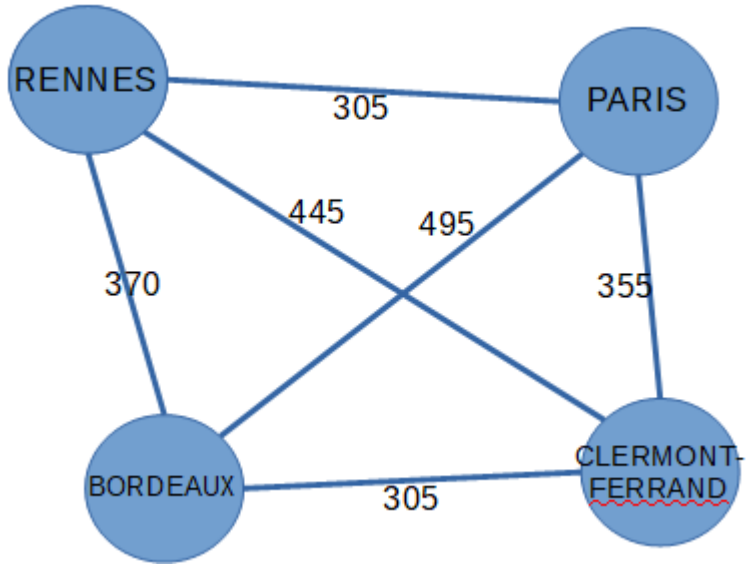
Exemple :

folium.Marker([49.1,0.0], **popup=**"trouPaumé",**icon=folium.Icon**(color='red'))**.add_to**(Macarte)

folium.PolyLine([(49.1,0),(48.5,0),(48.5,1)],**color=**"blue",**weight=**2.5, **opacity=**0.8)**.add_to**(carte) → on trace un trajet bleu de largeur 2,5px et d'opacité 0,8 qui passe par les 3 points aux coordonnées notées.

2- Réaliser le graphe non orienté pondéré des trajets à vol d'oiseau entre ces 4 points. (utilisez la formule ou le programme réalisé en exercice 1)

Rq : On peut représenter un « graphe » qui représente la disposition des points sur votre carte. En admettant qu'on peut aller de chaque point vers tous les points, ce graphe donnerait cela :



On trouve sur les lignes de liaison les valeurs de chaque liaison(ici des distances). C'est un graphe non orienté(il n'y a pas de sens obligatoire sur chaque ligne) avec des arrêtes de poids différents.

Cette représentation peut aider à la résolution de problème comme le chemin le plus court entre deux points ou celui nécessaire pour passer par tous les points une seule fois en un minimum de distance.

Base pour Exemple :

```
import folium
import os
adressefichierfinal = input("Veuillez indiquer l'adresse de la photo finale")
os.chdir(adressefichierfinal) # détermine le dossier de travail

carte=folium.Map([49,0.0], zoom_start=9) # détermine le centre de la carte avec un zoom donné
folium.Marker([49.1, 0.0], popup="trouPaumé",icon=folium.Icon(color='red')).add_to(carte) # un repère placé, rouge avec un nom
Points=[(49.1,0),(48.5,0),(48.5,1),(49.1,0),(50.1,0.9),(48.1,-1),(49.1,0),] #on crée des points pour tracer un trajet
folium.PolyLine(Points, color="blue", weight=2.5, opacity=0.8).add_to(carte)#on trace un trajet bleu...
carte.save('MaCarte01.html') # enregistre la carte sous fichier format html
```

Projet :

Situation :

Jean-Louis a 15 ans et c'est déjà dur de s'appeler Jean-Louis aujourd'hui !
Il à envie de visiter Paris avec Mireille, sa grand-mère de 87ans(c'est bizarre mais bon!). Elle a du mal à marcher mais c'est la grève des transports(métro, bus, taxi, Uber…), , c'est vraiment la poisse !
En plus, elle veut absolument voir 6 monuments avant de repartir par là où elle arrive demain matin : La gare du Nord. Le monuments sont les suivants :

ARC_TRIOMPHE = [48.873804 , 2.29512]
EIFFEL = [48.858053 , 2.2944991]
JARDIN_DES_PLANTES = [48.843971 , 2.359786]
LOUVRE = [48.860572 , 2.337657]
ORSAY =[48.86004 , 2.326533]
QUAI_BRANLY =[48.860903 , 2.297582]

Et pour la gare du nord :
GARE_NORD = [48.882505 , 2.354784]

Ressources :

- Vous pouvez utiliser les variables monuments telles qu'elles sont présentées ici, ça vous facilitera la tâche
- Vous pouvez créer une liste avec vos variables :

lieux = [ARC_TRIOMPHE, EIFFEL, GARE_NORD, JARDIN_DES_PLANTES, LOUVRE, ORSAY , QUAI_BRANLY]

Rappel : dans la liste 'lieux', lieux[2] correspond aux coordonnées de la gare du nord, lieux [3][1] correspond à la longitude du jardin des plantes.

Niveau 1 :

Créer la carte de Paris en format html qui permet de visualiser les 6 endroits du parcours parisien, attribuer un marqueur à chaque endroit avec le nom de l'endroit associé en popup et afficher le trajet en couleur rouge.

Niveau 2 :

Niveau 1 + Utiliser vos fonctions pour évaluer la distance total du parcours.

Niveau 3 :
Trouver le trajet le plus court adapté aux vieilles jambes de Mireille en tâtonnant mais e utilisant votre joli programme.

Niveau Ultime :
Créer un programme qui trouve le chemin à vol d'oiseau le plus court pour parcourir autant de points que l'on veut à Paris.
Le programme doit demander le nom des monuments et les deux coordonnées à l'utilisateur pour chaque monument.