LOCALISATION – Trottinettes en libre-service

Activité

CAPACITES ABORDEES

- Décrire le fonctionnement de la géolocalisation.
- Contribuer à OpenStreetMap de façon collaborative.
- Décoder une trame NMEA.
- Utiliser un logiciel pour calculer un itinéraire.

MATERIELS NECESSAIRE

- Un ordinateur avec Python 3.x et la bibliothèque *folium* installée.
- Environnement de travail Pyzo (ou autre).

Cette activité permet d'aborder différentes compétences associées au thème de la localisation du programme de SNT.

Je tiens à remercier mon collègue Mohamed Hachem, professeur de technologie au collège Janson de Sailly, pour son implication et sa participation à la production dans cet article.

Figure 1 : Trottinettes Bird

Contenu

1 INTRODUCTION

Actuellement, nous retrouvons beaucoup de trottinettes en libre-service dans les grandes villes françaises. Il est très facile de trouver une trottinette à partir de son smartphone via les applications dédiées. Cette activité vise à mettre en évidence le moyen de géolocalisation de ces trottinettes dans un premier temps puis nous verrons comment il est possible à partir de la trame NMEA générée par la balise GPS de récupérer et de tracer sur une carte un chemin parcouru par un utilisateur.

- Comment sont-elles géolocalisées ?
- Comment elles apparaissent sur une carte lorsqu'elles sont disponibles ?
- Comment récupérer l'information sur leur localisation ?
- Comment est calculé le trajet entre vous et la trottinette ou le trajet que vous avez effectué ?

Dans cet article, nous nous intéresserons dans un premier temps au fonctionnement de la géolocalisation et comment il est possible de déterminer les coordonnées d'un point en fonction du système de projection choisi ainsi que de calculer de la distance entre deux points à partir de ce système de projection.

2 PRINCIPE DE FONCTIONNEMENT DE LA GEOLOCALISATION

2.1 CONSTITUTION D'UN TRACEUR

Les objets connectés et en particuliers les trottinettes électriques sont généralement équipées d'émetteur GPS (traceur, tracker ou balise). Avant de voir comment fonctionne un traceur GPS, commençons par détailler de quoi il est constitué.

Il est composé :

- d'un **module** GPS (Figure 2).

Figure 2 : Module NEO-6M GPS

- d'un **module de téléphonie mobile GSM GPRS** (Figure 3), qui lui permet d'envoyer ses informations de position soit par SMS, soit par internet. Le traceur se connecte à Internet pour y envoyer ses informations vers un **serveur internet**, identifié par une adresse **IP**.

Figure 3 : Module SIM900 GSM GPRS

Donc les positions recueillies par le traceur GPS partent via internet sur le site de géolocalisation de Lime au minimum toutes les minutes.

2.2 HISTORIQUE DU GPS

Le GPS « Global Positioning System » est un système de positionnement par satellites créé par les Américains pendant la Guerre Froide. Opérationnel depuis 1995, sa précision pour les civils était volontairement dégradée autour d'une centaine de mètres.

Ce n'est qu'en mai 2000 que le président Clinton a décidé de laisser libre accès à la précision maximale qui est de 5 à 10 mètres, et a du coup ouvert la voie à la commercialisation de nombreux types de récepteurs GPS pour des usages les plus variés dont nos traceurs GPS. L'Europe (Galileo), la Chine, la Russie et l'Inde, conscients de l'intérêt stratégique d'un tel système de positionnement par satellites développent depuis quelques années des systèmes concurrents.

2.3 LES COORDONNEES GEOGRAPHIQUES ET LEUR SYSTEME DE PROJECTION

Lorsque l'on doit localiser un objet sur la Terre, il est nécessaire dans un premier temps de représenter la Terre sur une surface plane. On se confronte alors au problème de représentation d'une sphère sur un plan ce que l'on appelle aussi une **projection**. On peut aussi parler plus généralement de représenter la surface d'une géométrie non euclidienne dans une géométrie euclidienne. On peut se retrouver confronter à ce problème lorsque l'on essaye d'étaler la peau d'un fruit à géométrie sphérique (pomme, orange, etc.) sur une surface plane (par exemple une table). Nous sommes alors obligés de déformer, voire de déchirer, la peau pour pouvoir l'étaler.

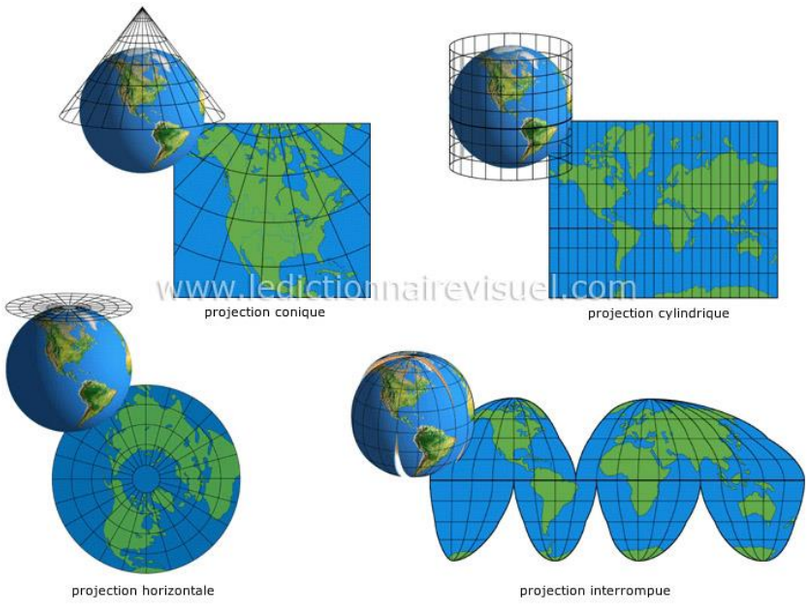


Figure 4 : Projection de la Terre sur une surface plane

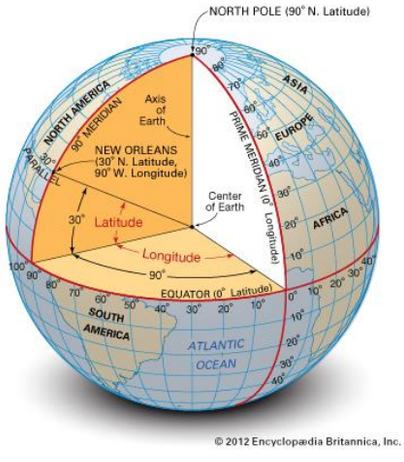
C'est exactement le même problème avec la Terre comme le montre la Figure 4 : Projection de la Terre sur une surface planeFigure 4. Cette figure présente différent type de projection (conique, cylindrique, horizontale ou interrompue). Il est donc nécessaire de définir un système de projection (noté Coordinate Reference System, CRS) commune pour ensuite utiliser des coordonnées de l'objet à repérer dans ce système. Dans ce cas, une coordonnée géographique 2D est caractérisée par une paire de coordonnées, et le système de projection de cette paire de coordonnées. Une paire de coordonnées sans son CRS n'est pas une coordonnée géographique. Ce système s'étend aux coordonnées géographiques 3D, il faut alors un triplet de coordonnées (la troisième étant une « altitude ») et un CRS.

Pour unifier les représentations des CRS, nous utilisons le code « EPSG » montre le Tableau 1.

Par exemple, pour le CRS **WGS84** qui est une projection très utilisée et en particulier par le GPS, le code EPSG correspondant est 4979 si l'on considère aussi l'altitude dans les coordonnées (donc une représentation en 3D) et 4326 si l'on ne considère que la longitude et la latitude (donc une représentation en 2D). Lorsqu'on manipule des données cartographiques, il faut s'attendre à devoir jongler entre des coordonnées exprimées dans différents CRS. Il en existe de très nombreux. En particulier, en France, il est très fréquent de rencontrer 3 systèmes : le WGS 84, le Lambert 93 et le Lambert 2 étendu.

Tableau 1 : Code ESPG

Code	Nom	EPSG	Remarques
RGF93	Réseau Géodésique Français 1993	6171 (système géocentrique), 4965 (3D), 4171 (2D)	Système français légal (décret 2000-1276 du 26 décembre 2000). Identique à l'ETRS89 au 1/1/1993. Compatible avec le WGS84 pour des précisions égales ou supérieures à 10 m (c'est-à-dire 15 m etc.).
NTF	Nouvelle Triangulation Française	2D : 4807 (Paris, grade) ou 4275 (Greenwich, degré). 3D : 7400 (Paris, grade)	Système français périmé mais encore largement utilisé.
ETRS89	European Terrestrial Reference System 1989	4937 (3D), 4258(2D)	Système européen actuel
ED50	European Datum 1950	4230	Système européen périmé
WGS84	World Geodetic System 1984	4979 (3D), 4326 (2D)	Système mondial très utilisé notamment avec le GPS.



Nous allons nous intéresser plus particulièrement au CRS **WGS 84** qui est basé sur la projection de Mercator et qui est le système le plus standard, parce qu'il est lié au GPS. Autrement dit, lorsqu'un téléphone (par exemple) se géolocalise par GPS, il renvoie au système une paire de coordonnées longitude/latitude exprimées dans le système WGS 84.

En WGS 84, on a donc une paire de coordonnées constituée d'une longitude et d'une latitude. La longitude 0 correspond au méridien de Greenwich, et varie de -180 (vers l'Ouest) à +180 (vers l'Est). La latitude 0 correspond à l'équateur (la ligne, pas le pays) et varie de -90 (vers le Sud) à +90 (vers le Nord).

On remarque très vite que votre **latitude** est l'angle entre le segment [centre de la Terre, point le plus proche de vous sur l'équateur] et le segment [centre de la Terre, votre position] tandis que votre longitude est l'angle entre le segment [centre de la Terre, point le plus proche de vous sur l'équateur] et le segment [centre de la Terre, point le plus proche de Greenwich sur l'équateur].

2.4 CALCUL DE DISTANCES

Le calcul de distance entre deux points dépend du système de coordonnées utilisé. Prenons l'exemple de deux systèmes de coordonnées couramment utilisés : le système WGS 84 (utilisé dans le GPS) et le système Lambert 93. Vous pouvez utiliser ce site https://projection.dogeo.fr/point-to-coords pour réaliser la conversion entre deux systèmes de projection.

2.4.1 Utilisation du système WGS 84

En utilisant la projection WGS 84, on peut obtenir la distance entre deux points ce que l'on appelle (« vol d'oiseau » ou distance géodésique) en considérant la Terre comme une sphère parfaite à l'aide de la formule suivante :

$$distance = r * \Delta\sigma$$

$$\Delta\sigma = \arccos(sin\phi_1 * sin\phi_2 + cos\phi_1 * cos\phi_2 * \cos(\Delta\lambda))$$

$$\Delta\lambda = |\lambda_1 - \lambda_2|$$

Les angles ϕ_1 et ϕ_2 correspondent aux latitudes, $\lambda_1 et\ \lambda_2$ correspondent aux longitudes sont exprimés en radians et r est le rayon de la Terre à position considéré.

On remarque dans ce cas que l'expression de la distance est plus complexe que dans un repère orthonormé. De plus de faible variation de coordonnées longitudinales ou de latitudes entraine de forte variation de positions.

2.4.2 Utilisation du système de Lambert 93

Pour calculer des distances entre deux points, nous pouvons alors utiliser un autre système de projection appelé système de Lambert, valable uniquement pour des coordonnées situées à l'intérieure du cadre rouge défini sur la Figure 5. Le système Lambert 93 (EPSG 2154), correspond à un découpage de cette zone rouge en carreaux, considérés comme pouvant être aplatis plus précis mais limité à la région à l'intérieur de ce rectangle.

Figure 5 : Zone d'utilisation du système de Lambert 93

En Lambert 93, on ne parle plus de longitude et latitude (puisqu'il n'est plus ici question d'angle à l'équateur ou au méridien de Greenwich), mais de coordonnées X et Y (vous trouverez aussi les appellations Easting pour X et Northing pour Y). Ce X et ce Y représentent une distance horizontale et verticale (respectivement) en mètres à un point 0, 0 (situé quelque part au large du Gabon, pas loin du point 0, 0 en WGS 84, mais pas exactement… déformation oblige !).

En connaissant les coordonnées de deux points dans le système de Lambert, nous pouvons déterminer la distance entre ces points en utilisant la formule de Pythagore :

$$distance = \sqrt{(\Delta X^2 + \Delta Y^2)}$$

Remarque

Ces différents calculs de distance entraînent nécessairement une certaine imprécision : par exemple, la distance euclidienne donne 0,681 km de moins que la distance ce Vincenty pour la distance Brest-Nice, qui est d'environ 1047km (soit 0.065% d'erreur).

3 ACTIVITES AUTOUR DU FONCTIONNEMENT DU GPS

Nous allons maintenant nous intéresser plus précisément à la localisation d'un objet à partir du système GPS.

ACTIVITE

Après avoir visionné les vidéos suivantes vous allez répondre aux questions :

Vidéo 1 : https://www.youtube.com/watch?v=WoqpQbWdacQ#action=share

Vidéo 2 : https://www.youtube.com/watch?v=e79tSIpLiDk#action=share

EXERCICE

Combien faut-il de satellites au minimum pour avoir sa position

Quel principe est utilisé par le GPS ?

Quelle est la précision du GPS Galliléo ?

Combien y a-t-il de satellites dans le système Galliléo ?

Combien de personnes peuvent être connectés en même temps sur le système Galliléo ?

EXERCICE

On cherche à retrouver la position d'un point O dont les distances aux trois satellites représentés par les points A, B et C sont connues. On donne : OA = 9,5 cm, OB = 8,6 cm et OC = 7,3 cm (sachant qu'en réalité la distance AB est de 7,1).

A.

B.

C.

4 AFFICHAGE DE LA POSITION DES TROTTINETTES ELECTRIQUES EN TEMPS REEL

Nous allons essayer de comprendre comment les trottinettes s'affichent en temps réel sur une carte (Figure 6) lorsqu'elles sont disponibles à la location.

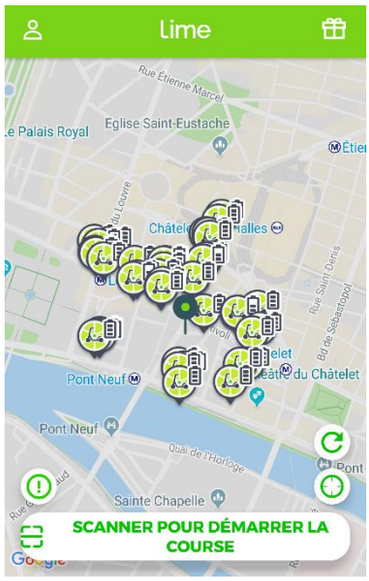


Figure 6: Disponibilité des trottinettes

Pour cela nous allons utiliser OpenStreetMap qui est une carte ouverte (participative) afin d'y ajouter des points d'intérêts. Dans un premier temps, nous allons utiliser le site https://www.openstreetmap.org/ pour déterminer les coordonnées GPS d'un lieu que nous allons ensuite utiliser pour simuler l'ajout de points d'intérêts sur une carte OpenStreetMap à partir du langage Python et de la bibliothèque folium.

4.1 OPEN STREET MAP

OpenStreetMap est un projet de cartographie qui a pour but de constituer une base de données géographiques libre du monde, en utilisant le système GPS et d'autres données libres. Il a été mis en route en juillet 2004 par Steve Coast au University College de Londres.

ACTIVITE

Vous trouverez ci-dessous un lien vers une vidéo décrivant l'utilisation d'OSM.

https://www.youtube.com/watch?v=zJSGOpqa9ew&feature=youtu.be

Comme le montre la Figure 7, OpenStreetMap utilise une base de données géographique afin que ces dernières soient utilisées pour faire du calcul d'itinéraires, du fond de carte, du géocodage ou de la localisation de points d'intérêts qui soient fixes ou mobiles.

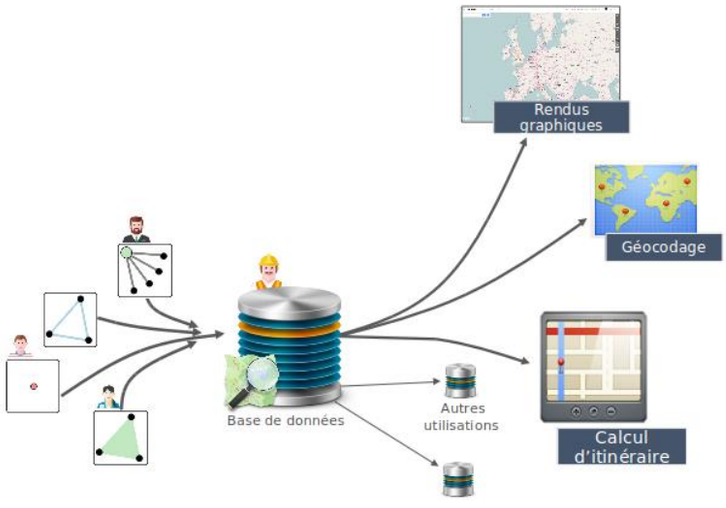
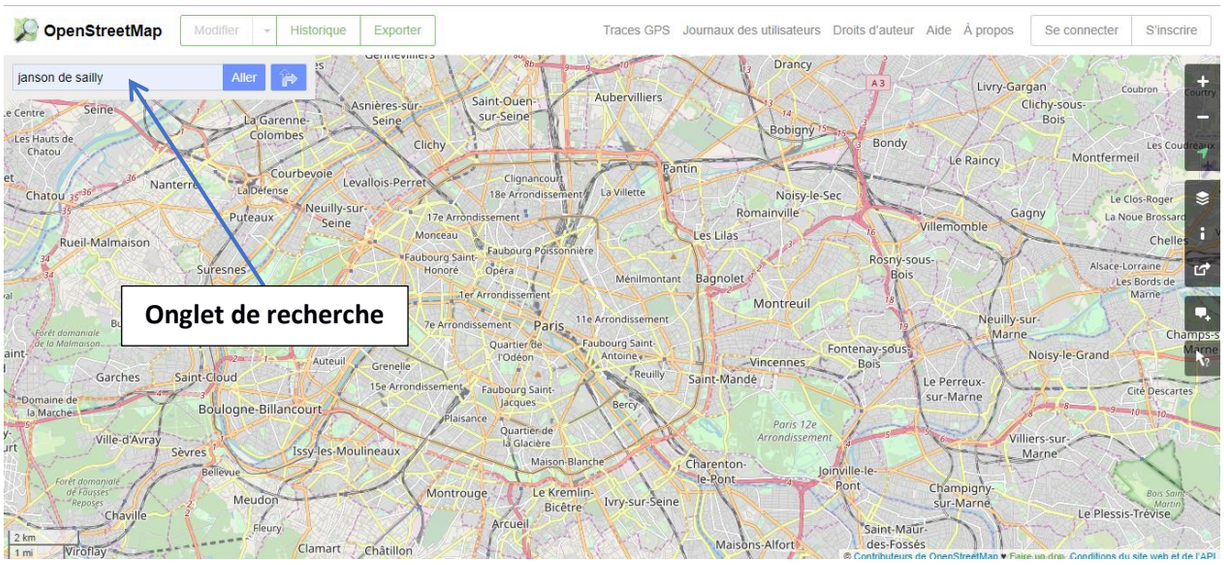


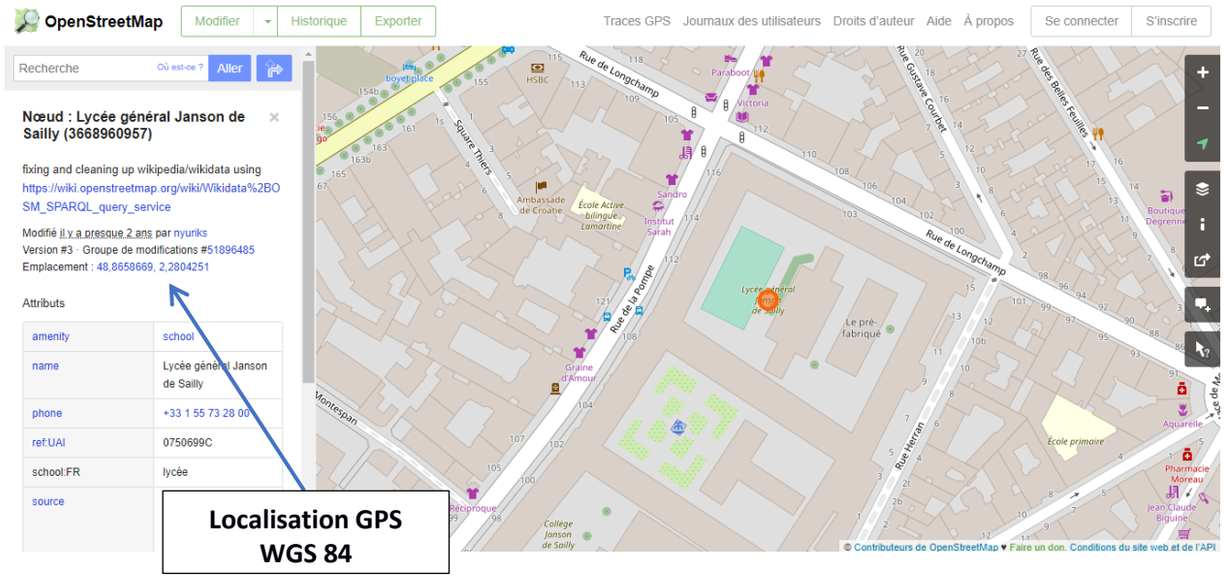
Figure 7 : Crédits de l'image ci-dessus : François Lacombe – Barrymieny – Webalys

ACTIVITE

A partir du site https://www.openstreetmap.org/ nous allons déterminer les coordonnées GPS d'un lieu.

Une fois la page internet chargée, vous pouvez rechercher un lieu à partir de l'onglet de recherche (par exemple ici nous recherchons la localisation du lycée Janson de Sailly à Paris).





Maintenant que nous avons obtenu la localisation souhaitée [48.8658669, 2.2804251], nous allons l'utiliser sous Python pour ajouter des points d'intérêts à notre carte OSM autour de ce point GPS.

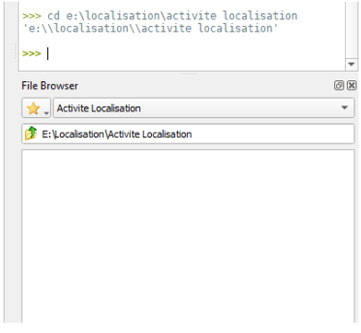
4.2 INSTALLATION DE LA BIBLIOTHEQUE FOLIUM SOUS PYTHON

Dans un premier il est nécessaire de créer un dossier dans lequel seront sauvegardées les activités suivantes et en particuliers les cartes personnalisées qui seront générées.

Créez un dossier et nommez-le par exemple "Activite Localisation ".
Il faut maintenant indiquer sous Pyzo que vous voulez travailler dans ce dossier.
Pour cela vous pouvez entrer la commande suivante :

```
cd e:\localisation\activite localisation
```

Cette commande permet de changer le dossier courant de travail (cd pour Current Directory).



Nous allons installer les bilbliothèque folium et Image si ces dernières n'ont pas encore été installées. L'installation de bibliothèques se fait par la commande :

```
pip install folium
pip install Image
```

4.3 AJOUT D'UN POINT D'INTERET SUR UNE CARTE AVEC OPENSTREETMAP

Une fois la bibliothèque installée et le dossier créé nous allons pouvoir ouvrir une carte OSM à partir de Python.

Utilisation du fichier **lycee.py**

ACTIVITE

```
import folium
map = folium.Map(location = [48.8658669, 2.2804251], zoom_start=20)
map.save('maCarte1.html')
```

Ces commandes permettent d'importer la bibliothèque folium puis de créer une carte à partir d'OSM centrée sur le point GPS de coordonnées [48.8658669, 2.2804251] (pour rappel il s'agit des coordonnées GPS que nous avions déterminée pour le lycée Janson de Sailly. Plus généralement nous avons "folium.Map(location=[latitude, longitude])". Il suffit donc de renseigner la bonne longitude et la bonne latitude pour que la carte soit centrée sur le point désiré). Le paramètre zoom_start permet de réaliser un zoom plus ou moins important. Plus la valeur est importante plus la zoom sera important. La dernière instruction permet de sauvegarder l'objet map sous forme de page html appelée ici *maCarte1.*

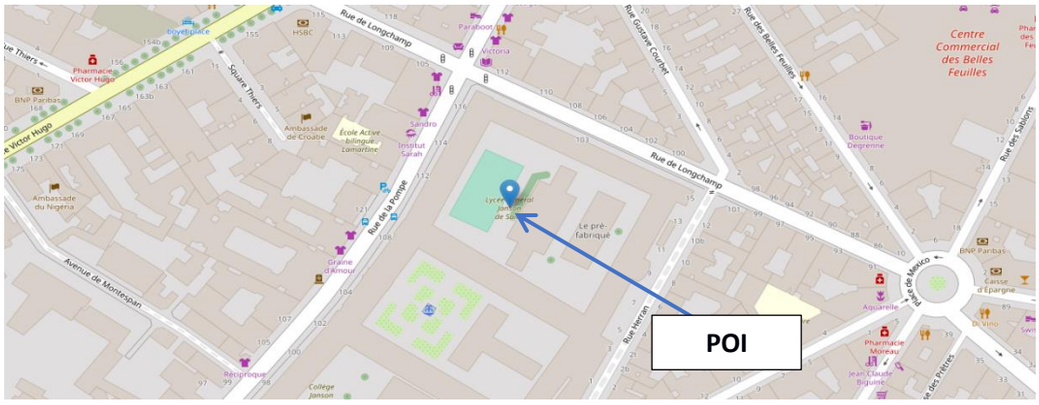
Une fois le code ci-dessus exécuté, rendez-vous dans le répertoire que vous avez créé. Vous devriez trouver un fichier "maCarte1.html". Double-cliquez sur ce fichier, cela devrait normalement ouvrir un navigateur web : la carte centrée sur les coordonnées GPS choisies est à votre disposition. Notez bien que nous avons une véritable carte et pas une simple image (il est possible de zoomer ou de se déplacer).

ACTIVITE

On va maintenant voir comment on peut ajouter des POI (Points Of Interest) sur la carte. Un point d'intérêt sera simplement défini par ses coordonnées (latitude et longitude).

```
import folium
map = folium.Map(location = [48.8658669, 2.2804251], zoom_start=20)
folium.Marker([48.8658669, 2.2804251]).add_to(map)
map.save('maCarte1.html')
```

Nous avons uniquement ajouté la ligne "folium.Marker...", il faut juste renseigner les coordonnées souhaitées (ici 48.8658669 pour la latitude et 2.2804251 pour la longitude). Il est possible d'ajouter plusieurs marqueurs sur une même carte, il suffira d'ajouter autant de ligne "folium.Marker([latitude, longitude]).add_to(map)" que de marqueurs désirés.



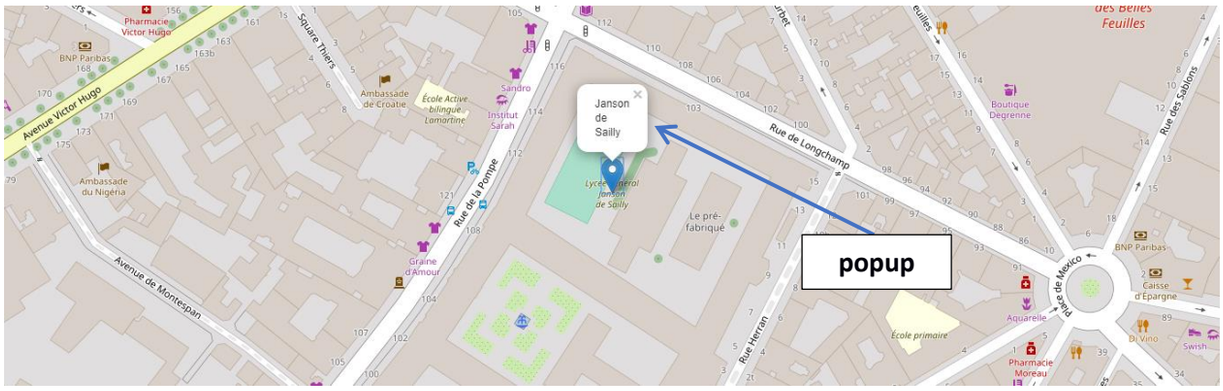
Nous allons maintenant ajouter le nom du POI ainsi qu'une image associée. Pour notre exemple, nous allons ajouter un titre à notre POI. Pour cela nous allons remplacer l'instruction :

```
folium.Marker([48.8658669, 2.2804251]).add_to(map)
```

Par l'instruction suivante :

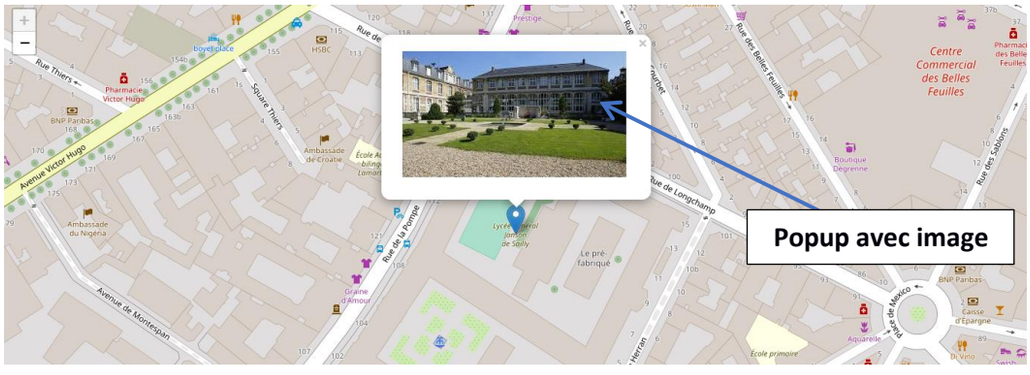
```
folium.Marker([48.8658669, 2.2804251], popup = 'Janson de Sailly').add_to(map)
```

Le paramètre **popup** permet alors d'ajouter une information sur le POI.



Pour finir nous allons ajouter une image correspondant à notre POI comme on le retrouve sur l'application de location de trottinettes (Figure 6). Pour cela, il est nécessaire d'importer la bibliothèque **base64** qui va nous permettre d'encoder l'image sous format html que nous allons encapsuler dans un cadre pour générer le **popup**. On utilise aussi la bibliothèque **PIL** pour déterminer la taille de l'image utilisée.

```
import folium
import base64
from PIL import Image

## Ces deux instructions permettent de déterminer la taille de l'image dont le nom est ici « JDS.jfif »
photo = Image.open('JDS.jfif', 'r')
largeur, hauteur = photo.size

## Ces instructions permettent d'encoder l'image et de l'encapsuler dans un cadre IFrame
encoded1 = base64.b64encode(open('JDS.jfif', 'rb').read()).decode()
html = '<img src="{}">'.format
iframe1 = folium.IFrame(html(encoded1), width=largeur+20, height=hauteur+20)
image = folium.Popup(iframe1, max_width=2650)

## Ces instructions permettent d'ajouter un popup que il a déjà été fait précédemment.
map = folium.Map(location = [48.8658669, 2.2804251], zoom_start=20)
folium.Marker([48.8658669, 2.2804251], popup = image).add_to(map)
map.save('maCarte1.html')
```



C'est ainsi que les trottinettes apparaissent sur la carte de l'application.

EXERCICE (utilisation du fichier trottinettes.py)

On suppose maintenant que 3 trottinettes sont disponibles aux alentours du lycée aux adresses suivantes : 112 rue de longchamp, 86 rue de la pompe et 25 rue decamps.

Ecrire une suite d'instructions permettant de marquer sur la carte la disponibilité de ces 3 trottinettes. Pour cela, vous devez récupérer les coordonnées GPS de chacune puis les repérer sur votre carte en utilisant un popup.

5 DECODAGE D'UNE TRAME NMEA 0183

Maintenant que nous savons afficher sur une carte issue d'OSM des points d'intérêts il semble intéressant de savoir comment sont obtenues les coordonnées GPS de chaque trottinette en temps réels. Comme nous l'avons vu chaque trottinette est équipée d'un traceur GPS. Chaque traceur envoi régulièrement (toutes les secondes d'horloge interne du GPS) la position de la trottinette via une trame NMEA à la vitesse de transmission est de 9600 bps.

Dans le cas où l'on doit récupérer une trottinette l'application utilise notre position et calcule le meilleur itinéraire.

Notre position est donnée la plupart du temps par la puce GPS du smartphone lorsque l'on utilise l'application.

Quand le GPS est désactivé l'application peut utiliser les infos des antennes relais pour trianguler notre position approximative ou encore uiliser la géolocalisation Wifi. Celle-ci exploite la position connue de certains réseaux wifi pour déterminer la position d'un appareil. Un smartphone équipé d'une puce Wifi pourra alors se baser sur les connexions wifi qu'il détecte à proximité pour estimer son emplacement géographique. La précision dépend de la puissance des points d'accès Wifi, c'est-à-dire quelques dizaines de mètres.

Certains systèmes de géolocalisation téléphone combinent les 3 méthodes pour maximiser la fiabilité de la localisation.

Le téléphone renvoie une trame GPS de type GGA (trame utilisée pour connaître la position courante du récepteur GPS).

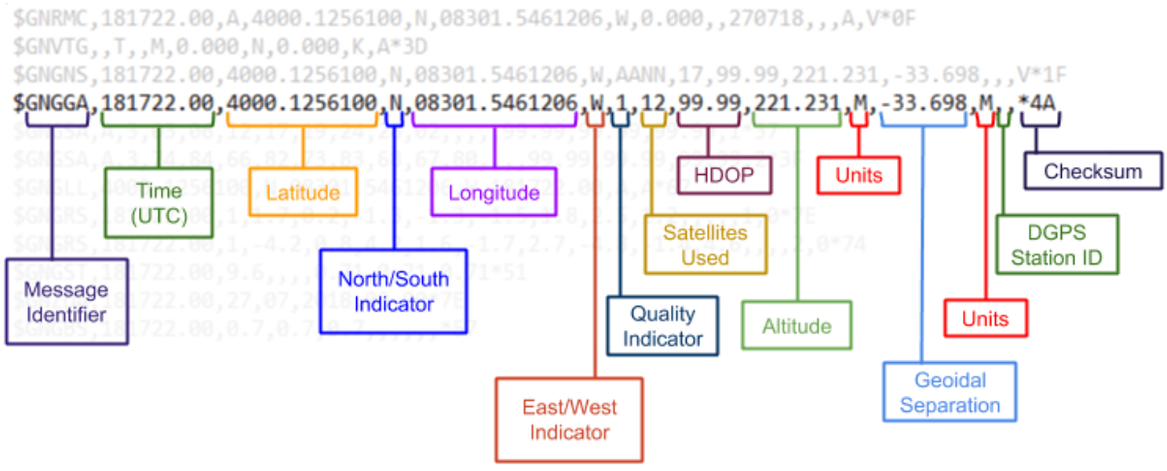


Figure 8: Exemple de trame GNGGA

Une trame NMEA est composée de 82 caractères au maximum. La Figure 8 représente un exemple de trame NMEA. Parmi ces caractères, on retrouve la latitude sous forme de degrés, minutes, secondes (exemple 4000.1256100 correspond à 40 degrés, 00 minutes et 12 secondes) suivi de l'indicateur Nord ou Sud, de même pour la longitude.

Exemple de trame NMEA

```
$GNGGA,152802.00,4345.194141,N,00122.272205,E,1,09,1.3,130.8,M,49.8,M,,*75
```

$GNGGA : Type de trame
152802.00 : Trame envoyée à 15h28m02,000s (heure UTC)
4345.194141,N : Latitude 43.7553917° Nord = 43°45'19.41" Nord
00122.272205,E : Longitude 1.3741666666666668° Est = 1°22'27.2205" Est
1 : Type de positionnement (le 1 est un positionnement GPS)
09 : Nombre de satellites utilisés pour calculer les coordonnées
1.3 : Précision horizontale ou HDOP (Horizontal dilution of precision)
130.8,M : Altitude 130.8, en mètres
,,,,,0000 : D'autres informations peuvent être inscrites dans ces champs
***75** : Somme de contrôle de parité, un simple XOR sur les caractères précédents

EXERCICE

A partir de la trame NMEA fournie dans le document annexe, déterminer la position de l'appareil utilisant la balise GPS.

Déterminer aussi la durée de transmission depuis le fichier en annexe.

6 Calcul d'itineraires

Pour pouvoir trouver le plus court chemin les GPS utilisent un algorithme (il diffère en fonction des marques) qui se rapproche de l'algorithme de Dijkstra dont vous trouverez une vidéo explicatve à partir du lien suivant :

https://www.youtube.com/watch?v=MybdP4kice4&feature=youtu.be

Pour pouvoir modéliser un trajet, nous allons utiliser la bibliothèque pyroutelib3 qui nous permet de calculer des itinéraires à l'aide des cartes OSM.

Nous commençons par importer la bibliothèque pyroutelib3.

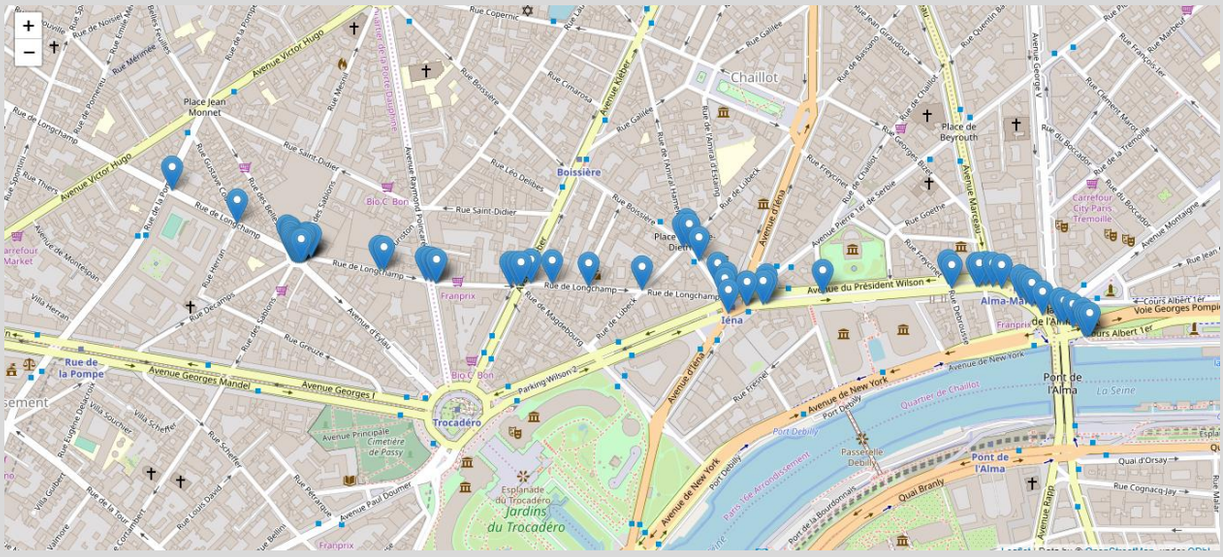
```
pip install pyroutelib3
```

ACTIVITE

Nous allons dans cette activité voir comment il est possible de tracer sur une carte OSM, un trajet à partir de Python.

Une fois la bibliothèque installée, nous allons utiliser la class **Router** comme le montre l'exemple suivant. Dans ce script, nous définissons le type de véhicule utilisé (dans notre cas un vélo assimilable à une trottinette. Nous définissons ensuite les points de départ et d'arrivée pour notre trajet. Une fois ces deux extrémités connues, la méthode **.doroute** permet de déterminer des points de passages pour notre trajet. Le chemin étant connu, on utilise la méthode **router**.**nodeLatLon** pour déterminer les coordonnées GPS (latitude et longitude) de nos points de passage représentés par la suite par des POI sur la carte OSM.

```
import folium
from pyroutelib3 import Router
router = Router("cycle") # type de véhicule pouvant être car, cycle, foot, horse, tram, train

depart = router.findNode(48.8665424, 2.2803787) # nœud correspondant au 112 rue de longchamp
arrivee = router.findNode(48.86417, 2.30237) # pont de l'alma

status, route = router.doRoute(depart,arrivee) # création des routes
if status == 'success':
    routeLatLons = list(map(router.nodeLatLon,route))

m = folium.Map(location = [48.8658669, 2.2804251], zoom_start=15)

for coord in routeLatLons:
    coord=list(coord)
    folium.Marker(coord).add_to(m)

m.save('maCarte3.html')
```



EXERCICE

Vous pouvez maintenant modifier le programme pour afficher votre propre chemin en faisant varier les moyens de transports (car, cycle, foot, horse, tram, train).