https://mesmaths.com/spip.php?article422

3-itinéraire routier

- SNT - 6-Cartographie -

Date de mise en ligne : samedi 25 mai 2019

Grâce à la fonction **Router** de la bibliothèque **pyroutelib3**, il va être possible de générer automatiquement un itinéraire entre deux points donnés.

pour commencer

Ci-dessous un code permettant de voir le fonctionnement des fonctions relatives à Router ; copiez/collez-le dans Pyzo et faites-le fonctionner :

code à copier/coller

```
##import des bibliothèques
 from pyroutelib3 import Router
 import folium

 ##choix du type de parcours et de l'icone
 router = Router("car")#on cherche à construire un parcours pour : car, cycle, foot, horse, tram, train
 icone="car"#choix de l'icone des marqueurs

 ##on identifie les points de départ et d'arrivée
 point_depart = (45.128046, 5.588483)
 point_arrivee = (45.0667,  5.55)

 depart = router.findNode(point_depart[0], point_depart[1])
 arrivee = router.findNode(point_arrivee[0],  point_arrivee[1])
 status, route = router.doRoute(depart, arrivee)
 if status == 'success':
 routeLatLons = list(map(router.nodeLatLon, route))
 #routeLatLons est un tuple qui stocke les latitudes et les longitudes
 #routeLatLons[0] est la liste des latitudes
 #routeLatLons[1] est la liste des longitudes

 ##initialisation de la carte et choix de l'échelle
 c= folium.Map(location=[(point_depart[0]+point_arrivee[0])/2, (point_depart[1]+point_arrivee[1])/2],
zoom_start=13)#carte centrée sur le milieu du segment [depart ; arrivee]
 #on marque le départ
 folium.Marker(routeLatLons[0], popup="Départ", icon=folium.Icon(icon=icone, prefix="fa",
color="green")).add_to(c)
 #on marque l'arrivée
 folium.Marker(routeLatLons[-1], popup="Arrivée après "+str(distance)+ " km", icon=folium.Icon(icon=icone,
prefix="fa", color="red")).add_to(c)

 ##on trace la route
 folium.PolyLine(routeLatLons, color="red", weight=2.5, opacity=1).add_to(c)#on trace la route en une
```

```
couleur voulue

 ##carte sauvegardée
 c.save('maRoute.html')
```

A faire : choisissez un autre lieu de départ et un autre lieu d'arrivée ; modifier le type de mode de circulation et observez.

distance

Il faut tout d'abord bien noter que dans ce programme, `routeLatLons` est une liste qui stocke les coordonnées de nombreux points qui composent le parcours.

Dans cette bibliothèque, **une fonction permet de calculer directement la distance entre deux points** dont on connaît les coordonnées (latitude, longitude).

Sa syntaxe est : `distance( coord(point) , coord(point suivant) )`.

A faire : créer une liste `d[]` qui contiendra la distance entre deux points successifs du parcours, et une liste `d_cum[]` qui contiendra la distance cumulée entre le point de départ et un point du parcours.

Aide : la fonction `sum(d)` retourne la somme des valeurs contenues dans la liste `d`.

réponse

```
##calcul des distances cumulées
 L=len(routeLatLons)#taille de la liste = nombre de points
 d=[]#initialisation de la distance : liste vide
 d_cum=[]#initialisation de la distance cumulée: liste vide
 for i in range(1, L):
 d.append(router.distance(routeLatLons[i-1],routeLatLons[i]))#liste des distances entre deux points
 d_cum.append(sum(d))#liste des distances cumulées
 distance=round(d_cum[-1], 2)#écriture arrondie à deux chiffres après la virgule
```

marquage de l'itinéraire

On peut à présent marquer des points sur l'itinéraire donnant par exemple le cumul de la distance au fur et à mesure.

On rappelle :

- pour marquer un point : `folium.Circle((latitude, longitude), radius=2)` place un cercle de rayon 2 (un point) au lieu dont les coordonnées (latitude, longitude) sont précisées.
- `d_cum=[i]` donne la valeur de la distance cumulée entre le point de départ et ième point du parcours.
- pour indiquer du texte relatif à un point, on ajouter comme paramètre `popup=" texte "` dans la fonction `folium.Circle` précédente.

réponse

```
##on marque certains points
 pas=10
 for i in range(1, L, pas):#on positionne des marqueurs tous les ...pas... points
 folium.Circle(routeLatLons[i], radius=2, popup="point n°"+str(i)+" :  "+str(round(d_cum[i],2))+"
km").add_to(c)
```

résultat attendu