5

Géographie

Dans ce chapitre, nous allons récupérer la carte d'une région géographique, l'annoter et la sauvegarder en utilisant le module Folium. La visualisation sera faite à l'aide du module Pillow ou bien en utilisant le module `webbrowser` de la bibliothèque standard de Python. Si la latitude et la longitude du lieu sont connues, le module GeoPy permet de récupérer le nom du lieu et de l'ajouter en annotation de la carte. Si le lieu est en France, l'altitude d'un lieu peut être obtenue à l'aide des services web du site de l'Institut Géographique National.

INSTALLATION NÉCESSAIRE

Dans ce chapitre, nous aurons besoin des modules complémentaires suivants :

- `geopy`
- `folium`
- `pillow`
- `requests`

Codes sources dans le dossier `chapitre_geo` 

5.1. Obtenir une carte et l'annoter

Le module Folium permet d'obtenir une carte centrée sur les cordonnées d'un lieu défini par la latitude et longitude et la carte peut être ensuite visualisée dans un navigateur en utilisant la bibliothèque standard de Python `webbrowser`.

Exemple 5.1 : *folium_ex1.py*

On importe les modules `os` ❶, `webbrowser` ❷ et `folium` ❸. Pour obtenir la carte, on construit un objet `Map` ❹ avec en argument les coordonnées du lieu (latitude et longitude) dans un type `list`.

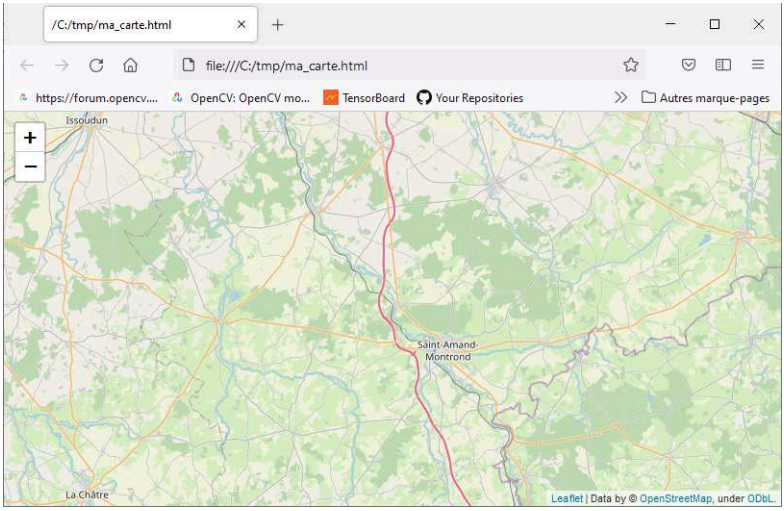
Pour visualiser la carte, il faut d'abord la sauvegarder en fichier de type HTML. En ❺, la variable `chemin_fichier` définit le chemin du fichier en utilisant la méthode `save` de la classe Map ❻ avec en argument le chemin complet du fichier. On peut ensuite visualiser la carte dans le navigateur en appelant la fonction `open` de `webbrowser` avec en argument l'URL ❼.

```
import os ❶
import webbrowser ❷
import folium ❸

carte = folium.Map([46.763, 2.425]) ❹
chemin_fichier = os.path.splitdrive(os.getcwd())[0]+"/tmp/ma_carte.html" ❺
carte.save(chemin_fichier) ❻
webbrowser.open("file://" + chemin_fichier) ❼
```

La carte centrée autour d'un point défini par sa latitude et sa longitude sélectionnées apparaît dans le navigateur (voir Figure 5.1). Notez bien que la méthode `save` de la classe Map ne sauvegarde pas une carte mais une page au format HTML qui contient un programme JavaScript permettant de visualiser une carte. Les boutons de zoom ajoutés sur la page web sont gérés par le code JavaScript.

Figure 5.1 : *Carte OpenStreetMap centrée sur 46° 45' 47" N, 2° 25' 2" E*



Exemple 5.2 : *folium_ex2.py*

Dans cet exemple, nous sauvegardons la carte au format PNG sans les boutons de zoom et en changeant l'échelle initiale.

On importe les modules Python `io` ❶, `folium` ❷ et `PIL.Image` ❸. Pour obtenir la carte, on construit un objet `Map` ❹ avec en premier argument les coordonnées du lieu (latitude et longitude) dans un type `list`, en argument nommé `zoom_start` ❺ la valeur 15 (échelle de 300m par unité sur la carte), en argument nommé `zoom_control` ❻ la valeur `False` pour ne pas afficher les boutons +/- qui permettent de changer d'échelle et en dernier argument nommé `control_scale` ❼ la valeur `True` pour afficher une échelle sur la carte.

En ❽, on convertit les données HTML constituant la carte construite au format PNG en appelant la méthode `_to_png` de la classe Map dont le résultat est du type `Bytes`. L'argument de `_to_png` est un nombre égal au temps donné avant de faire la copie de l'image construite par le module Selenium. La variable `img_data` contient le résultat de l'appel à `_to_png`.

```
import io ❶
import folium ❷
import PIL.Image ❸

carte = folium.Map([46.763, 2.425],❹
                   zoom_start=15,❺
                   zoom_control=False,❻
                   control_scale=True) ❼
img_data = carte._to_png(5) ❽
img = PIL.Image.open(io.BytesIO(img_data)) ❾
img.show() ❿
img.save("/tmp/ma_carte.png") ⓫
```

On choisit de visualiser cette image avec Pillow. Le contenu de la variable nommée `img_data` de type `Bytes` est converti en flux binaire en construisant un objet BytesIO ❾ de la bibliothèque standard de Python `io`. Cet objet est donné en argument de la fonction `open` du module `Image` de Pillow. On peut visualiser la carte dans Pillow en appelant la méthode `show` ❿ de la classe Image. Pour terminer, l'image contenant la carte est sauvegardée en appelant la méthode `save` ⓫.