29/03/2017

Etienne BOURDEAU

Laboratoire de Physique corpusculaire, AUBIERE

Manuel d’utilisation

Map\_generator.py, V1.0



Summary:

[Introduction au programme 2](#_Toc478563432)

[Analyse du Programme 2](#_Toc478563433)

[Imports 2](#_Toc478563434)

[Création de la carte 2](#_Toc478563435)

[Variables Globales 2](#_Toc478563436)

[Programme principal 3](#_Toc478563437)

[Création de variables d’environnement 3](#_Toc478563438)

[Vérification de la présence des variables 3](#_Toc478563439)

[Création des marqueurs 4](#_Toc478563440)

[Sauvegarde de la carte 5](#_Toc478563441)

[Sources 6](#_Toc478563442)

[Folium 6](#_Toc478563443)

# Introduction au programme

Ce programme a été créé par M. BOURDEAU Etienne, stagiaire au Laboratoire de Physique Corpusculaire d’Aubière. Il a été codé sous Python 3.6, et répond à une demande du laboratoire, qui avait besoin d’un moyen d’afficher sur un fonds de carte les informations rassemblées par Decodeb64.py (pour plus d’informations sur ce programme, veuillez vous référer à son manuel d’utilisation.).

Map\_generator.py utilise Folium, une application du module Javascript ‘leaflet.js’ à Python. Vous pouvez télécharger Folium depuis GitHub : <https://github.com/python-visualization/folium>

Si vous utilisez un IDE pour lancer ce programme, comme PyCharm ou Visual Studio Code, Folium peut être téléchargé automatiquement.

# Analyse du Programme

## Imports

* Folium est le seul module importé. Il utilise leaflet.js (voir Sources pour avoir des informations sur leaflet) et l’applique via des commandes Python. Il permet de créer des cartes interactives utilisant un grand nombre d’options, bien que ce programme n’utilise que des marqueurs basiques. Pour plus d’informations sur Folium, veuillez-vous référer aux Sources.

## Création de la carte

Utiliser Folium vous permet de créer une carte locale en HTML, en utilisant OpenStreetMaps.

La création de la carte prend ces deux lignes :

1. D’abord, créez une variable correspondant aux coordonnées sur lesquelles vous souhaitez concentrer votre carte.

Var1 = (VotreLatitude, VotreLongitude)

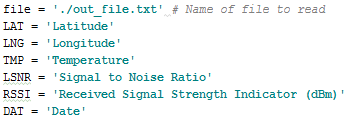
1. Puis, créez une autre variable ayant la valeur :

Var2 = folium.Map(location=Var1, zoom\_start=17)

Votre carte est maintenant enregistrée dans la mémoire de Python, cependant le programme ne crée pas encore le fichier HTML. Nous verrons à la fin du programme comment sauvegarder la carte, mais d’abord nous devons lui appliquer des marqueurs.

Note de l’auteur : La valeur derrière zoom start peut varier en fonction du niveau de rapprochement dont vous avez besoin.

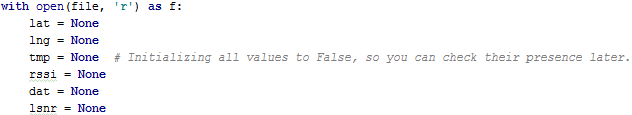
## Variables Globales



D’abord, on crée toutes les variables dont on va avoir besoin. D’abord, le fichier dont on veut extraire les informations, puis les motifs à partir desquels on veut récupérer les valeurs.

## Programme principal

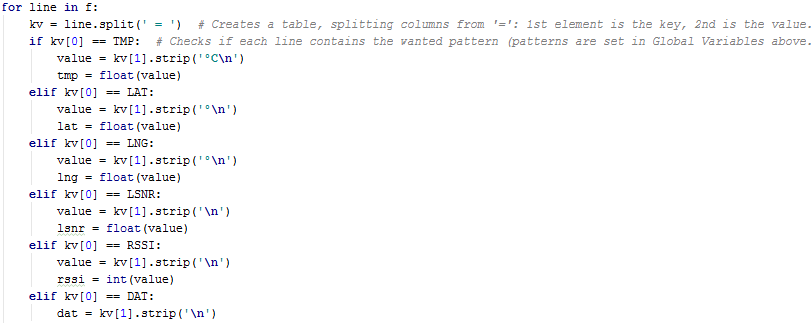
### Création de variables d’environnement



Tout d’abord, nous avons besoin d’ouvrir le fichier inscrit (voir ‘Variables Globales’). Pour y faire de nouveau appel plus simplement, nous allons le surnommer ‘f’.

Ensuite, on passe les valeurs de toutes les variables d’environnement à ‘None’, ce qui équivaut à ‘False’, afin de pouvoir vérifier leur présence plus tard.

### Vérification de la présence des variables



Cette boucle est longue, mais plutôt simple :

1 – D’abord, on crée une boucle ‘for line in f :’, ce qui signifie que nous allons vérifier le fichier ligne par ligne (Rappel : ‘line’ est une variable Python qui désigne automatiquement une ligne d’un fichier)

2 – Ensuite, nous allons séparer le fichier. Le fichier de sortie généré par ‘Decodeb64.py’ est toujours sous la forme ‘clé’ ‘=’ ‘valeur’. Séparer le fichier à partir du caractère ‘=’ va créer un tableau, qu’on appelle ‘kv’ (pour key value, ou clé valeur) dont la première colonne (kv[0]) sera la clé, et la seconde colonne (kv[1] sera la valeur.

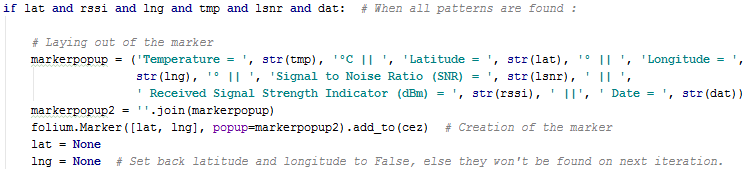
3 – Pour chaque valeur que vous souhaitez ajouter sur votre carte, vous aurez besoin de 3 lignes :

**If/elif** kv[0] == VariableGlobale:  
 value = kv[1].strip(**'caractèrespostvaleur'**)  
 envvar = float(value)

La commande ‘strip’ permet de ne pas prendre en compte les caractères après la valeur souhaitée, comme ‘°C’.

## Création des marqueurs

Votre carte est créée, et les variables que vous voulez y ajouter sont extraites. Il ne reste qu’à les ajouter sur la carte :



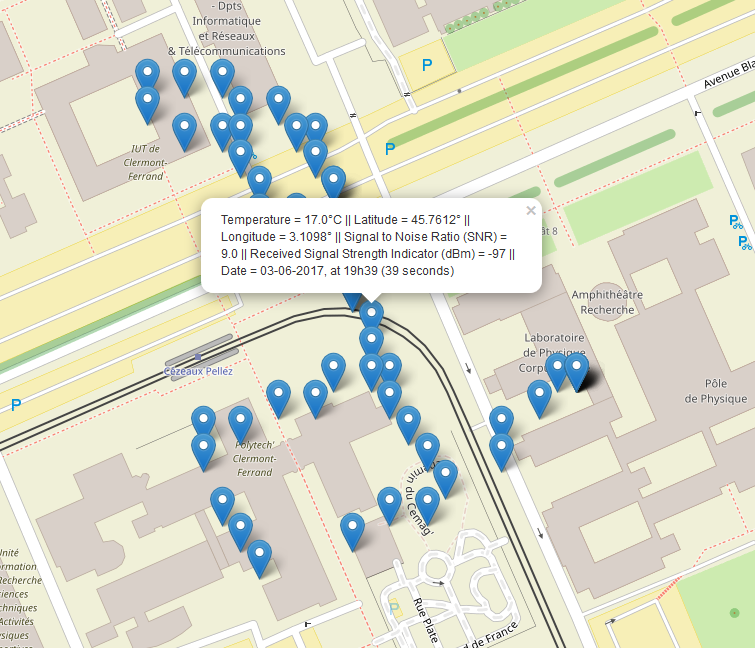
1 – Cette condition est un peu longue, mais elle signifie ‘quand tous les motifs ont été trouvés’

*Note : Sous Python, si un motif est trouvé, sa valeur passe automatiquement à True, ce qui explique cette formulation ainsi que leur initialisation à None.*

2 – On initialise une variable « markerpopup » (ou nom que vous souhaitez), égale à la mise en forme que l’on souhaite voir apparaitre sur le marqueur. Cette formulation n’accepte que les chaines de caractères, pour nos valeurs numériques il est donc nécessaire de préciser ‘str’ devant, afin qu’elles soient traitées comme des chaines.

3 – L’ensemble de ces chaines crée un « tuple », un tableau de valeurs. La commande ‘’.join permet d’en faire une chaine de caractères.

4 – On utilise la commande folium.Marker’[envvar de la latitude, envvar de la longitude], popup = markerpopup2).add\_to(nomdevotrecarte) afin d’ajouter dynamiquement les latitudes et longitudes relevées, ainsi que les autres informations relevées à chaque itération de la boucle. Ainsi, on obtient un résultat du type :



## Sauvegarde de la carte

Maintenant, votre carte est en mémoire et vos marqueurs sont créés. Il vous suffit de cette ligne pour sauvegarder le tout dans un fichier html :

Var2.save(**'./nameofyourmap.html'**)

# Sources

### Folium

Leaflet.js official page:

<http://leafletjs.com/>

Introduction to Folium (French):

<http://www.portailsig.org/content/python-leaflet-folium-ou-comment-creer-des-cartes-interactives-simplement>

Folium Official Page:

<https://pypi.python.org/pypi/folium>