

PRÁCTICA 8

CURSO BÁSICO PYTHON: FOLIUM

No description has been provided for this image

Grado Tecnologías Interactivas | Tecnologías de la Información Geográfica

FOLIUM es una biblioteca de visualización de Python que se desarrolló con el único fin de visualizar datos geoespaciales. Es una biblioteca completamente gratuita.

Tabla de Contenido

- Acerca de los conjuntos de datos (DataFrame)
- Mapa Puntual
- Mapa de calor
- Mapa de calor con función tiempo
- Tareas a realizar

Base de datos: <https://datos.madrid.es/portal/site/egob>

Documentación: <https://python-visualization.github.io/folium/>

Diferentes plugins de Folium: <https://python-visualization.github.io/folium/plugins.html>

Alumnos:

Indicar el nombre de los alumnos responsables

Estudio de la accidentalidad de la ciudad de Madrid mediante mapa de calor y puntual.

Caso de estudio

Crear un mapa de calor y puntual de los accidentes originados por vehiculos y bicicletas en el a 2022

CREACION BASE DE DATOS ESTUDIO

```
In [ ]: # Librerias necesarias
import openpyxl
import pandas as pd
import numpy as np
import folium
from folium import plugins
import webbrowser
from pyproj import Transformer
```

```
In [ ]: # Creamos la Base de datos
base_datos= pd.read_excel('2022_Accidentalidad.xlsx')
print(base_datos.dtypes) # imprime el tipo de campo
print("Numero de registros =" , len( base_datos)) # imprime el tamaño de
```

Preparación Base de datos

```
In [ ]: # Elimina los registros en blanco en los campos de coordenadas
accidentes_df = base_datos.dropna(subset=['coordenada_x_utm', 'coordenad
```

Transformación de coordenadas

La geolocalización en el fichero de accidentes del ayuntamiento de Madrid se realiza con coord UTM ETRS89 en el Huso 30. Para representar coordenadas en FOLIUM se necesitan coordenadas geográficas ETRS89

```
In [ ]: # Establecer los sistemas de transformación epsg:xxxx de entrada y epsg:
transformacion = Transformer.from_crs('epsg:.....', 'epsg:.....')
puntos = list(zip(accidentes_df.coordenada_x_utm, accidentes_df.coordenad

# Se recomienda trabajar con una copia independiente
accidentes_df = accidentes_df.copy()
coorggeo = np.array(list(transformacion.itransform(puntos)))
accidentes_df.loc[:, 'longitud'] = coorggeo[:, 0]
accidentes_df.loc[:, 'latitud'] = coorggeo[:, 1]

accidentes_df.head()
```

Selección de registros base de datos

```
In [ ]: print(accidentes_df["tipo_persona"].unique())

In [ ]: # Crear una nueva Dataframe que cumple las condiciones establecidas
acci_conductor = accidentes_df.loc[(accidentes_df['tipo_persona']=='Cond

# si quisieramos introducir más condicionantes --- & (accidentes_df['ti
```

Mapa Puntual (MarkerCluster)

Los mapas puntuales permiten plasmar variables georeferenciadas, para un mejor entendimien parte del usuario de la distribución de los mismas.

Iconos con prefijo 'fa': <https://fontawesome.com/v5.15/icons?d=gallery&p=2&m=free>

```
In [ ]: print(accidentes_df["tipo_vehiculo"].unique())
```

```

In [ ]: # Crear el mapa base donde representar
        mapa_accidentes = folium.Map(location=(40.43,-3.65), tiles = 'OpenStreet

# Crear diferentes dataframes, con tipo _persona= Conductor y Vehiculos=
coche_df=acci_conductor.loc[(acci_conductor['tipo_vehiculo']=='Turismo')
bici_df=acci_conductor.loc[(acci_conductor['tipo_vehiculo']=='Bicicleta'

# crear un objeto de grupo de marcas para los incidentes en el DataFrame
coches = plugins.MarkerCluster( name="Accidentes_coche",).add_to(mapa_ac
bicis = plugins.MarkerCluster( name="Accidentes_bicis",).add_to(mapa_acc

# procesar el DataFrame y agregar cada punto de datos al grupo de marcas
for lat, lng, label, in zip(coche_df.latitud, coche_df.longitud, coche_d
    folium.Marker(
        location=[lat, lng],
        icon=folium.Icon(color="orange", icon="car", prefix = 'fa'),
        popup=label,
    ).add_to(coches)

for lat, lng, label, in zip(bici_df.latitud, bici_df.longitud, bici_df['
    folium.Marker(
        location=[lat, lng],
        icon=folium.Icon(color="blue", icon="bicycle", prefix = 'fa'),
        popup=label,
    ).add_to(bicis)

#Añadir controles
folium.LayerControl().add_to(mapa_accidentes)
draw = plugins.Draw(export=True)
draw.add_to(mapa_accidentes)

#Muestra mapa
mapa_accidentes

# Salvar WebMapping
# mapa_accidentes.save("madrid_accidentes.html")
# webbrowser.open_new_tab('madrid_accidentes.html')

```

Ubicación de contenedores vidrio y envases e Madrid. Mapa puntual.

```

In [ ]: '''
        La base de datos es Contenedores_varios.csv, utilizamos ahora un fichero
        Es opcional colocar el tipo de separador y la codificación. Por defecto
        con formato sep=';', encoding = 'cp1252')
        '''

        contenedores_df = pd.read_csv('Contenedores_varios.csv', sep=';', low_mem
        print(contenedores_df["Tipo Contenedor"].unique())
        print(contenedores_df.dtypes) # imprime el tipo de campo

```

```

In [ ]: # Crear el mapa base donde representar
mapa_contenedores = folium.Map(location=(40.43,-3.65), tiles = 'OpenStre

# Crear diferentes dataframes
vidrio_df=contenedores_df.loc[(contenedores_df['Tipo Contenedor']=='VIDR
envases_df=contenedores_df.loc[(contenedores_df['Tipo Contenedor']=='ENV

# crear un objeto de grupo de marcas para los incidentes en el DataFrame
vidrio = plugins.MarkerCluster( name="Contenedores Vidrio").add_to(mapa_
envases = plugins.MarkerCluster( name = 'Contenedores envases').add_to(m

# procesar el DataFrame y agregar cada punto de datos al grupo de marcas
for lat, lng, label in zip(vidrio_df.LATITUD, vidrio_df.LONGITUD, vidrio
    folium.Marker(
        location=[lat, lng],
        icon=folium.Icon(color="green", icon="info-sign"),
        popup=label,
    ).add_to(vidrio)

for lat, lng, label in zip(envases_df.LATITUD, envases_df.LONGITUD, enva
    folium.Marker(
        location=[lat, lng],
        icon=folium.Icon(color="orange", icon="info-sign"),
        popup=label,
    ).add_to(envases)

# Añadir control de mapas
folium.LayerControl().add_to(mapa_contenedores)

#Muestra mapa
mapa_contenedores

# Crear mapas
# mapa_contenedores.save("madrid_contenedores.html")
# webbrowser.open_new_tab('madrid_contenedores.html')

```

Mapa de Calor (HeatMap)

Los mapas de calor, o heatmaps, son un tipo de representación dentro de de la simbología de r que nos permite mostrar al usuario los puntos calientes o conjuntos de datos más relevantes de una nube de puntos. Con una combinación de opciones de simbología y datos cuantitativos provenientes de base de datos georeferenciadas, obtendras mapas que reflejan concentraciones distribuciones espaciales de variables no continuas en el espacio.

```

In [ ]: # Utilizaremos el plugging HeatMap
# la lista de datos puede ser de 2 formas (Latitud, Longitud) o (Latitud
# Mapa de calor de accidentes de Madrid
mapa_calor = folium.Map(location=(40.43,-3.65), tiles = 'cartodb positro
plugins.HeatMap(data=acci_conductor[['latitud', 'longitud']], radius=15,
folium.LayerControl().add_to(mapa_calor)
mapa_calor

```

```
In [ ]: # publicar el mapa
mapa_calor.save("heatmap_acci.html")
webbrowser.open_new_tab('heatmap_acci.html') #Abre el HTML en una ventan
```

Mapa de calor con evolución en el tiempo (HeatMapWithTime)

Este plugin nos permite generar mapas de calor de diferentes periodos de tiempo especificados representarlos mediante automatización con un play que nos permite representar la evolución fenómeno

```
In [ ]: # Crear una nueva Dataframe que cumpla unas determinadas condiciones
acci_df=accidentes_df.loc[(accidentes_df['tipo_persona']=='Conductor') &

# Crear un campo en el Dataframe con el valor del mes extraido de la fecha
import datetime
mes = pd.DatetimeIndex(acci_df['fecha']).month
acci_df = acci_df.assign(mes = mes)

# Generar lista para el slider del visualizador
lista_tiempo = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Juli

# Generar la lista de pesos
lista_peso = [] # Crear una lista vacia
acci_df = acci_df.assign(peso = 1) # crear un campo nuevo con valor por

# Rellenar la lista con los campos (Latitud, Longitud y Peso) necesarios
for x in acci_df['mes'].sort_values().unique():
    lista_peso.append(acci_df.loc[acci_df['mes']==x, # Agrupar los valo
                        ['latitud', 'longitud', 'peso']].groupby(['latitud', 'lon

# Generar el mapa
madrid_map = folium.Map(location=(40.43, -3.65), control_scale = True, ti

# Ejecutar el plugin HeatMapWithTime
plugins.HeatMapWithTime(lista_peso, radius = 30, index = lista_tiempo,
                        auto_play = False, min_opacity = 0.5, max_opacit

# publica el mapa
folium.LayerControl().add_to(madrid_map)
madrid_map.save("madrid_acci.html")
webbrowser.open_new_tab('madrid_acci.html')
```

Mapa Dual (DualMap)

Este plugin nos permite en una misma ventana Html tener 2 mapas

Ubicación de contenedores vidrio y envases e Madrid. Mapa dual puntual.

```

In [ ]: '''
        La base de datos es Contenedores_varios.csv, utilizamos ahora un fichero
        Es opcional colocar el tipo de separador y la codificación. Por defecto
        con formato sep=';', encoding = 'cp1252')
        '''

        contenedores_df = pd.read_csv('Contenedores_varios.csv', sep=';', low_mem
        contenedores_df.head()

In [ ]: print(contenedores_df["Tipo Contenedor"].unique())

In [ ]: # Crear el mapa dual donde representar los elementos
        mapa_dual = plugins.DualMap(location=[40.4, -3.65], tiles=None, zoom_sta

        # map tiles
        folium.TileLayer('OpenStreetMap').add_to(mapa_dual.m1)
        folium.TileLayer('CartoDB Positron').add_to(mapa_dual.m2)

        # Crear diferentes dataframes, con Tipo Contenedor= VIDRIO y ENVASES
        vidrio_df=contenedores_df.loc[(contenedores_df['Tipo Contenedor']=='VIDR
        envases_df=contenedores_df.loc[(contenedores_df['Tipo Contenedor']=='ENV

        # crear un objeto de grupo de marcas para los diferentes contenedores en
        vidrio = plugins.MarkerCluster( name="Contenedores Vidrio").add_to(mapa_
        envases = plugins.MarkerCluster( name = 'Contenedores envases').add_to(m

        for lat, lng, label in zip(vidrio_df.LATITUD, vidrio_df.LONGITUD, vidrio
            folium.Marker(
                location=[lat, lng],
                icon=folium.Icon(color="green", icon="info-sign"),
                popup=label,
            ).add_to(vidrio)

        for lat, lng, label in zip(envases_df.LATITUD, envases_df.LONGITUD, enva
            folium.Marker(
                location=[lat, lng],
                icon=folium.Icon(color="orange", icon="info-sign"),
                popup=label,
            ).add_to(envases)

        # Añadir control de mapas
        folium.LayerControl().add_to(mapa_dual)

        # Crear mapas
        mapa_dual.save("madrid_dual.html")
        webbrowser.open_new_tab('madrid_dual.html')

```

TAREA 1

Realizar un webMap de la gestión de residuos en la ciudad de Ma

Crear capas diferenciadas:

2 Mapas de calor con los contenedores de los residuos vidrio y los residuos envases.

4 mapas puntuales con la ubicación de los contenedores diferenciados por el tipo de residuo (vidrio, envases, papel, organico).

Guardar como: contenedores_"apellido alumno".html

```
In [ ]: # Librerías necesarias
import openpyxl
import pandas as pd
import numpy as np
import folium
from folium import plugins
import webbrowser
from pyproj import Transformer
```

```
In [ ]: # Abrir la base de datos Contenedores_varios.csv
```

```
# publicar el mapa. Contenedores_"apellido alumno".html
```

TAREA 2

Realizar webmapping con mapas de calor de accidentes en 2023 distribuidos por meses (HeatMapWithTime)

Realizar varios mapas de calor de accidentes producidos en la ciudad de Madrid, a lo largo del año distribuido por meses, que se han visto involucrados conductores diferenciados por sexo (Elegir Hombre o Mujer) y con vehículos distintos al turismo.

```
In [1]: # Abrir la base de datos 2023_Accidentalidad.xlsx
# Recuerda limpiar la base de datos y transformar las coordenadas
```

```
# publica el mapa. Accidentes_Madrid_"apellido alumno".html
```


