zzs4l5n0y

January 26, 2025

Dr. Jesús Martí Gavilá. Departamento de Ingeniería Cartográfica, Geodesia y fotogrametría

PRÁCTICA 8

CURSO BÁSICO PYTHON: FOLIUM

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

```
<ul>
    <li>
        <a>Acerca de los conjuntos de datos (DataFrame)</a>
    </li>
    <li>
        <a>Mapa Puntual</a>
    </li>
    <li>
        <a>Mapa de calor</a>
    </li>
    <li>
        <a>Mapa de calor con función tiempo</a>
    </li>
    <li>
        <a>Tareas a realizar</a>
    </li>
</ul>
```

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 año 2022

CREACION BASE DE DATOS ESTUDIO

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

[ ]:
```
# 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 la␣
 ↪base de datos
```

1 Preparación Base de datos

[ ]:
```
# Elimina los registros en blanco en los campos de coordenadas
accidentes_df = base_datos.dropna(subset=['coordenada_x_utm',␣
 ↪'coordenada_y_utm'])
```

2 Transformación de coordenadas

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

[ ]:
```
# Establecer los sistemas de transformación epsg:xxxx de entrada y epsg:yyyyy␣
 ↪de salida
transformacion = Transformer.from_crs('epsg:.....','epsg:.....
 ↪',always_xy=True)
puntos = list(zip(accidentes_df.coordenada_x_utm,accidentes_df.
 ↪coordenada_y_utm)) # Crea una lista con todos los puntos

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

accidentes_df.head()
```

3 Selección de registros base de datos

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

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

# si quisieramos introduccir más condicionantes --- &␣
↪(accidentes_df['tipo_vehiculo']=='Turismo')&␣
↪(accidentes_df['rango_edad']=='De 18 a 20 años')]
```

Mapa Puntual (MarkerCluster)

Los mapas puntuales permiten plasmar variables georeferenciadas, para un mejor entendimiento por parte del usuario de la distribucción de los mismas.

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

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

[ ]:
```
# Crear el mapa base donde representar
mapa_accidentes = folium.Map(location=(40.43,-3.65), tiles = 'OpenStreetMap',␣
↪zoom_start = 12)

# Crear diferentes dataframes, con tipo _persona= Conductor y Vehiculos=␣
↪Turismo y Bicicleta
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_accidentes)
bicis = plugins.MarkerCluster( name="Accidentes_bicis",).add_to(mapa_accidentes)

# procesar el DataFrame y agregar cada punto de datos al grupo de marcas creado␣
↪anteriormente
for lat, lng, label, in zip(coche_df.latitud, coche_df.longitud,␣
↪coche_df['num_expediente']): #dos formas diferentes de llamar a un campo
    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['num_expediente']): #dos formas diferentes de llamar a un campo
    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 en Madrid. Mapa puntual.

[ ]:
```
'''
La base de datos es Contenedores_varios.csv, utilizamos ahora un fichero CSV
Es opcional colocar el tipo de separador y la codificación. Por defecto es , y␣
 ↪UTF-8 peró hay ficheros csv
con formato sep=';', encoding = 'cp1252')
'''
contenedores_df = pd.read_csv('Contenedores_varios.csv', sep=';',low_memory =␣
 ↪False)
print(contenedores_df["Tipo Contenedor"].unique())
print(contenedores_df.dtypes) # imprime el tipo de campo
```

[ ]:
```
# Crear el mapa base donde representar
mapa_contenedores = folium.Map(location=(40.43,-3.65), tiles = 'OpenStreetMap',␣
 ↪zoom_start = 12)

# Crear diferentes dataframes
vidrio_df=contenedores_df.loc[(contenedores_df['Tipo Contenedor']=='VIDRIO')]
envases_df=contenedores_df.loc[(contenedores_df['Tipo Contenedor']=='ENVASES')]

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

# procesar el DataFrame y agregar cada punto de datos al grupo de marcas creado␣
 ↪anteriormente
for lat, lng, label in zip(vidrio_df.LATITUD, vidrio_df.LONGITUD,␣
 ↪vidrio_df['Descripcion Modelo']):
```

```
    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,␣
 ↪envases_df['Descripcion Modelo']):
    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 mapas que nos permite mostrar al usuario los puntos calientes o conjuntos de datos más relevantes dentro 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 o distribuciones espaciales de variables no continuas en el espacio.

[ ]:
```
# Utilizaremos el pluging HeatMap
# la lista de datos puede ser de 2 formas (Latitud, Longitud) o (Latitud,␣
 ↪Longitud y Peso).
# Mapa de calor de accidentes de Madrid
mapa_calor = folium.Map(location=(40.43,-3.65), tiles = 'cartodb positron',␣
 ↪zoom_start = 12)
plugins.HeatMap(data=acci_conductor[['latitud', 'longitud']], radius=15,␣
 ↪use_local_extrema=True, name="Mapa de calor").add_to(mapa_calor)
folium.LayerControl().add_to(mapa_calor)
mapa_calor
```

[ ]:
```
# publicar el mapa
mapa_calor.save("heatmap_acci.html")
webbrowser.open_new_tab('heatmap_acci.html') #Abre el HTML en una ventana del␣
 ↪navegador
```

Mapa de calor con evolución en el tiempo (HeatMapWithTime)

Este plugin nos permite generar mapas de calor de diferentes periodos de tiempo especificados y representarlos mediante automatizacion con un play que nos permite represesentar la evolución del fenómeno

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

# 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', 'Julio',␣
 ↪'Agosto', 'Septiembre', 'Octubre', 'Noviembre','Diciembre']

# 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␣
 ↪defecto 1

# Rellenar la lista con los campos (Latitud, Longitud y Peso) necesarios para␣
 ↪HeatMap
for x in acci_df['mes'].sort_values().unique():
   lista_peso.append(acci_df.loc[acci_df['mes']==x,    # Agrupar los valores con␣
 ↪latitud y longitud igual
                  ['latitud','longitud','peso']].
 ↪groupby(['latitud','longitud']).sum().reset_index().values.tolist())

# Generar el mapa
madrid_map = folium.Map(location=(40.43,-3.65), control_scale = True, tiles =␣
 ↪"cartodb positron", zoom_start = 12)

# Ejecutar el pluging HeatMapWithTime
plugins.HeatMapWithTime(lista_peso, radius = 30, index = lista_tiempo,
                        auto_play = False, min_opacity = 0.5, max_opacity = 1,␣
 ↪use_local_extrema=True, name="Accidentes_Turismo_2022").add_to(madrid_map)

# 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 en Madrid. Mapa dual puntual.

[ ]:
```
'''
La base de datos es Contenedores_varios.csv, utilizamos ahora un fichero CSV
Es opcional colocar el tipo de separador y la codificación. Por defecto es , y␣
 ↪UTF-8 peró hay ficheros csv
con formato sep=';', encoding = 'cp1252')
'''
contenedores_df = pd.read_csv('Contenedores_varios.csv', sep=';',low_memory =␣
 ↪False)
contenedores_df.head()
```

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

[ ]:
```
# Crear el mapa dual dondce representar los elementos
mapa_dual = plugins.DualMap(location=[40.4, -3.65], tiles=None, zoom_start=20)

# 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']=='VIDRIO')]
envases_df=contenedores_df.loc[(contenedores_df['Tipo Contenedor']=='ENVASES')]

# crear un objeto de grupo de marcas para los diferentes contenedores en el␣
 ↪DataFrame
vidrio = plugins.MarkerCluster( name="Contenedores Vidrio").add_to(mapa_dual.m1)
envases = plugins.MarkerCluster( name = 'Contenedores envases').
 ↪add_to(mapa_dual.m2)

for lat, lng, label in zip(vidrio_df.LATITUD, vidrio_df.LONGITUD,␣
 ↪vidrio_df['Descripcion Modelo']):
    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,␣
 ↪envases_df['Descripcion Modelo']):
    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 Madrid.

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

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

[ ]:
```
# 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 2023 y distribuido por meses, que se han visto involucrados conductores diferenciados por sexo (Elegir: Hombre o Mujer) y con vehiculos distintos al turismo.

[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
```

[ ]:
```

```