## PREPARACIÓN PREVIA

En este ejercicio vamos a trabajar con mapas, para lo que usaremos la librería folium[1]. Para instalar la librería folium abra una ventana de comandos de Anaconda (Anaconda Prompt) y ejecute el siguiente comando:

```
conda install -c conda-forge folium
```

En este proyecto trabajaremos con datos proporcionados por la diputación de Cádiz de centros sanitarios de los municipios con población inferior a 50.000 habitantes de la provincia de Cádiz.

Los datos de los que partimos en el proyecto están en el fichero `centrosSanitarios.csv`. Si abre el fichero, comprobará que es un fichero en formato CSV (los datos están separados por punto y coma). Un extracto del mismo lo puede ver en la Figura 1:

```
NOMBRE;LOCALIDAD;LATITUD;LONGITUD;ESTADO;NUM_CAMAS;TIENE_ACCESO_DISCAPACITADOS;TIENE_UCI
CONSULTORIO ZAHARA DE LOS ATUNES; BARBATE; 36.135051666002795; -5.843455923196172; BUENO; 0; true; false
CENTRO DE SALUD BARBATE; BARBATE; 36.189308321456515; -5.925475089376914; BUENO; 0; true; false
CONSULTORIO JEDULA; ARCOS DE LA FRONTERA; 36.72460010332263; -5.931456684613025; BUENO; 0; true; false
```

**Figura 1. Extracto del fichero centrosSanitarios.csv**

Como puede observar cada línea del fichero tiene el nombre de un centro sanitario, la localidad, la latitud, la longitud, el estado del centro, el número de camas que tiene, si tiene acceso para discapacitados y si tiene UCI. (en ambos casos el valor false representa que no tiene acceso o no tiene UCI, respectivamente).

En el proyecto tendremos que implementar algunas funciones que nos ayuden a explotar los datos que disponemos. El objetivo del ejercicio es que trabaje creando un proyecto desde cero, e implemente tanto las funciones como los tests para comprobar que las funciones trabajan como se espera. El proyecto Spyder creado debe tener la estructura que se muestra en la Figura 2:

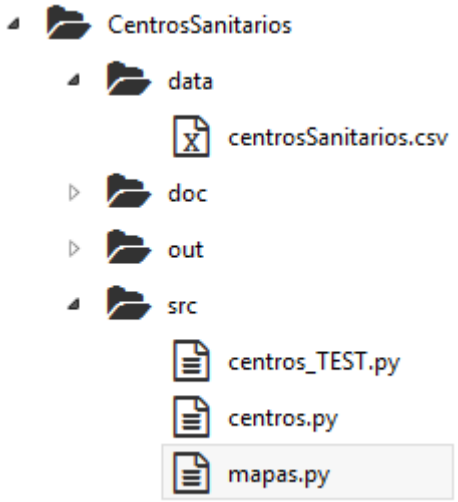


**Figura 2. Estructura del proyecto**

Implemente las funciones que se especifican a continuación en el módulo centros.py. Una vez implementada una función, implemente su test correspondiente en el módulo centros_TEST.py y compruebe que funciona como se espera.

[1] https://python-visualization.github.io/folium/quickstart.html

## *lee_centros*

**Definition :** `lee_centros(fichero)`

Lee el fichero de entrada y devuelve una lista de tuplas de tipo CentroSanitario

**ENTRADA:**

- fichero: nombre del fichero de entrada -> str

**SALIDA:**

- lista de tuplas CentroSanitario(nombre,localidad, latitud, longitud, estado, num_camas, acceso_minusvalidos, tiene_uci) -> [(str, str, float, float, str, int, bool, bool)]

Cada línea del fichero se corresponde con los datos de un centro sanitario y se representa con una tupla con los siguientes valores:

- Nombre del centro sanitario
- Localidad en la que está situado el centro sanitario
- Latitud
- Longitud
- Estado
- Número de camas
- Es accesible
- Tiene UCI

## *num_total_camas_centros_accesibles*

**Definition :** `num_total_camas_centros_accesibles(centros)`

Calcula el número total de camas de los centros que son accesibles para discapacitados.

**ENTRADA:**

- centros: lista de tuplas CentroSanitario(nombre,localidad, latitud, longitud, estado, num_camas, acceso_minusvalidos, tiene_uci) -> [(str, str, float, float, str, int, bool, bool)]

**SALIDA:**

- total camas-> int

Toma como entrada una lista de tuplas CentroSanitario(nombre,localidad, latitud, longitud, estado, num_camas, acceso_minusvalidos, tiene_uci) y produce como salida un entero correspondiente al número total de camas de los centros accesibles para discapacitados.

## *centros_con_uci_cercanos*

**Definition :** `centros_con_uci_cercanos(centros, punto, umbral)`

Selecciona los centros que están a una distancia menor o igual que un umbral del punto dado como parámetro

**ENTRADA:**

- centros: lista de tuplas CentroSanitario(nombre,localidad, latitud, longitud, estado, num_camas, acceso_minusvalidos, tiene_uci) -> [(str, str, float, float, str, int, bool, bool)]
- punto: tupla con la latitud y logitud del punto -> (float, float)
- umbral: distancia mayor en la que deben estar los centros ->float

**SALIDA:**

- lista de tuplas-> (str, str, float, float)

Toma como entrada una lista de tuplas CentroSanitario(nombre,localidad, latitud, longitud, estado, num_camas, acceso_minusvalidos, tiene_uci) y produce como salida una lista con el nombre del centro, la localidad, la latitud y la longitud de los centros situados a una distancia menor o igual que el umbral del punto dado como parámetro.

## distancia

**Definition :** `distancia(latitud1, longitud1, latitud2, longitud2)`

Función auxiliar.

**ENTRADA:**

- latitud1: latitud de un punto -> float
- longitud1: longitud de un punto -> float
- latitud2: latitud de un punto -> float
- longitud2: longitud de un punto -> float

**SALIDA:**

- distancia -> float

Toma como entrada la latitud y longitud de dos puntos y devuelve la distancia euclídea

## media_coordenadas

**Definition :** `media_coordenadas(centros)`

Calcula un punto cuya latitud es la media de las latitudes de los centros que se pasan como parámetro y cuya longitud es la media de las longitudes de los centros.

**ENTRADA:**

- centros: lista de tuplas (nombre, localidad, latitud, longitud) => (str, str, float, float)

**SALIDA:**

- tupla (latitud, longitud) -> (float, float)

Toma como entrada una lista de tuplas (nombre, localidad, latitud, longitud) y devuelve una tupla (media_latitud, media_longitud) con la media de las latitudes de los centros y la media de las longitudes

## generar_mapa

**Definition :** `generar_mapa(centros, fichero)`

Genera un archivo html con un mapa en el que están geolocalizados los centros que se pasan como parámetro.

**ENTRADA:**

- centros: lista de tuplas (nombre, localidad, latitud, longitud) => (str, str, float, float)
- fichero: nombre del archivo html generado

Toma como entrada una lista de tuplas (nombre, localidad, latitud, longitud) y genera un archivo html con un mapa y los iconos geolocalizados.

Para implementar la función `generar_mapa` ayúdese de las funciones auxiliares que se implementan en el módulo mapas.py. Además, tenga en cuenta que:

1. Primero debe crear un mapa
2. Después debe ir creando marcadores y añadiéndolos al mapa. Use `marcador.add_to(mapa)` para añadir un marcador al mapa.
3. Una vez añadidos todos los marcadores, guarde el mapa en el archivo html con `mapa.save(fichero)`.

El resultado deber ser un fichero con un mapa similar al de la Figura 3.

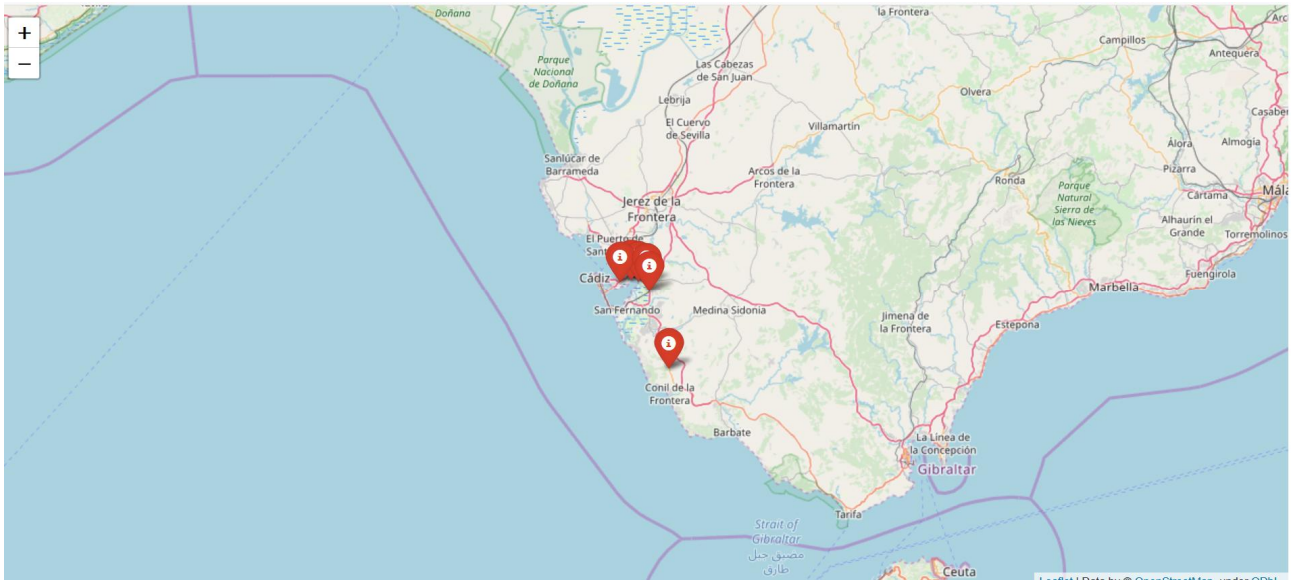


**Figura 3. Mapa con centros de salud geolocalizados**

## RETO

La fórmula de la distancia Euclídea empleada en este proyecto no es la más adecuada cuando se quieren calcular distancias entre dos puntos del globo terrestre, ya que no tiene en cuenta la curvatura de la tierra. Para calcular la distancia entre dos puntos del globo terrestre se usa una aproximación que viene dada por la fórmula de Harvesine[1]. Implemente una función distancia_harvesine para que en el proyecto se hagan unos cálculos más realistas.

[1] https://es.wikipedia.org/wiki/F%C3%B3rmula_del_haversine