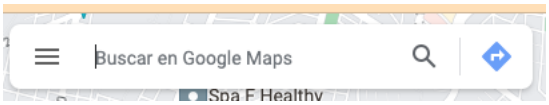
Para esta práctica vamos a trabajar con datos generados por Google Maps

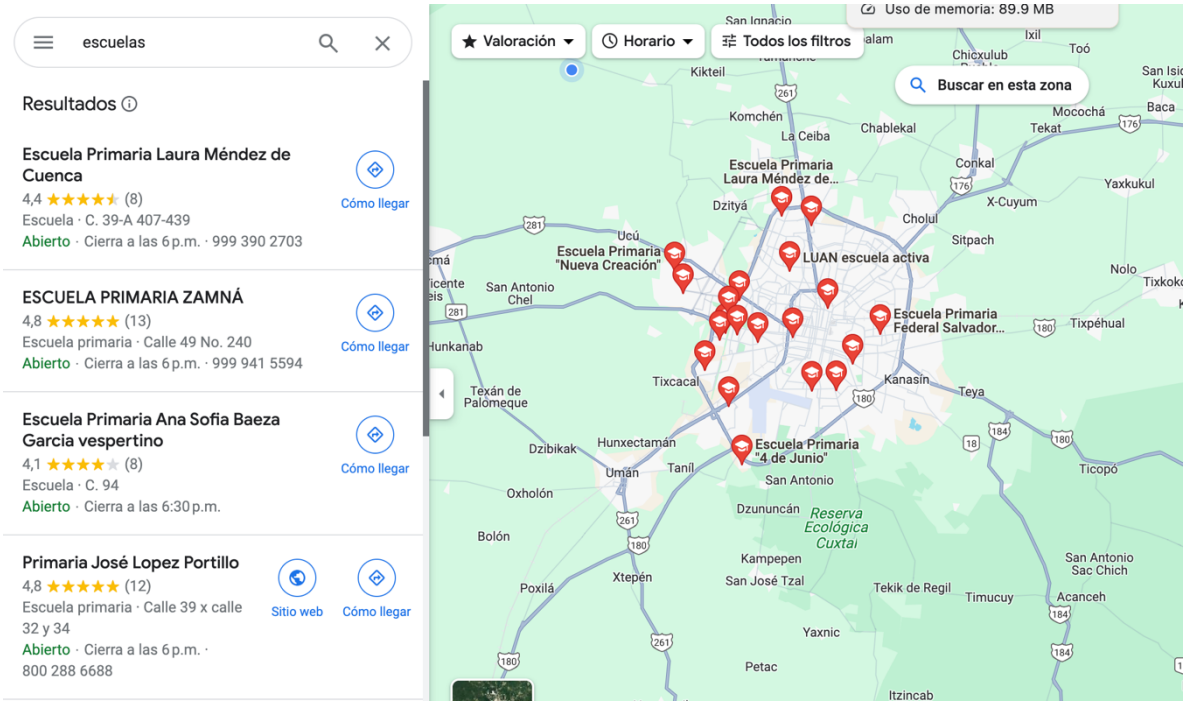
Lo primero es entrar a la página de Google Maps (https://www.google.com.mx/maps/). Verás que Google Maps automáticamente detecta nuestra ubicación aproximada y nos muestra un mapa de la zona donde nos encontramos. De hecho, si ponemos atención, podremos ver en la url que se genera, unos valores que corresponden a las coordenadas del sitio donde nos encontramos, por ejemplo:
https://www.google.com.mx/maps/@20.9857609,-89.5214493,15z?entry=ttu

En este caso, me refiero a los valores 20.9857609 y -89.5214495.

Una vez que nos encontremos en Google Maps, vamos a ubicarnos en alguna zona de interés y vamos a escribir algún criterio de búsqueda en el cuadro destinado para ello en la página de Google Maps.



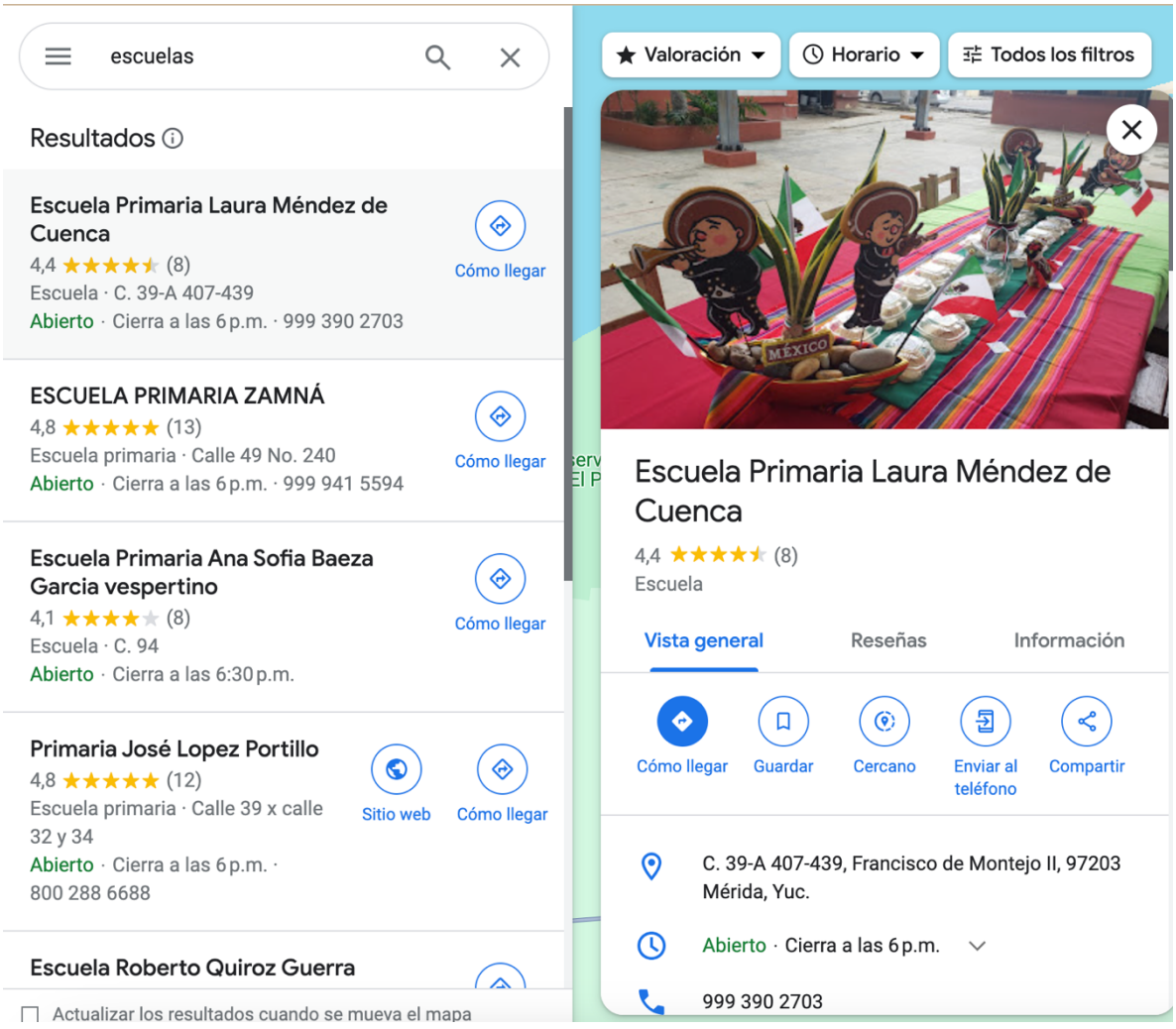
Por ejemplo, si estuviéramos en la ciudad de Mérida y necesitáramos encontrar alguna escuela veríamos algo como lo que se muestra a continuación.



Fíjate que esta búsqueda genera dos tipos de resultados. Del lado izquierdo muestra una lista de sitios encontrados que corresponden al criterio de búsqueda y, del lado derecho muestra su ubicación en el mapa.

Los datos por lo pronto se encuentran en Internet, pero ¿Qué puedo hacer si quiero integrarlos a un sistema?, es decir, ¿Puedo utilizar esta información para crear un mapa propio? Para buena suerte la respuesta es sí. De hecho, hay dos formas de hacerlo. La primera de ellas la vamos a llevar a cabo ahora y la segunda quedará como práctica para quien desee realizarla y aprender un poco más.

Entonces, lo que haremos es ir seleccionando opción por opción los resultados que se muestran en la lista que vemos en el lado izquierdo (en este caso escuelas). Por ejemplo, si seleccionamos la primera opción podremos ver algo como lo que se muestra en la siguiente imagen.



Puedes ver que aparece una nueva ventana con información detallada de la escuela que seleccionamos. Además, si seleccionas alguno de los puntos (escuelas) en el mapa, puedes observar que la url que se genera es algo parecido a:

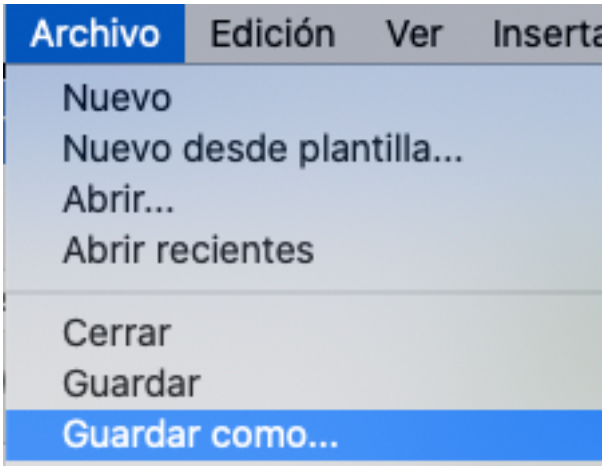
https://www.google.com/maps/place/Escuela+Carlos+Castro+Morales/@21.0428409,-89.9251025,11z/data=!4m10!1m2!2m1!1sescuelas!3m6!1s0x8f5673c3bd8ebfbf:0xf044684506f2673!8m2!3d20.9695771!4d-89.6377446!15sCghlc2N1ZWxhc5IBDnByaW1hcnlfc2Nob29s4AEA!16s%2Fg%2F1pv5y2y8m?authuser=0&entry=ttu

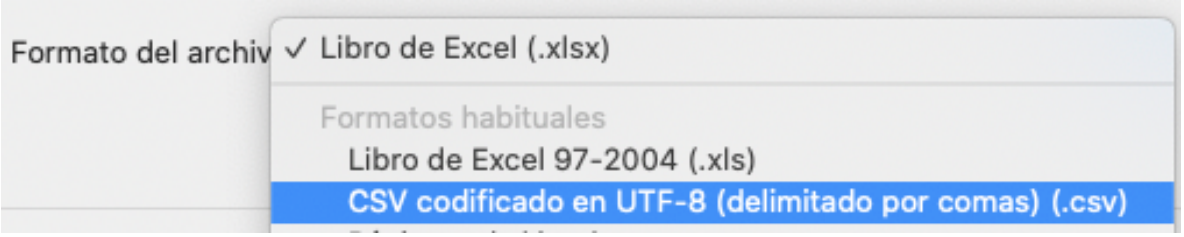
Podemos notar que esta url también contiene información, principalmente el nombre de la escuela y su ubicación dada a través de sus coordenadas.

Ahora vamos a crear un archivo para almacenar esta información. Para ello podemos usar Excel o cualquier otro software de hoja de cálculo que acostumbres a usar y vamos a poner, en la primera columna el nombre, en la segunda el valor de la latitud y en la tercera la longitud, las cuales podemos copiar y pegar de la url. Por ejemplo:

Nombre	Latitud	Longitud
Escuela Primaria Laura Méndez de Cuenca	21.0428409	-89.6449511
Escuela Primaria Zamná	21.0370622	-89.6255788
Escuela Primaria Ana Sofía Baeza García vespertino	20.996205	-89.7087103

Repite este proceso para al menos 10 escuelas o sitios que hayas elegido. Cuando termines no guardes el archivo como .xlsx sino como .csv, para ello elige la opción ‘Guardar como’ del menú ‘Archivo’ y, en el formato de archivo selecciona la opción CSV (delimitado por comas) como se muestra a continuación. Es altamente recomendable que no utilices espacios en blanco ni caracteres especiales en el nombre del archivo. Esto es porque, un poco más adelante vamos a utilizar ese archivo como fuente de datos en Google Colab, y los espacios y caracteres especiales como -/()&$·, etc no siempre son reconocibles en el código. El nombre que utilizaremos para guardar el archivo será escuelas.csv





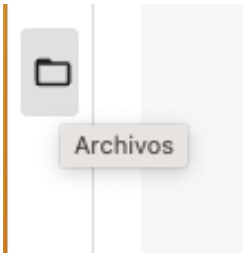
Una vez que tenemos un conjunto de datos (dataset) en un archivo csv vamos a usar una pizca de Python para transformarlos a un formato que podamos usar para visualizarlos en un mapa.

Entonces,

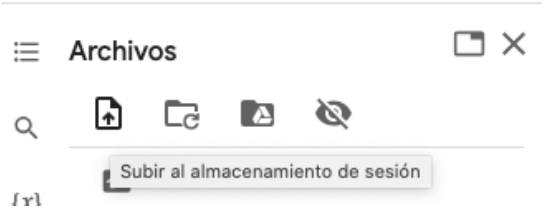
Entra de nuevo a **Google Colab.**

Para este ejercicio vamos a usar el archivo con la información de las escuelas como fuente de datos, y vamos a crear un archivo .js con los datos que contiene, los cuales fueron obtenidos de Google Maps.

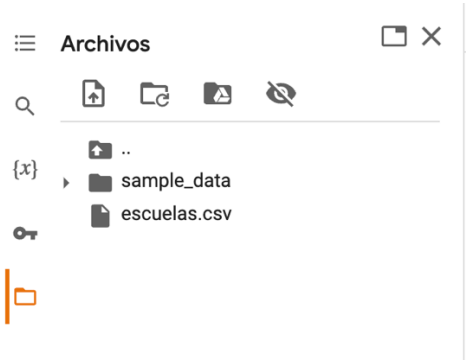
Ahora sí, podemos comenzar. No vamos a entrar en detalle con respecto al código en Python que vamos a escribir, solamente será una explicación sencilla. Comencemos por "subir" el archivo csv con la información de las escuelas a Google Colab. Para ello, localiza del lado izquierdo de Google colab el ícono con forma de carpeta/folder



Pulsa sobre él y, en el espacio que se abre pulsa sobre el ícono con una flecha hacia arriba.



Al hacer esto se abrirá una ventana que te permite seleccionar el archivo que quieras subir a Google Colab. Localiza en tu sistema de archivos el que tenga la información sobre las escuelas y súbelo a Google Colab. Al final deberás ver algo como lo que se muestra a continuación:



En la primera celda de código escribe lo siguiente:

```
import pandas as pd
```

Esta línea importa la librería que vamos a necesitar para ejecutar el código que vamos a escribir a continuación.

Si quieres conocer más a detalle qué son las librerías te recomiendo que le eches un ojo a este enlace:

https://acortar.link/ccFhin

Ahí podrás encontrar una breve explicación y algunos ejemplos de librerías para hacer visualizaciones en Python.

Además, para poder tener acceso a todas las funcionalidades de la librería Pandas, hemos creado un objeto al que se le ha llamado **pd**. Las palabras Obtejo, Método, Propiedades, etc., que verás en algunas partes de esta práctica y de la siguiente pertenecen al paradigma de la programación orientada a objetos. Si quieres conocer un poco más acerca de este paradigma te invito a que visites este enlace (https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/)

Vamos a utilizar la librería Pandas para leer el archivo csv que contiene los datos de las escuelas y almacenar temporalmente dichos datos en memoria en una estructura conocida como dataframe (cuadro de datos), es decir, que los datos estarán almacenados en forma de un cuadro o tabla con filas y columnas.

Agrega otra celda de código y escribe en ella lo siguiente:

```
df_escuelas = pd.read_csv('escuelas.csv')
df_escuelas.head()
```

En la primera línea estamos leyendo el achivo ‘escuelas.csv’ por medio de la función read_csv() invocada a partir del objeto pd de la librería Pandas. La segunda línea muestra el

encabezado (los primeros 5 registros) del archivo que acabamos de leer. Deberás ver algo similar a lo que se muestra en la siguiente imagen.

	Nombre	Latitud	Longitud
0	Escuela Primaria Laura Méndez de Cuenca	21.042841	-89.644951
1	Escuela Primaria Zamná	21.037062	-89.625579
2	Escuela Primaria Ana Sofía Baeza García vesper...	20.996205	-89.708710

Puedes ver que esta tabla tiene cuatro columnas, la primera de ellas corresponde al índice y no tiene encabezado, las otras tres corresponden a los campos o columnas del archivo escuelas.csv de donde estamos tomando los datos. En este caso el índice sirve para dar un número a cada fila/renglón/registro según el orden en el que esté.

Ahora que tenemos la información en memoria, procederemos a crear un archivo con extensión .js con una estructura u organización de datos en formato GeoJSON.

Si quieres conocer más a detallle cómo se conforma la estructura de un archivo GeoJSON puedes consultar el siguiente enlace:

https://geojson.org/

Agrega una nueva celda de código y escribe lo siguiente:

```
tamaño = len(df_escuelas)
i=1
```

La primera línea está calculando la longitud del dataframe df_escuelas, es decir, la cantidad de registros que contiene. La segunda línea simplemente está declarando una variable a la que se le llamó **i** con un valor inicial de 1.

Esto se hace porque lo que buscamos como resultado es un archivo con un contenido similar a lo siguiente:

```
var escuela = { 'type':'FeatureCollection',
'features': [
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Laura Méndez de Cuenca'},
 'geometry': {'type':'Point', 'coordinates':[ -89.644951, 21.042841]}},
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Zamná'},
 'geometry': {'type':'Point', 'coordinates':[-89.625579, 21.037062]}},
 {'type':'Feature',
 'properties':{'nombre':'Escuela primaria Amna Sofía Baeza García vespertino'},
 'geometry': {'type':'Point', 'coordinates':[-89.708710, 20.996205]}}
```

]};

Como te podrás dar cuenta, esta estructura contiene datos que siguen un patron. Puedes ver que se repite varias veces la estructura:

{'type':'Feature',
'properties':{'nombre':'Escuela Primaria Laura Méndez de Cuenca'},
'geometry': {'type':'Point', 'coordinates':[-89.644951, 21.042841]}},

Pero, por cada vez que se repite esta estructura cambian el valor del nombre y de las coordenadas.

Además, puedes ver que, siempre, después del valor de las coordenadas el texto termina con **}},** pero, en la última fila el texto termina con

}}
]};

Es decir, se elimina la coma después de la segunda llave y se agrega un corchete, una llave y un punto y coma para indicar que esos son los datos que corresponden al último registro y que, por lo tanto, ahí debe terminar el archivo.

Entonces, es por ello que es necesario conocer el tamaño o longitud del archivo csv que contiene los datos, y eso lo hacemos con la instrucción (tamaño = len(df_escuelas)). De igual forma, en el código, la variable **i** sirve para saber si estamos en el primero, segundo, tercero, etc., o en el último registro. De esa forma, cuando el programa haya leído todos los registros del csv y haya llegado al último, en lugar de escibir un }}, escribirá un }}]};

Para ver esto en funcionamiento agrega una nueva celda de código y escribe lo siguiente. Recuerda que, por cuestión de espacio, es posible que el código aparezca en un orden diferente a como lo veríamos en Google Colab, por lo que, si asi lo prefieres, puedes copiar y pegar.

```
jsfile = ""
jsfile = "var escuela = { 'type':'FeatureCollection', \n"
jsfile = jsfile + "'features': [ \n"

for index, row in df_escuelas.iterrows():
  nombre = row['Nombre']
  longitud = row['Longitud']
  latitud = row['Latitud']
  if i<tamaño:
    jsfile = jsfile +" {'type':'Feature', \n
'properties':{'nombre':'"+nombre+"'}, \n 'geometry': {'type':'Point',
'coordinates':["+str(longitud)+","+str(latitud)+"]}},\n"
```

```
    i+=1
  else:
    jsfile = jsfile +" {'type':'Feature', \n
'properties':{'nombre':'"+nombre+"'}, \n 'geometry': {'type':'Point',
'coordinates':["+str(longitud)+","+str(latitud)+"]}} \n]};"
```

En este código estamos creando una variable a la que hemos denominado **jsfile** (primera línea de este bloque de código) y a la cual le agregaremos el contenido del archivo que necesitamos, por lo pronto a manera de texto, para posteriormente copiar ese texto en el archivo .js que mencionamos anteriormente.

Las segunda y tercera líneas del bloque de código anterior comienzan a crear el texto que posteriormente almacenaremos en un archivo. A partir de ahí, se realiza una iteración registro por registro del contenido del dataframe que contiene los datos de las escuelas (df_escuelas) y, por cada iteración o vuelta se toma el contenido de lo que hay en la columna Nombre (**row['Nombre']**), en la columna Longitud (**row['Longitud']**) y en la columna Latitud (**row['Latitud']**), y se almacena en las variables **nombre**, **longitud** y **latitud**, resectivamente. Aquí hay que tener mucho cuidado de poner entre corchetes el nombre de la columna tal y como está en el dataframe, respetando mayúsculas y minúsculas.

Además, por cada iteración el programa verifica si el valor de **i** aún es menor que el de la variable **tamaño**

if i<tamaño:

de ser así, significa que aún no hemos llegado al último registro y, en ese caso, se agrega texto a la variable **jsfile** terminando con **}},** y se aumenta el valor de **i** en uno con `i+=1`

En caso de que el valor de **i** ya no sea menor al tamaño (else) se agrega texto a la variable **jsfile** terminando con **}}]};**

El aumentar el valor de i por cada iteración asegura que, en algún momento, este proceso se detendrá, es decir, **i** se usa como un contador que indica si ya hemos llegado al ultimo registro.

De igual forma, observa que después de esos caractares se agrega el caracter **\n** lo cual indica un salto de línea (enter). Esto se hace para que, al momento de escribir en el archivo el texto que estamos creando, el contenido no esté en una sola línea y se pueda entender mejor a la hora de leerlo.

De igual forma, en el texto que estamos creando podemos ver cosas como:

```
:{'nombre':'"+nombre+"'},
```

Y

```
'coordinates':["+str(longitud)+","+str(latitud)+"]
```

Estas líneas toman el valor almacenado en las variables **nombre, longitud y latitud** mencionadas anteriormente y las agregan al texto.

Pues bien, el bloque anterior debió de recorrer todo el contenido del dataframe registro por registro para ponerlo en formato GeoJSON dentro de la variable jsfile, la cual aún se encuentra en memoria. Ahora veamos qué tenemos en la variable **jsfile**. Escribe lo siguiente:

```
print(jsfile)
```

deberás ver algo similar a lo que se muestra a continuación pero con más registros:

```
var escuela = { 'type':'FeatureCollection',
'features': [
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Laura Méndez de Cuenca'},
 'geometry': {'type':'Point', 'coordinates':[-
89.6449511,21.0428409]}},
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Zamná'},
 'geometry': {'type':'Point', 'coordinates':[-
89.6255788,21.0370622]}},
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Ana Sofía Baeza García
vespertino'},
 'geometry': {'type':'Point', 'coordinates':[-89.7087103,20.996205]}}
]};
```

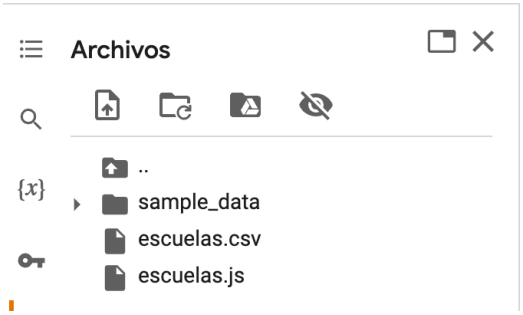
Para terminar esta parte, lo que sigue es copiar el contenido de la variable jsfile a un archivo con extensión .js

En una nueva celda de código escribe lo siguiente:

```
with open('escuelas.js', 'a') as archivo:
  archivo.write(jsfile)
archivo.close()
```

Lo que estamos haciendo es abrir un archivo con nombre ‘escuelas.js’ para agregar contenido en él, y después de abrirlo, escribir el contenido de jsfile en el archivo (archivo.write(jsfile)) y finalmente cerrar el archivo (archivo.close())

Aparentemente no hay ningún resultado, sin embargo, el archivo que acabamos de crear lo podremos ver en la sección de archivos, ¿te acuerdas?, del lado izquierdo de la pantalla de Google Colab.



De igual forma, si notas que el contenido del archivo no es el correcto, puedes eliminarlo y volverlo a crear. Para eso, pulsa sobre los tres puntos que aparecen al final del nombre del archivo y, en el menú que aparece, selecciona la opción **eliminar**

Una vez que localices el archivo tendrás que descargarlo a tu computadora haciendo click sobre los tres puntos que aparecen del lado derecho del nombre del archivo y seleccionando la opción descargar del menú.

Bien, ahora que ya tenemos los datos obtenidos desde Google Maps lo que sigue es visualizarlos en un mapa. Para ello vamos a hacer uso de una librería que se llama **leaflet.** Aquí te dejo un par de enlaces en donde puedes encontrar más información, te recomiendo que les eches un ojo.

https://en.wikipedia.org/wiki/Leaflet_(software)
https://leafletjs.com/

Como podrás ver, leaflet es una librería de javascript para desplegar mapas interactivos. En el segundo enlace puedes encontrar documentación, tutoriales y varios ejemplos.

Para esta parte de la práctica, vamos a crear una página web (.html) que haga uso de la librería leaflet para desplegar en un mapa los puntos que acabamos de obtener desde Google Maps y que están almacenados en el archivo **escuelas.js**

Lo primero es hechar una mirada al contenido del archivo

```
var escuela = { 'type':'FeatureCollection',
'features': [
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Laura Méndez de Cuenca'},
 'geometry': {'type':'Point', 'coordinates':[-
89.6449511,21.0428409]}},
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Zamná'},
 'geometry': {'type':'Point', 'coordinates':[-
89.6255788,21.0370622]}},
 {'type':'Feature',
```

```
 'properties':{'nombre':'Escuela Primaria Ana Sofía Baeza García
vespertino'},
 'geometry': {'type':'Point', 'coordinates':[-89.7087103,20.996205]}}
]};
```

Como puedes observar, el archivo comienza con la palabra **var escuela** y después una serie de datos que se encuentran contenidos entre llaves **{}** y entre corchetes **[].**

var escuela hace referencia a una variable que contiene el conjunto de datos que fueron extraidos de Google Maps. A partir de ahí comienza una serie de parejas de clave y valor, a manera de un diccionario en python. Puedes leer un poco más acerca de los diccionarios en https://ellibrodepython.com/diccionarios-en-python.

La primera pareja dice **'type':'FeatureCollection'**, es decir, comienza una colección de características.

Luego veremos

'type':'Feature',

Esto siginfica que comenzamos a describir las características del primer dato, dentro de las que encontramos como propiedades (properties), su nombre:

```
'properties':{'nombre':'Escuela Primaria Laura Méndez de Cuenca'},
```

Y después sus coordenadas.

```
'geometry': {'type':'Point', 'coordinates':[-89.6449511,21.0428409]}},
```

Esto hace referencia a la geometría, la cual se trata de un punto compuesto por un par de coordenadas.

Verás que este patrón o secuencia se repite hasta el final del contenido del archivo. Pues bien, eso es lo que queremos mostrar en un mapa. Es importante que consideres que los archivos **escuelas.js** y **mapa_escuelas.html** deben estar en el mismo directorio, es decir, en la misma carpeta.

Para crear el archivo que contiene el mapa (mapa_escuelas.html) abre un editor de textos como el block de notas, Sublime Text (https://www.sublimetext.com/3), visual studio code (https://code.visualstudio.com/download) o el de tu preferencia y haz lo siguiente:

Abre un archivo nuevo y guárdalo como mapa_escuelas.html

La primera parte del archivo mapa_escuelas.html corresponde al inicio de la página web y en la sección del encabezado, es decir, entre las etiquetas **<head>** y **</head>** se encuentran algunos enlaces hacia la librería leaflet. Esto es lo que nos va a permitir desplegar un mapa

en nuestra página. Además, puedes encontrar entre las etiquetas **<title>** y **</title>** el título que aparece en la pestaña de la página.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.8.0/dist/leaflet.css"
   integrity="sha512-
hoalWLoI8r4UszCkZ5kL8vayOGVae1oxXe/2A4AO6J9+580uKHDO3JdHb7NzwwzK5xr
/Fs0W40kiNHxM9vyTtQ=="
   crossorigin=""/>
 <script src="https://unpkg.com/leaflet@1.8.0/dist/leaflet.js"
   integrity="sha512-
BB3hKbKWOc9Ez/TAwyWxNXeoV9c1v6FIeYiBieIWkpLjauysF18NzgR1MBNBXf8/K
ABdlkX68nAhlwcDFLGPCQ=="
   crossorigin=""></script>
   <script type="text/javascript" src="https://code.jquery.com/jquery-
2.1.3.min.js"></script>
    <script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.js" data-require="leaflet@0.7.3"
data-semver="0.7.3"></script>

<title>MAPA ESCUELAS UIMQROO</title>
</head>
```

La segunda parte de la página pertenece al cuerpo de ésta, por eso se encuentra entre las etiquetas **<body>** y **</body>.** Agrega ambas al archivo mapa_escuelas.html debajo de la etiqueta </head> de la siguiente forma

```
<body>

</body>
```

Entre las etiquetas <body> y </body> escibe las siguientes líneas de código.

```
<div id="map" style="width: 100%; height: 800px; margin:auto">
<script src="escuelas.js" type="text/javascript"></script>
```

La primer hace referencia al espacio dentro de la página web donde se va a desplegar el mapa. La segunda indica que vamos a leer los datos que se encuentran en el archivo **escuelas.js**

Lo que sigue es una sección con un script (bloque de código) en javascript que indica que vamos a agregar los datos que vamos a leer del archivo al mapa a manera de una capa (layer).

```
<script>
var escuelas = L.layerGroup();
   var map = L.map('map', {
      center: [20.9857939,-89.7609942],
      zoom: 10,
      layers: [escuelas]
   });
```

Posteriormente se define la apariencia del mapa

```
var tiles = L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
      maxZoom: 19,
      attribution: '&copy; <a
href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
   }).addTo(map);
```

Ya que tenemos el diseño del mapa, la siguiente sección lee el archivo escuelas.js y, por cada característica (feature), obtiene su nombre y sus coordenadas y las agrega al mapa por medio de la función **addTo(map)** que se encuentra al final del código

```
//-------------------------------------------ESCUELAS--------------------
   function onEachFeature(feature, layer) {
      var popupContent = '<p>nombre = ' +
            feature.properties.nombre;

      if (feature.properties && feature.properties.popupContent) {
         popupContent += feature.properties.popupContent;
      }

      layer.bindPopup(popupContent).addTo(escuelas);
   }

   var escuelasLayer = L.geoJSON([escuela], {

      style: function (feature) {
         return feature.properties && feature.properties.style;
      },

      onEachFeature: onEachFeature,
   }).addTo(map);
 //----------------------------------------------------------
```

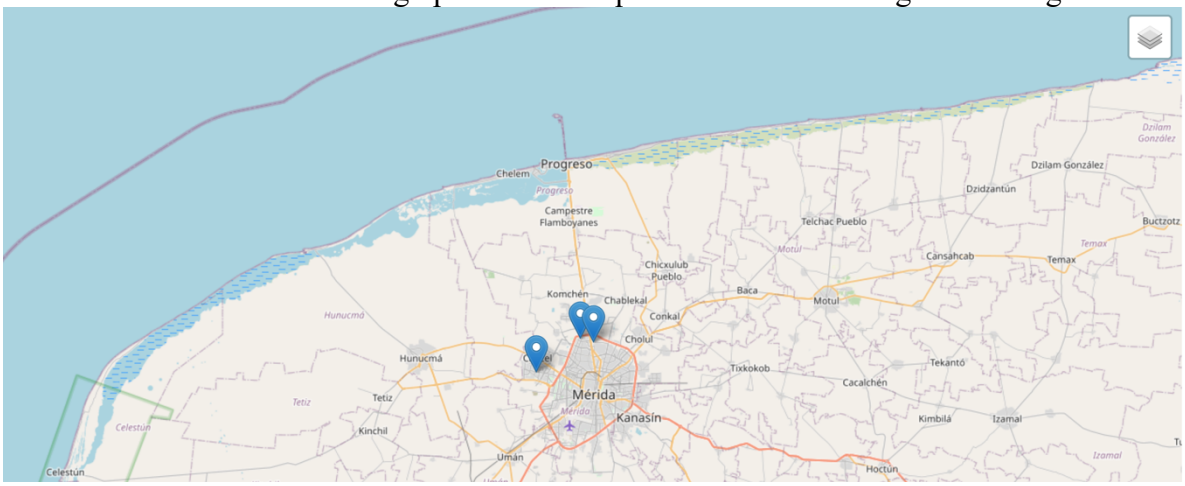
Nota que, en la línea L.geoJSON([**escuela**], la palabra que se encuentra entre corchetes es **escuela**. Esa palabra tiene que coincidir exactamente con el nombre de la variable que creamos al inicio del archivo GeoJSON

```
var escuela =
```

Hasta este punto del código, se han leído el archivo y se ha generado una capa de datos con su contenido, lo que falta ahora es visualizar esa capa de datos en el mapa.

```
var overlays = {
      'Escuelas': escuelas
   };

var layerControl = L.control.layers().addTo(map);

layerControl.addOverlay(escuelas, 'Escuelas');

</script>
</div> <!-- AQUI TERMINA EL MAPA -->

</body>
</html>
```

El resultado final deberá ser algo parecido a lo que se muestra en la siguiente imagen:



A manera de práctica, realiza lo siguiente:

1) Utiliza los datos recabados con la Aplicación móvil para crear un mapa con leaflet.

Vamos a agregar una capa de datos al mapa. Para ello, recupera el archivo direcciones.csv y conviértelo a .js de la siguiente manera.

Al código anterior agrega las siguientes celdas:

Leemos el archivo direcciones.csv y lo convertimos a un dataframe

```
df_direcciones = pd.read_csv('direcciones.csv')
```

Visualizamos el encabezado del dataframe

```
df_direcciones.head()
```

Calculamos la longitud del dataframe

```
len(df_direcciones)
```

Filtramos el dataframe original para quedarnos solo con aquellas respuestas que correspondan a 'El día'

```
df_dia = df_direcciones[(df_direcciones.resp1=="'El día'")]
df_dia
```

Calculamos el tamaño del nuevo dataframe y creamos la variable i que inica en 1

```
tamaño = len(df_dia)
i=1
tamaño
```

Recorremos el dataframe (df_dia) línea por linea para tomar los valores de las columnas resp1, longitud y latitud, con las que creamos el contenido de la variable jsfile que es de tipo GeoJson

```
jsfile = ""
jsfile = "var dia = { 'type':'FeatureCollection', \n"
jsfile = jsfile + "'features': [ \n"

for index, row in df_dia.iterrows():
  respuesta = row['resp1']
  longitud = row['longitud']
  latitud = row['latitud']
  if i<tamaño:
    jsfile = jsfile +" {'type':'Feature', \n
'properties':{'respuesta':'"+respuesta+"'}, \n 'geometry':
{'type':'Point',
'coordinates':["+str(longitud)+","+str(latitud)+"]}},\n"
    i+=1
  else:
    jsfile = jsfile +" {'type':'Feature', \n
'properties':{'respuesta':'"+respuesta+"'}, \n 'geometry':
```

```
{'type':'Point', 'coordinates':["+str(longitud)+","+str(latitud)+"]}}
\n]};\n"
```

Imprimimos el contenido de la variable para ver su contenido

```
print(jsfile)
```

Agregamos el contenido al archivo escuelas.js

```
with open('escuelas.js', 'a') as archivo:
  archivo.write(jsfile)
archivo.close()
```

Observa que ahora, el archive escuelas.js tiene el contenido de las escuelas y de las direcciones.

```
var escuela = { 'type':'FeatureCollection',
'features': [
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Laura Méndez de Cuenca'},
 'geometry': {'type':'Point', 'coordinates':[-89.6449511,21.0428409]}},
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Zamná'},
 'geometry': {'type':'Point', 'coordinates':[-89.6255788,21.0370622]}},
 {'type':'Feature',
 'properties':{'nombre':'Escuela Primaria Ana Sofía Baeza García vespertino'},
 'geometry': {'type':'Point', 'coordinates':[-89.7087103,20.996205]}}
]};
var dia = { 'type':'FeatureCollection',
'features': [
 {'type':'Feature',
 'properties':{'respuesta':'El día'},
 'geometry': {'type':'Point', 'coordinates':[-99.0827003,19.4464479]}},
 {'type':'Feature',
 'properties':{'respuesta':'El día'},
 'geometry': {'type':'Point', 'coordinates':[-99.1749605,19.4293039]}},
 {'type':'Feature',
 'properties':{'respuesta':'El día'},
 'geometry': {'type':'Point', 'coordinates':[-99.1147913,19.4300609]}},
 {'type':'Feature',
 'properties':{'respuesta':'El día'},
 'geometry': {'type':'Point', 'coordinates':[-99.1920307,19.4194815]}},
 {'type':'Feature',
 'properties':{'respuesta':'El día'},
```

```
 'geometry': {'type':'Point', 'coordinates':[-99.1891748,19.3228012]}},
 {'type':'Feature',
 'properties':{'respuesta':'El día'},
 'geometry': {'type':'Point', 'coordinates':[-99.1004137,19.4059402]}}
]};
```

De esta forma var escuela y var dia forman cada una una caoa del mapa. Ahora toca modificar el archivo html para poder visualizar ambas capas.

Para ello, vuelve a abrir el archivo escuelas.html en modo edición (no con el navegador).

Localiza la línea var escuelas = L.layerGroup(); y escribe debajo de ella la línea
var respuestas = L.layerGroup();

Localiza la línea layers: [escuelas] y cámbiala por layers: [escuelas, respuestas]

Después de la capa de escuelas escribe el siguiente código

```
//-----------------------------------------RESPUESTAS--------------------
   function onEachFeature_dia(feature, layer) {
      var popupContent = '<p>respuesta = ' +
            feature.properties.respuesta;

      if (feature.properties && feature.properties.popupContent) {
         popupContent += feature.properties.popupContent;
      }

      layer.bindPopup(popupContent).addTo(respuestas);
   }

   var diaLayer = L.geoJSON([dia], {

      style: function (feature) {
         return feature.properties && feature.properties.style;
      },
      onEachFeature: onEachFeature_dia,
   }).addTo(map);
  //-----------------------------------------------------------
```

Localiza el código:

```
var overlays = {
      'Escuelas': escuelas
   };
```

Y cámbialo por:

```
var overlays = {
        'Escuelas': escuelas,
        'Respuestas': respuestas
    };
```

Localiza la línea layerControl.addOverlay(escuelas, 'Escuelas'); y escribe debajo layerControl.addOverlay(respuestas, 'Respuestas');

Guarda el archivo y despliégalo en el navegador. Si es necesario aumenta o disminuye el zoom para ver los puntos.