Maps and Visualisation of Geographically Located Data

In this lesson we shall look at the use of maps and geographically located data.

Data with a geographic component is of huge importance in many types of investigation, and is relevant to commercial and social as well as enviromental applications.

Certain types of data are intrinsically geographical, in the sense that they directly describe the spatial and material structure of the world. This includes data regarding the actual shape of the earth's surface, its land-masses, oceans, mountains, valleys and plains. It also includes information about land type and land use and specific geographic objects: deserts, forests, orchards, buildings and cities. And we may also include regions that are defined by human conventions and legislation, such as countries, counties and national parks.

But there are many more kinds of data that, though not in themselves geographic, are linked to geographic locations --- for example, average temperatures or rainfall, populations of cities or countries, house prices, average incomes, crime levels, internet addresses. And we can often get insight into data by investigating its correspondence with geographic location. Linking data with geographic location immidiately provides a way to view it in terms of its distribution in space, and in particular in relation to a familiar map of our country or our world. This may reveal patterns in the data that were impossible to see by looking at tables of numbers. Futhermore, if we have two different types of data that are not intrinsically geographical but both are linked to geographic locations, we can use this to identify correlations that are only apparent when we consider their spatial distribution.

Map Visualisation in Jupyter

There are many packages that provide ways of visualising maps and geographically located data in Jupyter. Some of the most commonly used are:

- `geopandas` (https://geopandas.org/)
- `plotly` with `mapbox` (https://plotly.com/python/maps/)
- `folium` (https://python-visualization.github.io/folium/)
- `kepler.gl` (https://kepler.gl/)
- `geoviews` (https://geoviews.org/)
- `ipyleaflet` (https://ipyleaflet.readthedocs.io/en/latest/)

These all provide ways of displaying maps and adding graphics derived from data to map displays. Most of them also provide various ways that you can interact with the map data, such as panning, zooming, selecting and drawing.

The one we shall look at in this module is `ipyleaflet`. (The link in the list above will take you to the documentation.) Probably other choices will be more suitable in different cases, depending on what kind of mapping and map functionality you need. But `ipyleaflet` is fairly easy to use and comes with a convenient map of the world, which is can be zoomed in and out show any scale from the whole world to a street map.

Installing `ipyleaflet`

`ipyleaflet` should be very easy to install as long as you have a recent version of Anaconda. (At the time of writing, the latest version of the full Anaconda3 install was July 2020, so any version more recent than that should be fine.

- If using `conda` for package updates it is probably best to run it in a shell window. Under Linux, the `conda` program should be at `anaconda3/bin/conda` (The location of the `anaconda3` directory will depend on how you installed it.) In Windows you should have **Anaconda Prompt**

available by command search at the start menu. From there you can run `conda` commands.
  - (optional) `conda update --all`
  - `conda install -c conda-forge ipyleaflet`
- If using `pip`, you probably just need to do `pip install ipyleaflet` or possibly `pip3 install ipyleaflet` (You normally only need to use `pip3` if your main version of `pip` is for Python2. This which can be the case on systems which have Python2 built in as well as having Python3 installed afterwards.)

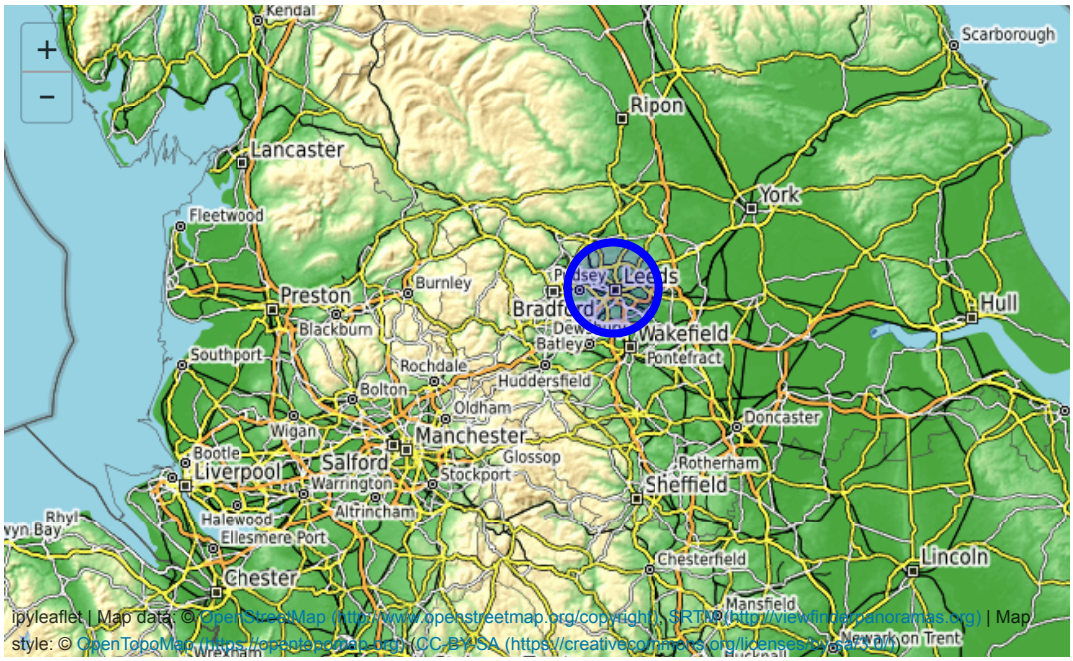
Other Info on `ipyleaflet`

- Thre is a long and quite informative tutorial on using `ipyleaflet` on YouTube (https://www.youtube.com/watch?v=PuJ_JUkahXQ)

Displaying a Map

The following code illustrates how easy it is to display a map of the world. In this case the we are using one of the maps provided, called `OpenTopoMap`. The latitude (https://en.wikipedia.org/wiki/Latitude) and longitude (https://en.wikipedia.org/wiki/Longitude) of Leeds are specified as the centre point of the map and a certain zoom level has been chosen. You can explore the map by zooming in and out and panning the map with the usual kinds of pointer movements.

In [25]:
```
from ipyleaflet import Map, basemaps, basemap_to_tiles, Circle, Polyline
LEEDS_LOC  = ( 53.8008,  -1.5491  )
WORLD_MAP = Map(basemap=basemaps.OpenTopoMap, center=LEEDS_LOC, zoom=1.5)
display(WORLD_MAP)
```



Let us now add something to the map. The easiest, but already quite informative is to put a circle on the map. We can package up the creation of a circle, setting its appearance and adding to the map into the convenient function `draw_circle_on_map`. Then it is eash to create a circle centred on Leeds.

Notice that the map has already been created and displayed by the cell above. When we run the cell below, the circle is displayed on that map (it does not create a new one). So if you have zoomed into somewhere away from Leeds, you will need to zoom out or pan the map to see the new circle around Leeds. Then you can zoom in and see Leeds in detail.

In [26]:
```
def draw_circle_on_map( amap, location,
                        radius = 10000, color="blue", fill_color=None ):
    if not fill_color:
```

```
        fill_color = color
    circle = Circle()
    circle.location = location
    circle.radius = radius
    circle.color = color
    circle.fill_color = fill_color
    amap.add_layer(circle)

draw_circle_on_map( WORLD_MAP , LEEDS_LOC )
```

Displaying Geo-Located Data on a Map

I now show a simple example of displaying some geo-located data on the world map. The example I have choosen is a **Significant Volcanic Erruptions** dataset by Stuart Tinsley on data.world. The data is and some additional information is available from here (https://data.world/stuartltinsley/volcanic-eruptions-data-set). And I have also put a copy of the CSV file on this module's data web page (https://teaching.bb-ai.net/P4DS/data/index.html).

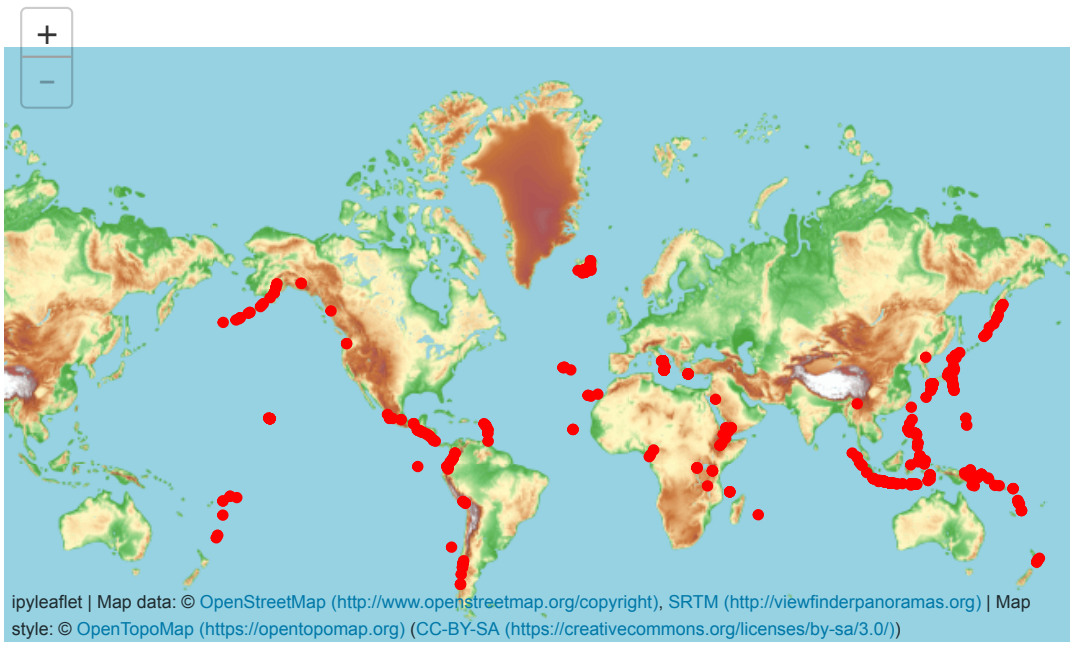
The following code also illustrates how you can load a CSV into a `pandas` `DataFrame` directly from a URL pointing to where the CSV is stored.

In [28]:
```
import pandas
erruptions_df = pandas.read_csv(
    'https://teaching.bb-ai.net/P4DS/data/volcanic_erruptions.csv')

#print(erruptions_df.columns.values)
WORLD_MAP2 = Map(basemap=basemaps.OpenTopoMap, center=LEEDS_LOC, zoom=1.5)
display(WORLD_MAP2)

erruptions_df.head()

for _, vals in erruptions_df.iterrows():
    lat = vals["latitude"]
    long = vals["longitude"]
    draw_circle_on_map( WORLD_MAP2, (lat,long), color="red")
```



Explanation of the Volcano Map Display

Here we see that the powerful volcanoes that have occurred during the past few thousand years are not randomly distributed over the Earth's surface. Their locations are actually along certain lines, which in

most cases also correspond to mountain ranges or chains of islands. This finding can be explained by the plate techtonics (https://en.wikipedia.org/wiki/Plate_tectonics) theory of the Earth's lithosphere (https://en.wikipedia.org/wiki/Lithosphere). According to that theory, the Earth's surface is broken up into a number of very large plates, which have cracks between them. It is along these cracks that earthquakes and volcanoes tend to occur.