basic_plot_5

February 3, 2023

[ ]:
```
# import primary modules
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plot
```

[ ]:
```
# import folium
import folium as fm
```

[ ]:
```
# define the world map
world_map = fm.Map()

# display world map
world_map
```

[ ]:
```
# define world map centered around Canada with low zoom level
world_map = fm.Map (location = [56.130, -106.35], zoom_start = 3)

# show world map
world_map
```

[ ]:
```
# create a map of Austin with high zoom level
world_map = fm.Map (location = [30.2672, -97.743], zoom_start = 8)

# display map
world_map
```

Stamen Toner Maps These are high-contrast black and white maps. They are perfect for data mashups and exploring river meanders and coastal zones.

[ ]:
```
# create a Stamen Toner map of the world centered around Canada
world_map = fm.Map(location = [56.130, -106.35], zoom_start = 4, tiles =␣
↪'Stamen Toner')

# display map
world_map
```

Stamen Terrain Maps These are maps that feature hill shading and natural vegetation colors. They showcase advanced labeling and linework generalization of dual-carriageway roads.

[ ]:
```
# create a Stamen Toner map of the world centered around Canada
world_map = fm.Map(location = [56.130, -106.35], zoom_start = 4, tiles =␣
 ↪'Stamen Terrain')

# display map
world_map
```

[ ]:
```
# create a Stamen Toner map of the world centered around Austin, TX
world_map = fm.Map(location = [30.2672, -97.7431], zoom_start = 4, tiles =␣
 ↪'Stamen Terrain')

# display map
world_map
```

Maps with Markers

- IncidntNum: Incident Number
- Category: Category of crime or incident
- Descript: Description of the crime or incident
- DayOfWeek: The day of week on which the incident occurred
- Date: The Date on which the incident occurred
- Time: The time of day on which the incident occurred
- PdDistrict: The police department district
- Resolution: The resolution of the crime in terms whether the perpetrator was arrested or not
- Address: The closest address to where the incident took place
- X: The longitude value of the crime location
- Y: The latitude value of the crime location
- Location: A tuple of the latitude and the longitude values
- PdId: The police department ID

[ ]:
```
# read crime data of San Francisco
df_crime = pd.read_csv ('SF_Crime_2016.csv')

# check the size of the data frame
print (df_crime.shape)

# check the head
df_crime.head()
```

[ ]:
```
# get the first 100 crimes in the df_crime dataframe
limit = 100
df_crime = df_crime.iloc[0:limit, :]
```

[ ]:
```
# San Francisco latitude and longitude values
sf_lat = 37.77
sf_long = -122.42
```

[ ]:
```
# create map and display it
sf_map = fm.Map(location = [sf_lat, sf_long], zoom_start = 12)

# display the map of San Francisco
sf_map
```

Superimpose the locations of the crimes onto the map - the way to do that in Folium is to create a feature group with its own features and style and then add it to the sf_map.

[ ]:
```
# instantiate a feature group for the crimes in the dataframe
crimes = fm.map.FeatureGroup()

# loop through the 100 crimes and add each to the incidents feature group
for lat, long, in zip(df_crime.Y, df_crime.X):
    crimes.add_child(
        fm.features.CircleMarker(
            [lat, long],
            radius=5, # define how big you want the circle markers to be
            color='yellow',
            fill=True,
            fill_color='blue',
            fill_opacity=0.6
        )
    )

# add pop-up text to each marker on the map
latitudes = list(df_crime.Y)
longitudes = list(df_crime.X)
labels = list(df_crime.Category)

for lat, long, label in zip(latitudes, longitudes, labels):
    fm.Marker([lat, long], popup=label).add_to(sf_map)

# add crimes to map
sf_map.add_child(crimes)
```

If the map is congested with markers a simple solution is to remove these location markers and just add the text to the circle markers themselves.

[ ]:
```
# create map and display it
sf_map = fm.Map(location = [sf_lat, sf_long], zoom_start = 12)

# loop through the 100 crimes and add each to the map
for lat, long, label in zip(df_crime.Y, df_crime.X, df_crime.Category):
```

```
    fm.features.CircleMarker(
        [lat, long],
        radius=5, # define how big you want the circle markers to be
        color='yellow',
        fill=True,
        popup=label,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(sf_map)

# show map
sf_map
```

The proper remedy is to group the markers into different clusters. Each cluster is then represented by the number of crimes in each neighborhood. These clusters can be thought of as pockets of San Francisco which you can then analyze separately.

To implement this, we start off by instantiating a MarkerCluster object and adding all the data points in the dataframe to this object.

[ ]:
```
from folium import plugins

# let's start again with a clean copy of the map of San Francisco
sf_map = fm.Map(location = [sf_lat, sf_long], zoom_start = 12)

# instantiate a mark cluster object for the incidents in the dataframe
crimes = plugins.MarkerCluster().add_to(sf_map)

# loop through the dataframe and add each data point to the mark cluster
for lat, long, label, in zip(df_crime.Y, df_crime.X, df_crime.Category):
    fm.Marker(
        location=[lat, long],
        icon=None,
        popup=label,
    ).add_to(crimes)

# display map
sf_map
```