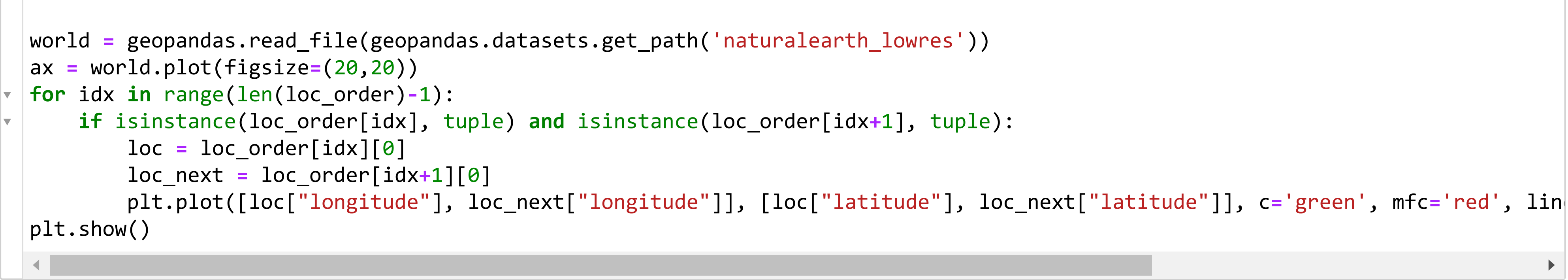
Homework Assignment 8

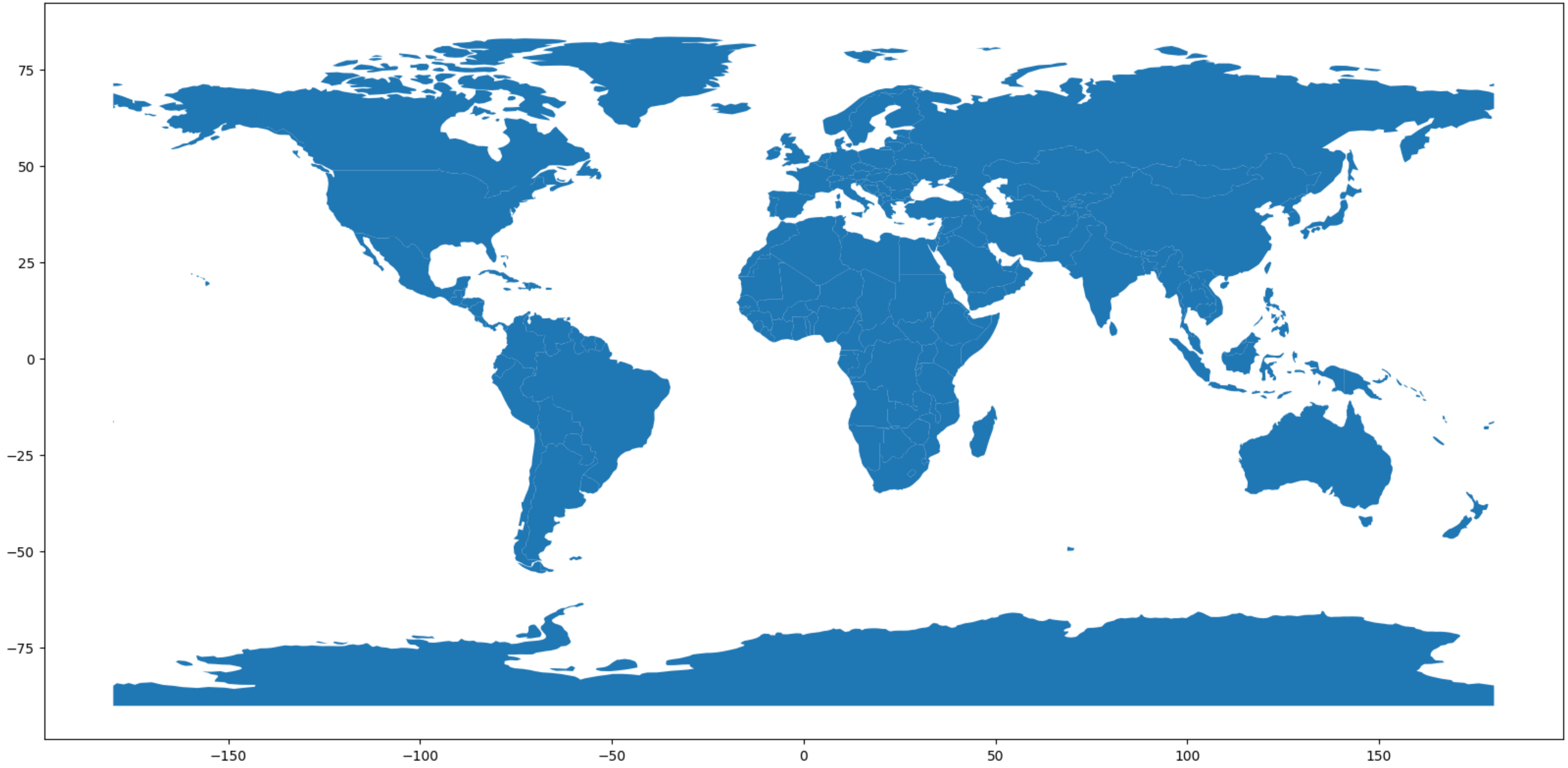
Question 1: Select a book or a movie's subtitle, and find the locations that appear in the book/movie. Then, draw these locations on a map (at least 20 locations) (30pt). Connect the discovered locations with lines according to their order of appearance in the book. For example, if location B appeared immediately after location A, then draw a line connecting locations A and B (20pt).

In [1]:

```
import os
import re
import json
import spacy
import geopandas
import itertools
import math
import warnings
from matplotlib import pyplot as plt
from geopy.geocoders import Nominatim

warnings.filterwarnings('ignore')

with open("The Hunger Games.txt", "r", encoding="utf-8") as file:
    text = file.read()

nlp = spacy.load('en_core_web_lg')

# Method to get locations from text
def get_locations_from_text(text):
    loc_dict= {}
    doc = nlp(text)
    for entity in doc.ents:
        if entity.label_ in {'GPE'}:
            loc = entity.text.lower().strip()
            loc_dict[loc] = loc_dict.get(loc, 0) + 1
    return loc_dict

locations_dict = get_locations_from_text(text)
locations = sorted(locations_dict.items(), key=lambda item: item[1], reverse=True)

geolocator = Nominatim(user_agent="Data Science Education App")
real_locations= []
for loc, appearance in locations:
    try:
        real_location = geolocator.geocode(loc)
        real_locations.append(dict(location=loc, appearance=appearance, latitude=real_location.latitude,longitude=real_locatio
    except:
        continue

loc_order = []
text_lower = text.lower()
for i in real_locations:
    loc_order += [(i, j.start(0)) for j in re.finditer(i["location"], text_lower)]
loc_order = sorted(loc_order, key=lambda x: x[1])
loc_order = [k for k,g in itertools.groupby(loc_order, lambda x: x[0])]
```



```
world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres'))
ax = world.plot(figsize=(20,20))
for idx in range(len(loc_order)-1):
    if isinstance(loc_order[idx], tuple) and isinstance(loc_order[idx+1], tuple):
        loc = loc_order[idx][0]
        loc_next = loc_order[idx+1][0]
        plt.plot([loc["longitude"], loc_next["longitude"]], [loc["latitude"], loc_next["latitude"]], c='green', mfc='red', lin
plt.show()
```



```
In [2]:  locations
```

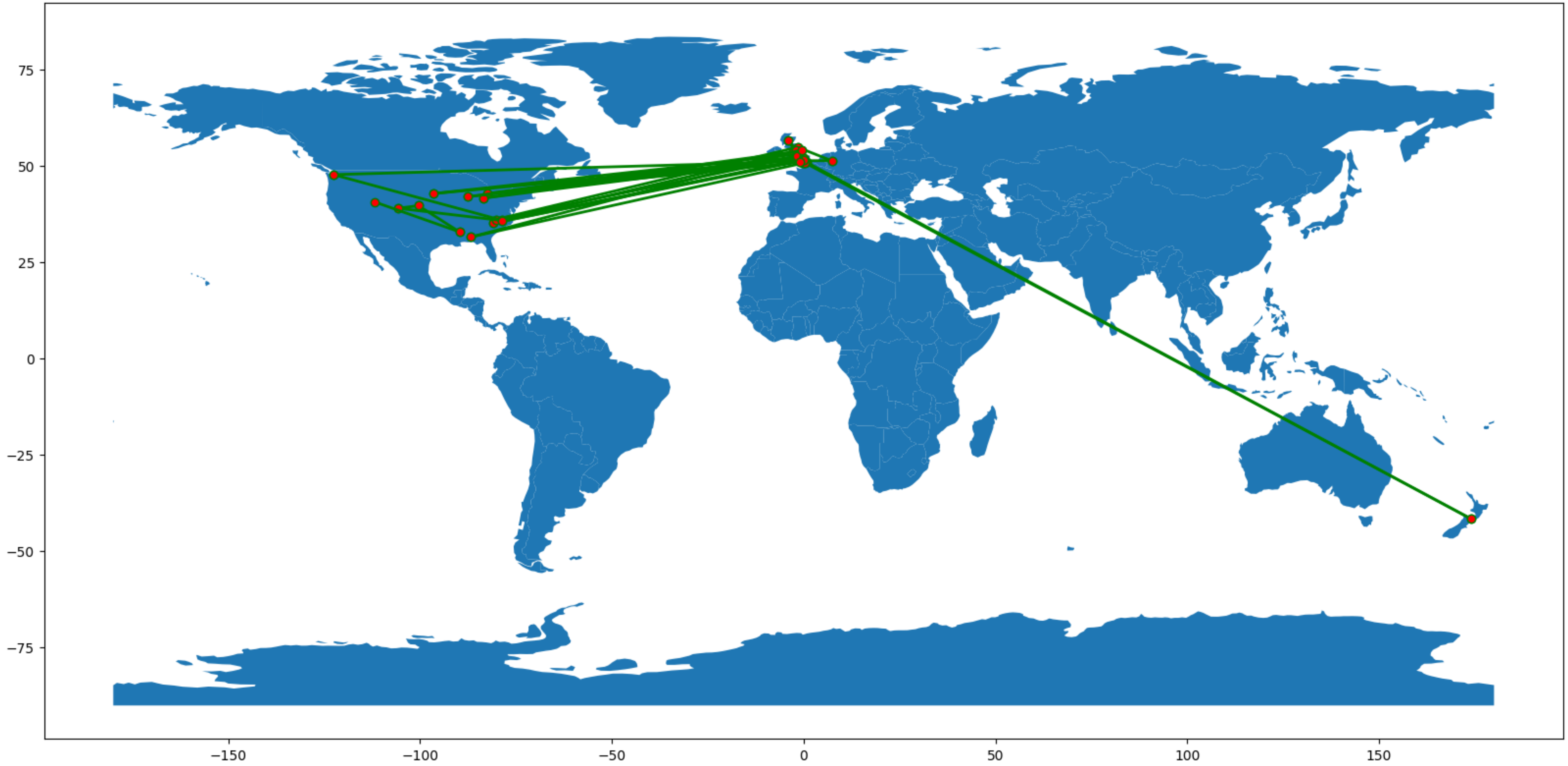
```
Out[2]: [('venia', 9),
         ('the city circle', 6),
         ('haymitch', 6),
         ('rue', 4),
         ('panem', 2),
         ('sixteens', 1),
         ('iftheyre', 1),
         ('octavia', 1),
         ('katniss', 1),
         ('cinna', 1),
         ('district eleven', 1),
         ('nick', 1),
         ('district 1', 1),
         ('district twelve', 1)]
```

There are no real locations in this book, so I've attached another book. Just kept this one to show that it works (:

In [3]:

```
import os
import re
import json
import spacy
import geopandas
import itertools
import math
import warnings
from matplotlib import pyplot as plt
from geopy.geocoders import Nominatim

warnings.filterwarnings('ignore')

with open("pride_and_prejudice.txt", "r", encoding="latin-1") as file:
    text = file.read()

nlp = spacy.load('en_core_web_lg')

# Method to get locations from text
def get_locations_from_text(text):
    stop_words = set([
        "i'll", "you're", "wasn't", "don't", "who?", "too?", "see?", "sir.", "ain't",
        "didn't", "run?", "wonderin'", "haven't", "nothin'", "what'll", "please?", "mayella!", "know\x97long time.",
        "school?", "right?\x94", "one", "nigger.", "it's right.", "washington", "the united states", "saying.", "present\x97at
        "world,", "ant.", "egypt", "america", "you's", "maycomb county", "the virgin forests", "face.\x94", "got.\x94", "think
        "books.\x94\n\natticus"
    ])
    loc_dict = {}
    doc = nlp(text)
    for entity in doc.ents:
        if entity.label_ == 'GPE' and entity.text.lower() not in stop_words:
            loc = entity.text.lower().strip()
            loc_dict[loc] = loc_dict.get(loc, 0) + 1
    return loc_dict

locations_dict = get_locations_from_text(text)
locations = sorted(locations_dict.items(), key=lambda item: item[1], reverse=True)

geolocator = Nominatim(user_agent="Data Science Education App")
real_locations = []
for loc, appearance in locations:
    try:
        real_location = geolocator.geocode(loc)
        real_locations.append(dict(location=loc, appearance=appearance, latitude=real_location.latitude, longitude=real_locati
    except:
        continue
```

```
world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres'))
ax = world.plot(figsize=(20,20))
for idx in range(len(real_locations)-1):
    loc = real_locations[idx]
    loc_next = real_locations[idx+1]
    plt.plot([loc["longitude"], loc_next["longitude"]], [loc["latitude"], loc_next["latitude"]], c='green', mfc='red', linewid
plt.show()
```



In [4]:

```
locations
```

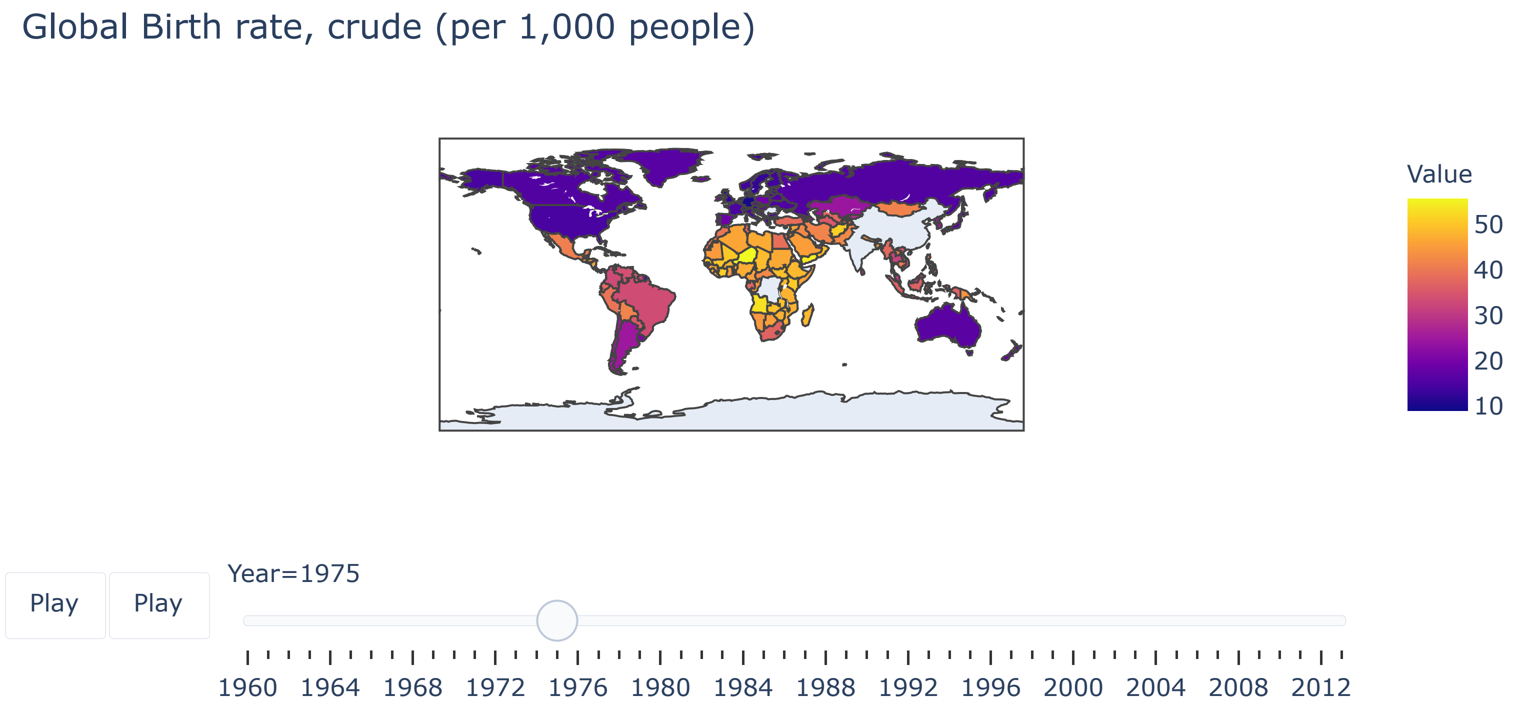
```
Out[4]: [('london', 55),
         ('charlotte', 42),
         ('hertfordshire', 41),
         ('netherfield', 34),
         ('derbyshire', 24),
         ('brighton', 21),
         ('georgiana', 13),
         ('bingley', 12),
         ('lucases', 9),
         ('lambton', 9),
         ('kent', 8),
         ('scotland', 8),
         ('england', 4),
         ('hunsford', 4),
         ('newcastle', 4),
         ('u.s.', 3),
         ('phillipses', 2),
         ('longbourn', 2),
         ('pemberley', 2),
         ('the united\nstates', 2),
         ('westerham', 1),
         ('york', 1),
         ('charlotte\nlucas', 1),
         ('cambridge', 1),
         ('liverpool', 1),
         ('matlock', 1),
         ('chatsworth', 1),
         ('oxford', 1),
         ('blenheim', 1),
         ('warwick', 1),
         ('kenilworth', 1),
         ('birmingham', 1),
         ('rosings park', 1),
         ('barnet', 1),
         ('eastbourne', 1),
         ('great britain', 1),
         ('gracechurch street', 1),
         ('gracechurch', 1),
         ('kympton', 1),
         ('scarborough', 1),
         ('meryton', 1),
         ('park', 1),
         ('united states', 1),
         ('1.e.8', 1),
         ('mississippi', 1),
         ('salt lake city', 1)]
```

Question 2: Select a country's' statistic from the [World Development Indicators dataset (https://www.kaggle.com/datasets/kaggle/world-development-indicators)](https://www.kaggle.com/datasets/kaggle/world-development-indicators) (Please notice there are several files in the dataset, such as *Indicators.csv*). Then, create a choropleth map displaying how the selected statistics changed over time (15pt)

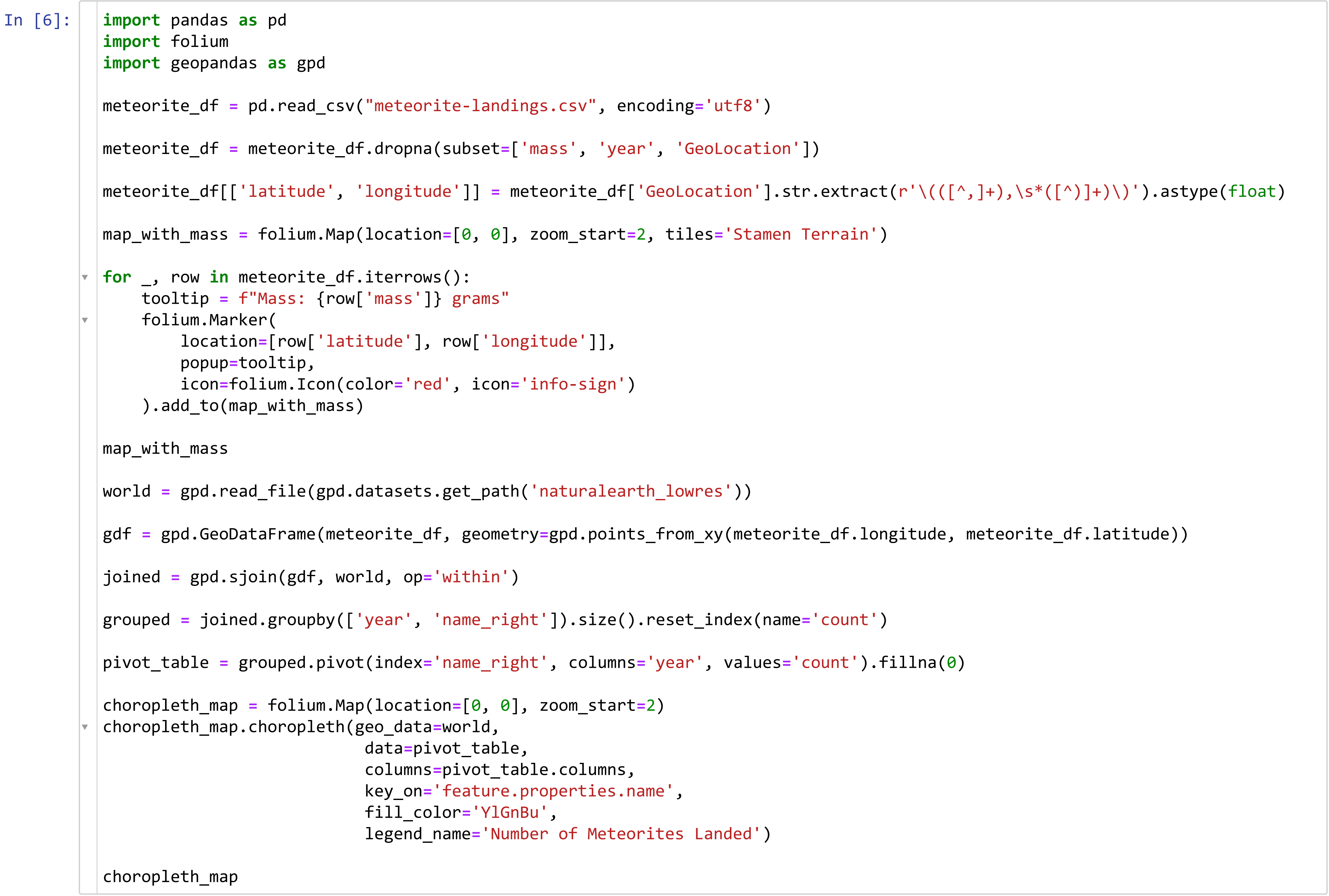
Bonus: Create a short animation that displays how the chosen statistics changed over time (15pt)

In [5]:

```
import pandas as pd
import plotly.express as px

indicators_df = pd.read_csv("Indicators.csv", encoding="utf8")

indicator_name = "Birth rate, crude (per 1,000 people)"
filter_indicators_df = indicators_df[indicators_df.IndicatorName == indicator_name]
filter_indicators_df = filter_indicators_df[~filter_indicators_df.CountryCode.isin(['CHN', 'IND'])]


fig = px.choropleth(filter_indicators_df,
                    locations="CountryCode",
                    locationmode="ISO-3",
                    color="Value",
                    animation_frame="Year",
                    width=800,
                    height=400,
                    title=f"Global {indicator_name}")

# Add Animation (Bonus)
fig.update_layout(updatemenus=[dict(type="buttons", buttons=[dict(label="Play",
                                                                  method="animate", args=[None, {"frame":
                                                            {"duration": 500, "redraw": True}, "fromcurrent": True,
                                                                                                 "mode": "immediate"}])])])

fig.show() #### NOTICE THAT THIS IS AN INTERATCTIVE FIGURE
```



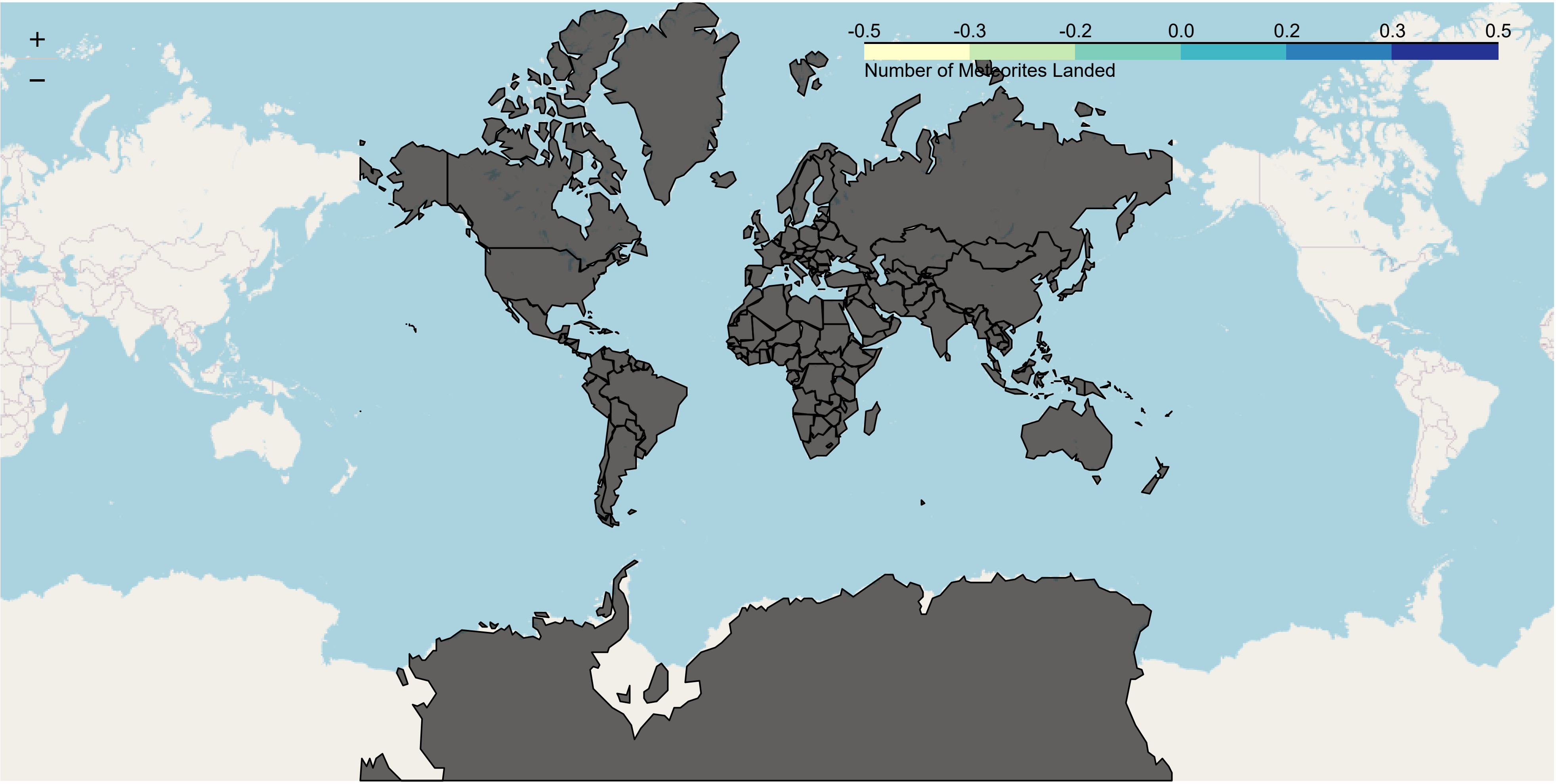
Question 3: Use data from the Meteorite Landings dataset (https://www.kaggle.com/datasets/nasa/meteorite-landings). Create maps that present where the Meteorites landed and their mass. Additionally, draw choropleth map that give information how many different meteorites landed in each country and how it changes over the years (35pt).



In [6]:

```python
import pandas as pd
import folium
import geopandas as gpd

meteorite_df = pd.read_csv("meteorite-landings.csv", encoding='utf8')

meteorite_df = meteorite_df.dropna(subset=['mass', 'year', 'GeoLocation'])

meteorite_df[['latitude', 'longitude']] = meteorite_df['GeoLocation'].str.extract(r'\(([^,]+),\s*([^)]+)\)').astype(float)

map_with_mass = folium.Map(location=[0, 0], zoom_start=2, tiles='Stamen Terrain')

for _, row in meteorite_df.iterrows():
    tooltip = f"Mass: {row['mass']} grams"
    folium.Marker(
        location=[row['latitude'], row['longitude']],
        popup=tooltip,
        icon=folium.Icon(color='red', icon='info-sign')
    ).add_to(map_with_mass)

map_with_mass

world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

gdf = gpd.GeoDataFrame(meteorite_df, geometry=gpd.points_from_xy(meteorite_df.longitude, meteorite_df.latitude))

joined = gpd.sjoin(gdf, world, op='within')

grouped = joined.groupby(['year', 'name_right']).size().reset_index(name='count')

pivot_table = grouped.pivot(index='name_right', columns='year', values='count').fillna(0)

choropleth_map = folium.Map(location=[0, 0], zoom_start=2)
choropleth_map.choropleth(geo_data=world,
                           data=pivot_table,
                           columns=pivot_table.columns,
                           key_on='feature.properties.name',
                           fill_color='YlGnBu',
                           legend_name='Number of Meteorites Landed')

choropleth_map
```

Out[6]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [7]:

```python
import folium
import plotly.express as px

meteorite_df = pd.read_csv("meteorite-landings.csv")

meteorite_df = meteorite_df.dropna(subset=['mass', 'year', 'GeoLocation'])

map_mass = folium.Map(location=[0, 0], zoom_start=2)

for idx, row in meteorite_df.iterrows():
    tooltip = f"Mass: {row['mass']} grams"
    folium.Marker(
        location=[row['reclat'], row['reclong']],
        popup=tooltip,
        icon=folium.Icon(color='red', icon='info-sign')
    ).add_to(map_mass)

map_mass.save("meteorite_landings_map.html")

meteorite_df['country'] = meteorite_df['GeoLocation'].apply(lambda x: x.split(',')[0].strip())

meteorite_df['year'] = pd.to_datetime(meteorite_df['year']).dt.year

meteorite_count_df = meteorite_df.groupby(['year', 'country']).size().reset_index(name='meteorite_count')

fig = px.choropleth(meteorite_count_df,
                    locations="country",
                    locationmode="country names",
                    color="meteorite_count",
                    animation_frame="year",
                    range_color=[0, meteorite_count_df['meteorite_count'].max()],
                    title="Meteorite Landings by Country Over Time"
                   )

fig.show()
```

Meteorite Landings by Country Over Time