**1. Using the GeoPandas Library**

The Python library GeoPandas makes it easy to work with geospatial data, including polygon geometries for countries.

**Install Required Libraries**

bash

Copy code

pip install geopandas

**Example Code**

python

Copy code

import geopandas as gpd

# Load a world map dataset

world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

# Display the first few rows of the dataset

print(world.head())

# Filter for a specific country (e.g., Algeria)

algeria = world[world['name'] == 'Algeria']

# Access polygon coordinates of Algeria

print(algeria.geometry.values[0]) # This will print the polygon geometry

**Output:** The geometry column contains shapely.geometry objects, which represent the country’s polygon. For example:

bash

Copy code

POLYGON ((-8.6844 27.3953, -8.6656 27.5895, ..., -8.6844 27.3953))

You can extract and manipulate these polygons as needed.

**2. Using GeoJSON Files**

GeoJSON files are commonly used to store geospatial data, including polygons for country boundaries.

**Steps:**

1. Download a GeoJSON file for countries:
   * Use resources like [Natural Earth](https://www.naturalearthdata.com/) or [GeoJSON Data Sources](https://github.com/datasets/geo-countries).
2. Parse the file using Python:

python

Copy code

import json

import geopandas as gpd

# Load GeoJSON data

with open('countries.geojson', 'r') as f:

data = json.load(f)

# Extract features

for feature in data['features']:

country\_name = feature['properties']['name']

geometry = feature['geometry'] # Contains the polygon/multi-polygon

print(country\_name, geometry)

**3. Using the shapefile Module**

You can use shapefiles (.shp), a common geospatial data format, to get country polygons.

**Install Required Libraries**

bash

Copy code

pip install pyshp shapely

**Example Code**

python

Copy code

import shapefile

from shapely.geometry import shape

# Load a shapefile

sf = shapefile.Reader("countries.shp")

# Access shapes and attributes

for record, geom in zip(sf.records(), sf.shapes()):

country\_name = record['name'] # Replace with the attribute for the country name

polygon = shape(geom.\_\_geo\_interface\_\_) # Converts to shapely geometry

print(country\_name, polygon)

**4. Using APIs (e.g., GADM, Natural Earth)**

Some APIs or services provide downloadable country polygon data:

* [GADM](https://gadm.org/): Offers high-resolution country shapes.
* [Natural Earth](https://www.naturalearthdata.com/): Provides free-to-use shapefiles and GeoJSON for countries.
* Use tools like the Google Maps Geocoding API or OpenStreetMap via libraries like osmnx to fetch polygon data.

**Example Using OpenStreetMap:**

python

Copy code

import osmnx as ox

# Get polygon for a specific country

country\_name = "Algeria"

country\_polygon = ox.geocode\_to\_gdf(country\_name)

print(country\_polygon)

**Key Considerations:**

1. **Resolution**: Higher-resolution datasets provide more detailed polygons but are larger and computationally heavier.
2. **Licensing**: Ensure you comply with the licensing terms of the data source.
3. **Projections**: Geospatial data may use different projections. Use libraries like GeoPandas to handle these seamlessly.

Let me know which approach you'd like to explore further or if you'd like detailed help with any specific method!

**You said:**

Given a csv file containing the following columns: Country\_code,Country\_name,Latitude,Longitude 1010,Algeria,28.0000272,2.9999825 1020,Angola,-11.8775768,17.5691241 1025,Benin,9.5293472,2.2584408 1030,Botswana,-23.1681782,24.5928742 1035,Burkina Faso,12.0753083,-1.6880314 1040,Burundi,-3.426449,29.9324519 write the code to obtain the polygon coordinates surrounding each country.

**ChatGPT said:**



To obtain the polygon coordinates for each country based on the provided CSV file, you can use geospatial libraries like GeoPandas and leverage geospatial datasets, such as the Natural Earth dataset. Here's the step-by-step code:

**Step 1: Install Required Libraries**

If not already installed, install the necessary libraries:

bash

Copy code

pip install geopandas

**Step 2: Code to Extract Polygon Coordinates**

python

Copy code

import pandas as pd

import geopandas as gpd

# Load the CSV file

csv\_file = 'countries\_with\_coordinates.csv' # Replace with the path to your CSV file

df = pd.read\_csv(csv\_file)

# Load a world map dataset with country polygons

world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

# Merge the CSV data with the world geospatial data

# Match by 'Country\_name' from the CSV and 'name' from the world dataset

geo\_df = pd.merge(

df,

world,

left\_on='Country\_name',

right\_on='name',

how='left'

)

# Check for missing polygons

missing\_polygons = geo\_df[geo\_df['geometry'].isnull()]

if not missing\_polygons.empty:

print("Missing polygons for the following countries:")

print(missing\_polygons[['Country\_name']])

# Extract polygon coordinates for each country

geo\_df['Polygon\_Coordinates'] = geo\_df['geometry'].apply(

lambda geom: geom.exterior.coords[:] if geom else None

)

# Display the DataFrame with polygon coordinates

print(geo\_df[['Country\_name', 'Polygon\_Coordinates']])

# Optionally, save the results to a new CSV file

geo\_df[['Country\_code', 'Country\_name', 'Latitude', 'Longitude', 'Polygon\_Coordinates']].to\_csv(

'countries\_with\_polygons.csv',

index=False

)

**Explanation of the Code:**

1. **Load CSV and Geospatial Data**: The CSV contains country names, and the geospatial dataset (from GeoPandas) provides polygon information.
2. **Merge the Data**: Match the countries in the CSV file to the polygons in the geospatial dataset using a common column (e.g., Country\_name in the CSV and name in the geospatial data).
3. **Handle Missing Polygons**: Identify any countries in the CSV that are not found in the geospatial dataset.
4. **Extract Polygon Coordinates**: Use the geometry column to extract polygon coordinates for each country.
5. **Save the Results**: Optionally, save the updated DataFrame, including polygon coordinates, to a new CSV file.

**Example Output:**

The Polygon\_Coordinates column will look like this:

css

Copy code

Country\_name Polygon\_Coordinates

Algeria [(-8.6844, 27.3953), (-8.6656, 27.5895), ...]

Angola [(12.5001, -5.6843), (12.4997, -5.6840), ...]

Benin [(1.8652, 11.1423), (1.8667, 11.1374), ...]

...

Let me know if you encounter any issues or need further clarification!

**You said:**

The following error message was obtained: The geopandas.dataset has been deprecated and was removed in GeoPandas 1.0. You can get the original 'naturalearth\_lowres' data from https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.

**ChatGPT said:**



The error indicates that the geopandas.datasets.get\_path method for accessing sample datasets like naturalearth\_lowres has been deprecated and removed. Instead, you need to manually download the dataset and load it using GeoPandas.

Here’s how to fix this:

**Step 1: Download the Dataset**

1. Visit the [Natural Earth data page](https://www.naturalearthdata.com/downloads/110m-cultural-vectors/).
2. Download the **"Admin 0 - Countries" shapefile** under the "Cultural" section.
   * File name: ne\_110m\_admin\_0\_countries.zip.

**Step 2: Load the Dataset in GeoPandas**

Extract the downloaded .zip file, which will include files like ne\_110m\_admin\_0\_countries.shp. Use the .shp file to load the data.

**Updated Code**

python

Copy code

import pandas as pd

import geopandas as gpd

# Load the CSV file

csv\_file = 'countries\_with\_coordinates.csv' # Replace with the path to your CSV file

df = pd.read\_csv(csv\_file)

# Load the Natural Earth shapefile

shapefile\_path = 'path/to/ne\_110m\_admin\_0\_countries.shp' # Replace with the actual path

world = gpd.read\_file(shapefile\_path)

# Merge the CSV data with the world geospatial data

# Match by 'Country\_name' from the CSV and 'NAME' from the shapefile

geo\_df = pd.merge(

df,

world,

left\_on='Country\_name',

right\_on='NAME',

how='left'

)

# Check for missing polygons

missing\_polygons = geo\_df[geo\_df['geometry'].isnull()]

if not missing\_polygons.empty:

print("Missing polygons for the following countries:")

print(missing\_polygons[['Country\_name']])

# Extract polygon coordinates for each country

geo\_df['Polygon\_Coordinates'] = geo\_df['geometry'].apply(

lambda geom: geom.exterior.coords[:] if geom and geom.type == 'Polygon' else None

)

# Display the DataFrame with polygon coordinates

print(geo\_df[['Country\_name', 'Polygon\_Coordinates']])

# Optionally, save the results to a new CSV file

geo\_df[['Country\_code', 'Country\_name', 'Latitude', 'Longitude', 'Polygon\_Coordinates']].to\_csv(

'countries\_with\_polygons.csv',

index=False

)

**Notes:**

1. **Shapefile Column Names**: Ensure you use the correct column names in the shapefile. For Natural Earth, country names are typically under the NAME column.
2. **Handle Multipolygons**: If a country has multiple polygons (e.g., island nations), you might need additional logic to handle MultiPolygon geometries. Update the lambda function to account for this:

python

Copy code

geo\_df['Polygon\_Coordinates'] = geo\_df['geometry'].apply(

lambda geom: [list(poly.exterior.coords) for poly in geom.geoms] if geom and geom.type == 'MultiPolygon'

else (geom.exterior.coords[:] if geom and geom.type == 'Polygon' else None)

)

**Key Links:**

* [Natural Earth Download Page](https://www.naturalearthdata.com/downloads/110m-cultural-vectors/)
* [GeoPandas Documentation](https://geopandas.org/)

Let me know if you need additional help!

**You said:**

Got the following error message : 'MultiPolygon' object has no attribute 'exterior'

**ChatGPT said:**



The error indicates that the code is attempting to access the .exterior attribute of a MultiPolygon object, which doesn’t have this attribute because MultiPolygon is a collection of multiple Polygon objects. To handle MultiPolygon geometries, you need to iterate over its constituent Polygon objects.

Here’s the updated code to handle both Polygon and MultiPolygon objects:

**Updated Code**

python

Copy code

# Extract polygon coordinates for each country

def get\_polygon\_coordinates(geometry):

if geometry is None:

return None

if geometry.type == 'Polygon':

# Return coordinates of a single Polygon

return list(geometry.exterior.coords)

elif geometry.type == 'MultiPolygon':

# Return coordinates of all Polygons in a MultiPolygon

return [list(poly.exterior.coords) for poly in geometry.geoms]

else:

return None

# Apply the function to the geometry column

geo\_df['Polygon\_Coordinates'] = geo\_df['geometry'].apply(get\_polygon\_coordinates)

# Display the DataFrame with polygon coordinates

print(geo\_df[['Country\_name', 'Polygon\_Coordinates']])

**Explanation:**

1. **Function to Handle Geometry Types**:
   * If the geometry is None, return None.
   * If the geometry is a Polygon, extract the coordinates using .exterior.coords.
   * If the geometry is a MultiPolygon, iterate over its .geoms (a list of Polygon objects) and extract the coordinates for each.
   * For other types of geometries (if present, e.g., Point, LineString), return None.
2. **Iterate over MultiPolygon Components**:
   * geometry.geoms is a list of Polygon objects within a MultiPolygon.
3. **Apply Function to Geometry Column**:
   * Use .apply() to process each row of the geometry column and store the results in a new column.

**Example Output:**

For a Polygon:

makefile

Copy code

Country\_name: Algeria

Polygon\_Coordinates: [(-8.6844, 27.3953), (-8.6656, 27.5895), ...]

For a MultiPolygon (e.g., a country with islands):

makefile

Copy code

Country\_name: Indonesia

Polygon\_Coordinates: [

[(-8.6844, 27.3953), (-8.6656, 27.5895), ...], # First Polygon

[(120.0, -5.0), (121.0, -5.5), ...] # Second Polygon

]

Let me know if this resolves the issue or if you encounter any further errors!

**You said:**

Given the above geo\_df[['Country\_code', 'Country\_name', 'Latitude', 'Longitude', 'Polygon\_Coordinates']], write code to display the countries on the world map

**ChatGPT said:**



To display the countries on a world map, you can use the matplotlib and geopandas libraries to plot the geo\_df GeoDataFrame, which includes the country polygons and their coordinates. Here's the code:

**Install Required Libraries**

Ensure you have the necessary libraries installed:

bash

Copy code

pip install geopandas matplotlib

**Code to Plot Countries on the World Map**

python

Copy code

import geopandas as gpd

import matplotlib.pyplot as plt

# Convert the DataFrame to a GeoDataFrame if not already done

geo\_df = gpd.GeoDataFrame(

geo\_df,

geometry=geo\_df['geometry'],

crs="EPSG:4326" # Ensure the correct coordinate reference system

)

# Create a base world map

world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

# Plot the base map

fig, ax = plt.subplots(figsize=(15, 10))

world.plot(ax=ax, color='lightgrey', edgecolor='black')

# Overlay the countries from geo\_df

geo\_df.plot(

ax=ax,

color='blue',

edgecolor='black',

alpha=0.5, # Transparency for overlay

legend=True

)

# Add annotations for each country

for idx, row in geo\_df.iterrows():

plt.text(

row['Longitude'],

row['Latitude'],

row['Country\_name'],

fontsize=8,

ha='center',

color='darkred'

)

# Add title and labels

plt.title("Countries on the World Map", fontsize=16)

plt.xlabel("Longitude", fontsize=12)

plt.ylabel("Latitude", fontsize=12)

# Show the plot

plt.show()

**Explanation of the Code:**

1. **Base World Map**:
   * Use the Natural Earth dataset to create a light grey base map with world.plot.
2. **Overlay Country Polygons**:
   * Use geo\_df.plot to overlay the country polygons (in blue) with some transparency (alpha=0.5).
3. **Add Country Annotations**:
   * Loop through the geo\_df rows and add country names at their Longitude and Latitude coordinates using plt.text.
4. **Adjust Plot Appearance**:
   * Add a title and labels for the axes for better readability.

**Example Output**

The plot will display:

* A light grey world map as the base.
* Blue polygons representing the countries in your geo\_df.
* Country names annotated at the latitude and longitude points.

Let me know if you'd like to add any enhancements, such as custom styling or interactivity!