Instruction Document: Handling and Visualizing Geospatial Data using GeoPandas

Introduction

This guide covers essential operations using GeoPandas for handling and visualizing geospatial data. You'll learn how to load geographic datasets, perform spatial operations, and create visualizations of geographic features like garbage collection zones.

---

1. Installing Required Libraries

Ensure you have Python installed, along with GeoPandas and other required libraries:

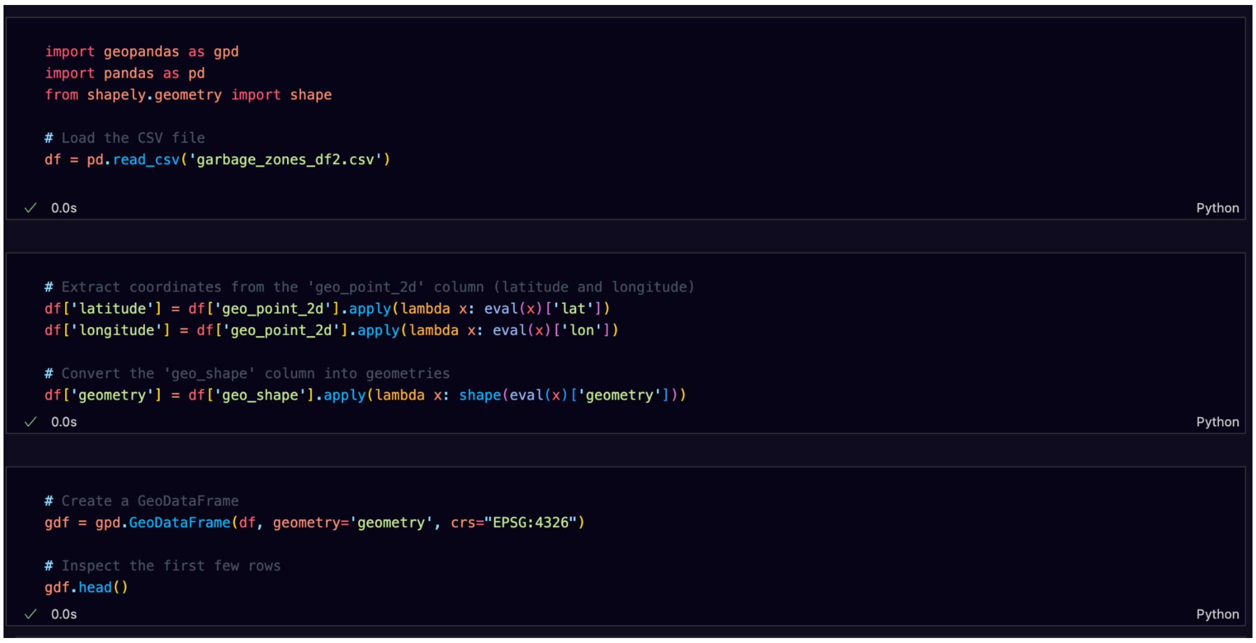
```
pip install geopandas shapely matplotlib
```

---

2. Loading the Dataset

GeoPandas supports reading multiple formats such as shapefiles, GeoJSON, and CSV files containing geographic coordinates.

Here, we load a CSV file that contains information about garbage collection zones, including `latitude`, `longitude`, and `geo_shape` (polygon information):



```python
import geopandas as gpd
import pandas as pd
from shapely.geometry import shape

# Load the CSV file
df = pd.read_csv('garbage_zones_df2.csv')
```

✓ 0.0s Python

```python
# Extract coordinates from the 'geo_point_2d' column (latitude and longitude)
df['latitude'] = df['geo_point_2d'].apply(lambda x: eval(x)['lat'])
df['longitude'] = df['geo_point_2d'].apply(lambda x: eval(x)['lon'])

# Convert the 'geo_shape' column into geometries
df['geometry'] = df['geo_shape'].apply(lambda x: shape(eval(x)['geometry']))
```

✓ 0.0s Python

```python
# Create a GeoDataFrame
gdf = gpd.GeoDataFrame(df, geometry='geometry', crs="EPSG:4326")

# Inspect the first few rows
gdf.head()
```

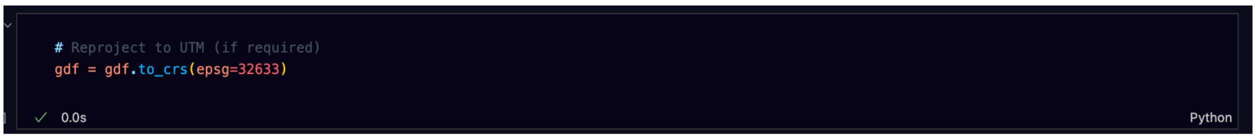
✓ 0.0s Python

---

3. Reprojecting to UTM (if needed)

Reprojecting a GeoDataFrame is essential when working with different datasets that may have different coordinate reference systems (CRS). GeoPandas makes it easy to convert between CRSs.

- `to_crs()`: Reprojects the geometries to a different CRS. For example, EPSG:32633 represents UTM Zone 33N, a projected CRS often used for area-based calculations.
- **Why reproject?**: Reprojecting is necessary for accurate spatial measurements (like area and distance), as geographic CRS like EPSG:4326 distort distances and areas at different latitudes.



```python
# Reproject to UTM (if required)
gdf = gdf.to_crs(epsg=32633)
```

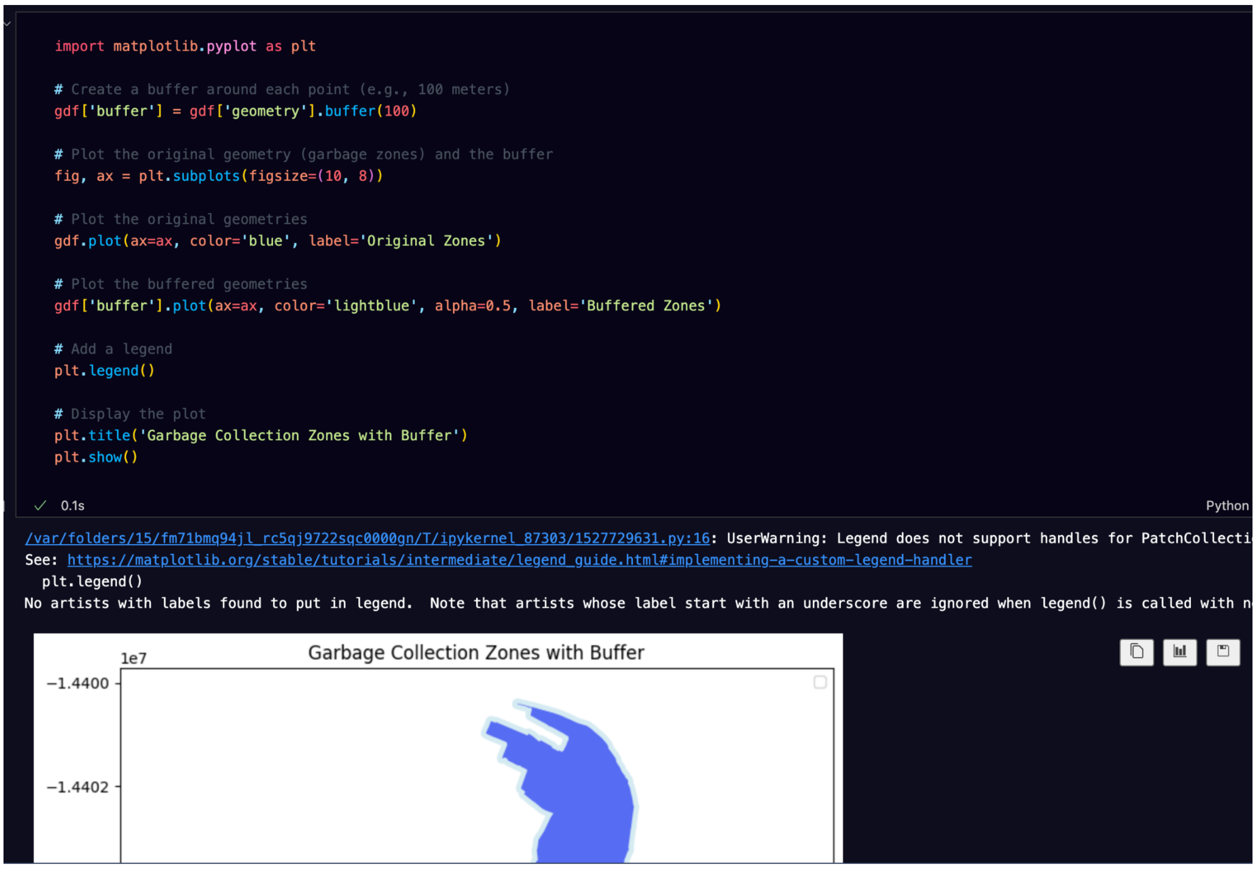
✓ 0.0s Python

---

4. Spatial Operations

a) Buffering

You can create a buffer around each geometry to represent proximity zones, like buffering garbage collection zones by 100 meters:



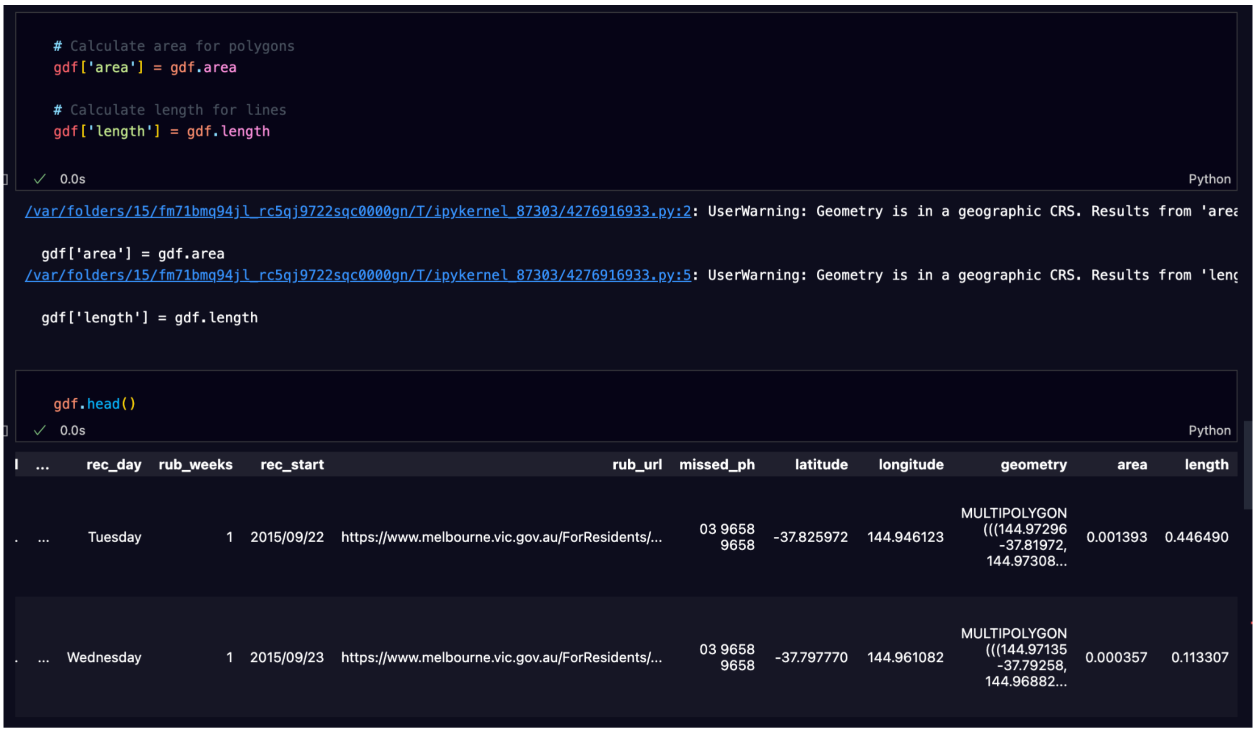
```python
import matplotlib.pyplot as plt

# Create a buffer around each point (e.g., 100 meters)
gdf['buffer'] = gdf['geometry'].buffer(100)

# Plot the original geometry (garbage zones) and the buffer
fig, ax = plt.subplots(figsize=(10, 8))

# Plot the original geometries
gdf.plot(ax=ax, color='blue', label='Original Zones')

# Plot the buffered geometries
gdf['buffer'].plot(ax=ax, color='lightblue', alpha=0.5, label='Buffered Zones')

# Add a legend
plt.legend()

# Display the plot
plt.title('Garbage Collection Zones with Buffer')
plt.show()
```

✓ 0.1s Python

```
/var/folders/15/fm71bmq94jl_rc5qj9722sqc0000gn/T/ipykernel_87303/1527729631.py:16: UserWarning: Legend does not support handles for PatchCollecti
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#implementing-a-custom-legend-handler
  plt.legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with n
```

b) Calculating Area and Length

To calculate the area of polygons or the length of lines in your dataset:



```python
# Calculate area for polygons
gdf['area'] = gdf.area

# Calculate length for lines
gdf['length'] = gdf.length
```

✓ 0.0s Python

```
/var/folders/15/fm71bmq94jl_rc5qj9722sqc0000gn/T/ipykernel_87303/4276916933.py:2: UserWarning: Geometry is in a geographic CRS. Results from 'area

  gdf['area'] = gdf.area
/var/folders/15/fm71bmq94jl_rc5qj9722sqc0000gn/T/ipykernel_87303/4276916933.py:5: UserWarning: Geometry is in a geographic CRS. Results from 'leng

  gdf['length'] = gdf.length
```

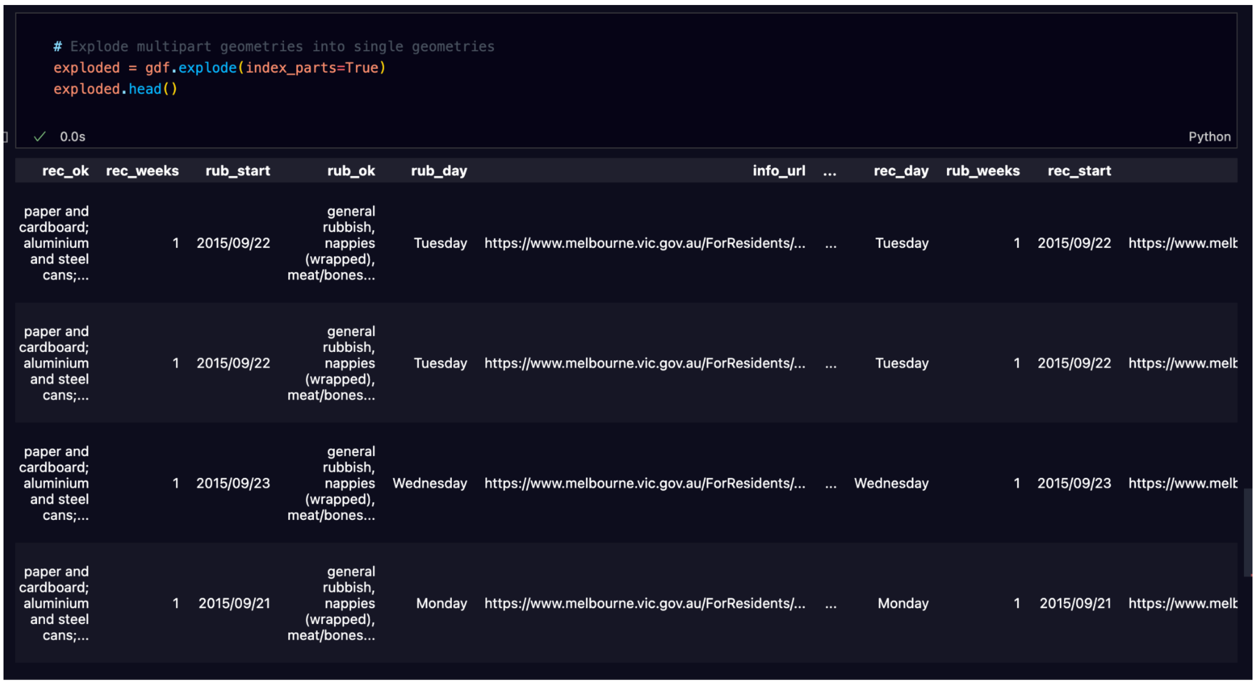
```python
gdf.head()
```

✓ 0.0s Python

l	...	rec_day	rub_weeks	rec_start	rub_url	missed_ph	latitude	longitude	geometry	area	length
.	...	Tuesday	1	2015/09/22	https://www.melbourne.vic.gov.au/ForResidents/...	03 9658 9658	-37.825972	144.946123	MULTIPOLYGON (((144.97296 -37.81972, 144.97308...	0.001393	0.446490
.	...	Wednesday	1	2015/09/23	https://www.melbourne.vic.gov.au/ForResidents/...	03 9658 9658	-37.797770	144.961082	MULTIPOLYGON (((144.97135 -37.79258, 144.96882...	0.000357	0.113307

c) Exploding Multi-Part Geometries

GeoPandas can handle geometries composed of multiple parts (e.g., a multi-polygon). If you need each part as a separate geometry, you can use the `explode()` function. **`explode()`**: Splits multi-part geometries into individual geometries. Each part becomes a separate row in the GeoDataFrame. This is useful when dealing with complex geometries like administrative boundaries or multi-part features.



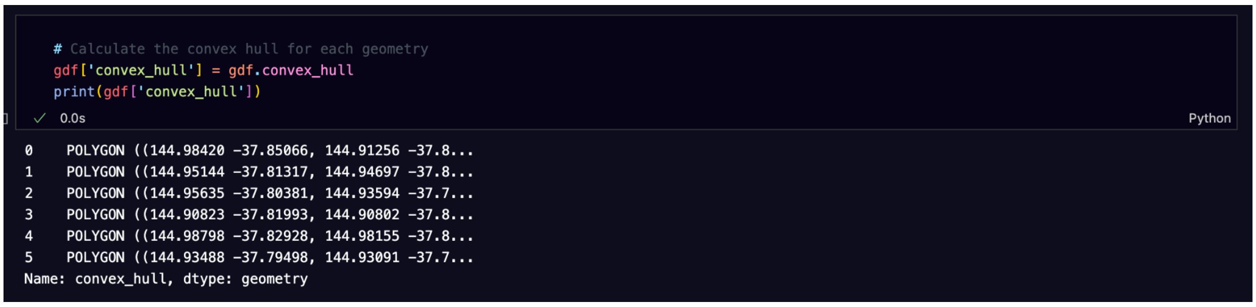
```python
# Explode multipart geometries into single geometries
exploded = gdf.explode(index_parts=True)
exploded.head()
```

✓ 0.0s Python

rec_ok	rec_weeks	rub_start	rub_ok	rub_day	info_url	...	rec_day	rub_weeks	rec_start	
paper and cardboard; aluminium and steel cans;...	1	2015/09/22	general rubbish, nappies (wrapped), meat/bones...	Tuesday	https://www.melbourne.vic.gov.au/ForResidents/...	...	Tuesday	1	2015/09/22	https://www.melt
paper and cardboard; aluminium and steel cans;...	1	2015/09/22	general rubbish, nappies (wrapped), meat/bones...	Tuesday	https://www.melbourne.vic.gov.au/ForResidents/...	...	Tuesday	1	2015/09/22	https://www.melt
paper and cardboard; aluminium and steel cans;...	1	2015/09/23	general rubbish, nappies (wrapped), meat/bones...	Wednesday	https://www.melbourne.vic.gov.au/ForResidents/...	...	Wednesday	1	2015/09/23	https://www.melt
paper and cardboard; aluminium and steel cans;...	1	2015/09/21	general rubbish, nappies (wrapped), meat/bones...	Monday	https://www.melbourne.vic.gov.au/ForResidents/...	...	Monday	1	2015/09/21	https://www.melt

d) Convex Hull

The convex hull is the smallest convex polygon that can enclose a geometry. It is useful for summarizing the extent of points or polygons. **`convex_hull`**: Calculates the convex hull for each geometry in the GeoDataFrame. This operation is useful for understanding the overall spatial extent of a set of geometries.



```python
# Calculate the convex hull for each geometry
gdf['convex_hull'] = gdf.convex_hull
print(gdf['convex_hull'])
```

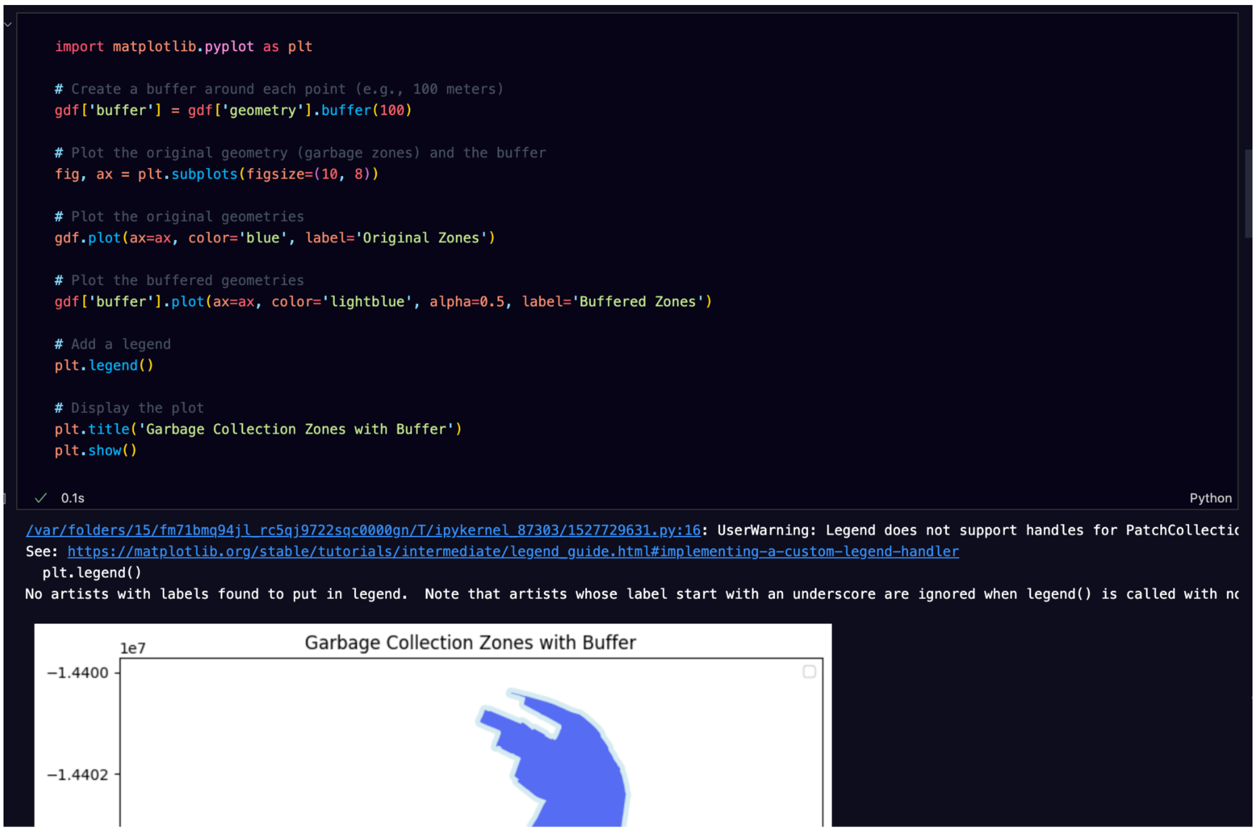
✓ 0.0s Python

```
0    POLYGON ((144.98420 -37.85066, 144.91256 -37.8...
1    POLYGON ((144.95144 -37.81317, 144.94697 -37.8...
2    POLYGON ((144.95635 -37.80381, 144.93594 -37.7...
3    POLYGON ((144.90823 -37.81993, 144.90802 -37.8...
4    POLYGON ((144.98798 -37.82928, 144.98155 -37.8...
5    POLYGON ((144.93488 -37.79498, 144.93091 -37.7...
Name: convex_hull, dtype: geometry
```

5. Visualizing Geospatial Data

a) Basic Plot

You can easily plot geographic data using the built-in plotting functionality of GeoPandas:



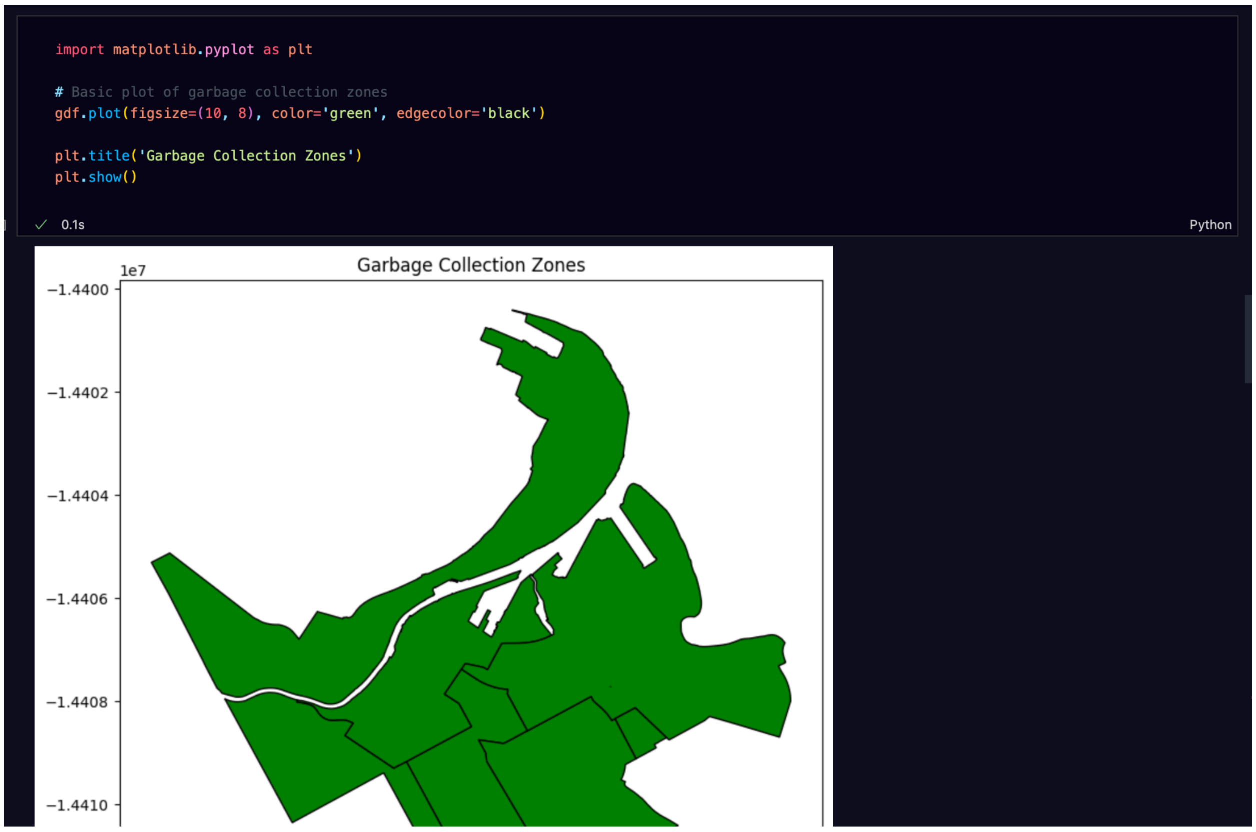
```python
import matplotlib.pyplot as plt

# Create a buffer around each point (e.g., 100 meters)
gdf['buffer'] = gdf['geometry'].buffer(100)

# Plot the original geometry (garbage zones) and the buffer
fig, ax = plt.subplots(figsize=(10, 8))

# Plot the original geometries
gdf.plot(ax=ax, color='blue', label='Original Zones')

# Plot the buffered geometries
gdf['buffer'].plot(ax=ax, color='lightblue', alpha=0.5, label='Buffered Zones')

# Add a legend
plt.legend()

# Display the plot
plt.title('Garbage Collection Zones with Buffer')
plt.show()
```

```
✓ 0.1s                                                                    Python
```

```
/var/folders/15/fm71bmq94jl_rc5qj9722sqc0000gn/T/ipykernel_87303/1527729631.py:16: UserWarning: Legend does not support handles for PatchCollecti
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#implementing-a-custom-legend-handler
  plt.legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with n
```

b) Choropleth Map

To create a choropleth map based on a specific column (e.g., day of waste collection):



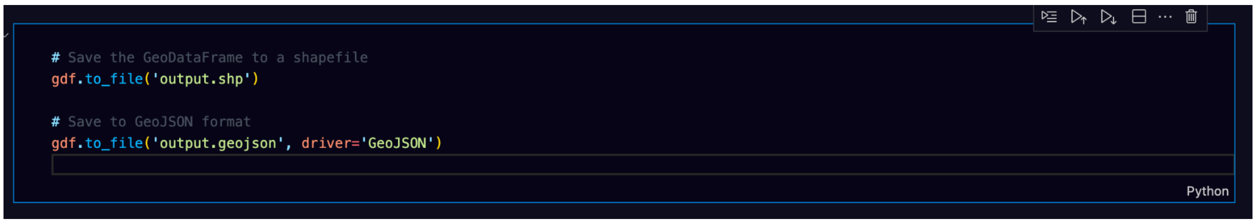
```
import matplotlib.pyplot as plt

# Basic plot of garbage collection zones
gdf.plot(figsize=(10, 8), color='green', edgecolor='black')

plt.title('Garbage Collection Zones')
plt.show()
```

```
✓ 0.1s                                                    Python
```

6. Saving Processed Data

You can export the results of your geospatial analysis back into standard geographic formats like shapefiles or GeoJSON.

- **`to_file()`**: Saves the `GeoDataFrame` into a file format. You can choose to save it as a shapefile (`.shp`) or GeoJSON (`.geojson`) depending on your needs.
- **`driver='GeoJSON'`**: Specifies the output format. GeoPandas uses `shapefiles` by efault unless the driver is specified.



```
# Save the GeoDataFrame to a shapefile
gdf.to_file('output.shp')

# Save to GeoJSON format
gdf.to_file('output.geojson', driver='GeoJSON')
```

```
Python
```

Conclusion

This guide has covered essential operations using GeoPandas, including loading geographic data, performing spatial operations such as buffering, calculating areas and lengths, and visualizing geospatial data. These tools are invaluable for handling geographic datasets and performing spatial analyses such as optimizing waste management routes or understanding spatial distribution patterns.