



VISUALIZING GEOSPATIAL DATA IN PYTHON

# Plotting with GeoJSON

Mary van Valkenburg

Data Science Program Manager, Nashville Software School

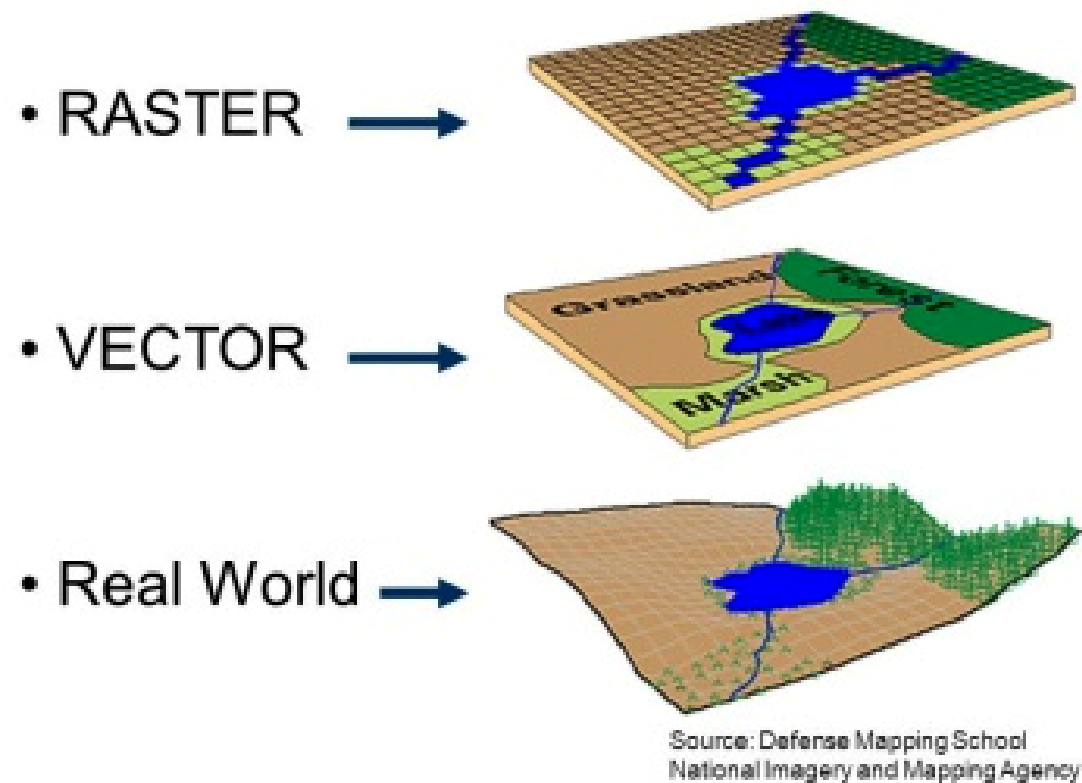
# Neighborhoods GeoJSON

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "name": "Historic Buena Vista"
      }, "geometry": {
        "type": "MultiPolygon", "coordinates": [[ [
          [-86.79511056795417, 36.17575964963348],
          [-86.79403325521203, 36.176723819622765],
          [-86.79395847673587, 36.176734201205555],
          [-86.79373059621346, 36.17641850227536],
```

```
neighborhoods = gpd.read_file('./data/neighborhood_boundaries.geojson')
neighborhoods.head(1)
```

name	geometry
Historic Buena Vista	(POLYGON ((-86.79511056795417 36.17575964963348)))

# Geopandas dependencies



- Fiona
  - provides a python API for OGR
- GDAL/OGR
  - GDAL for translating raster data
  - OGR for translating vector data

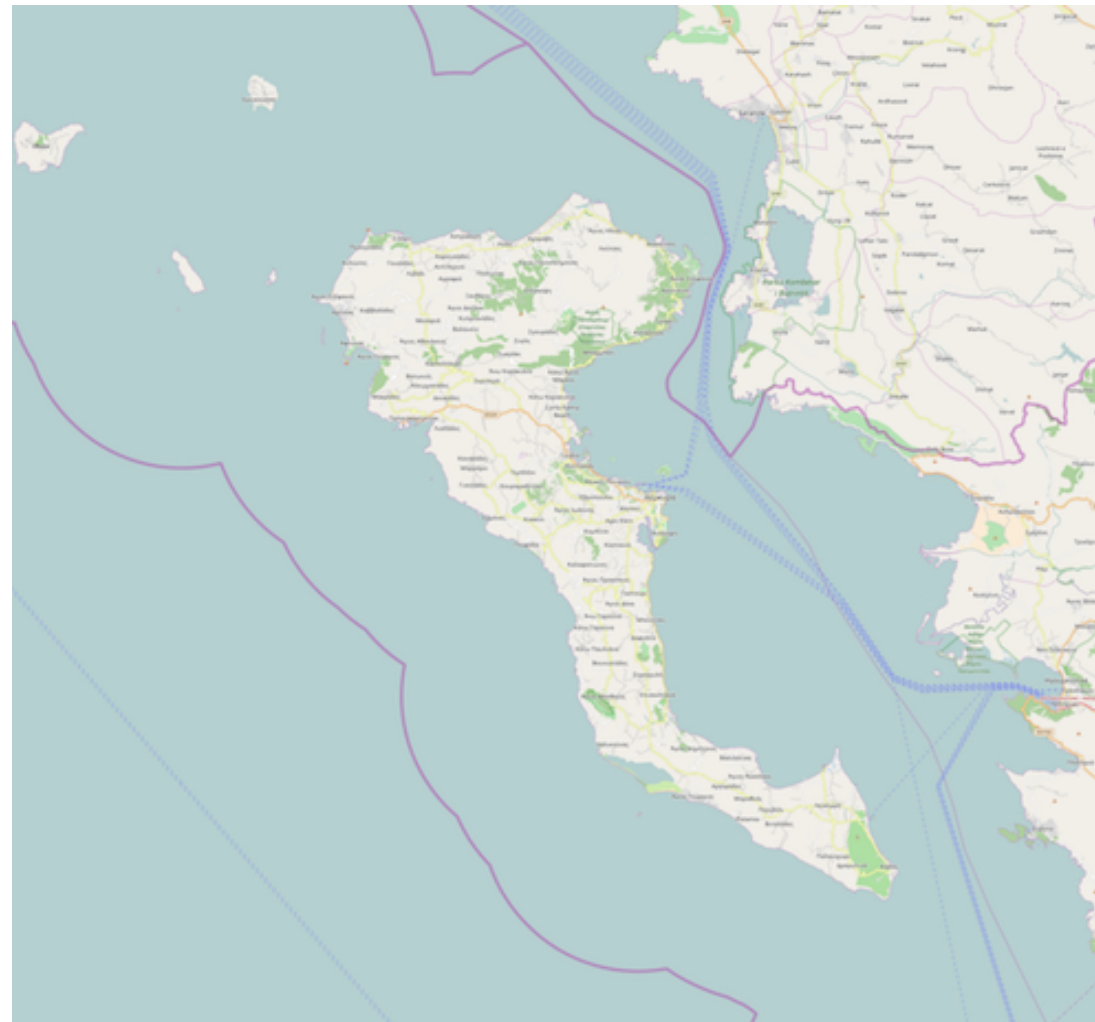


# Comparing raster and vector graphics

raster image of Corfu

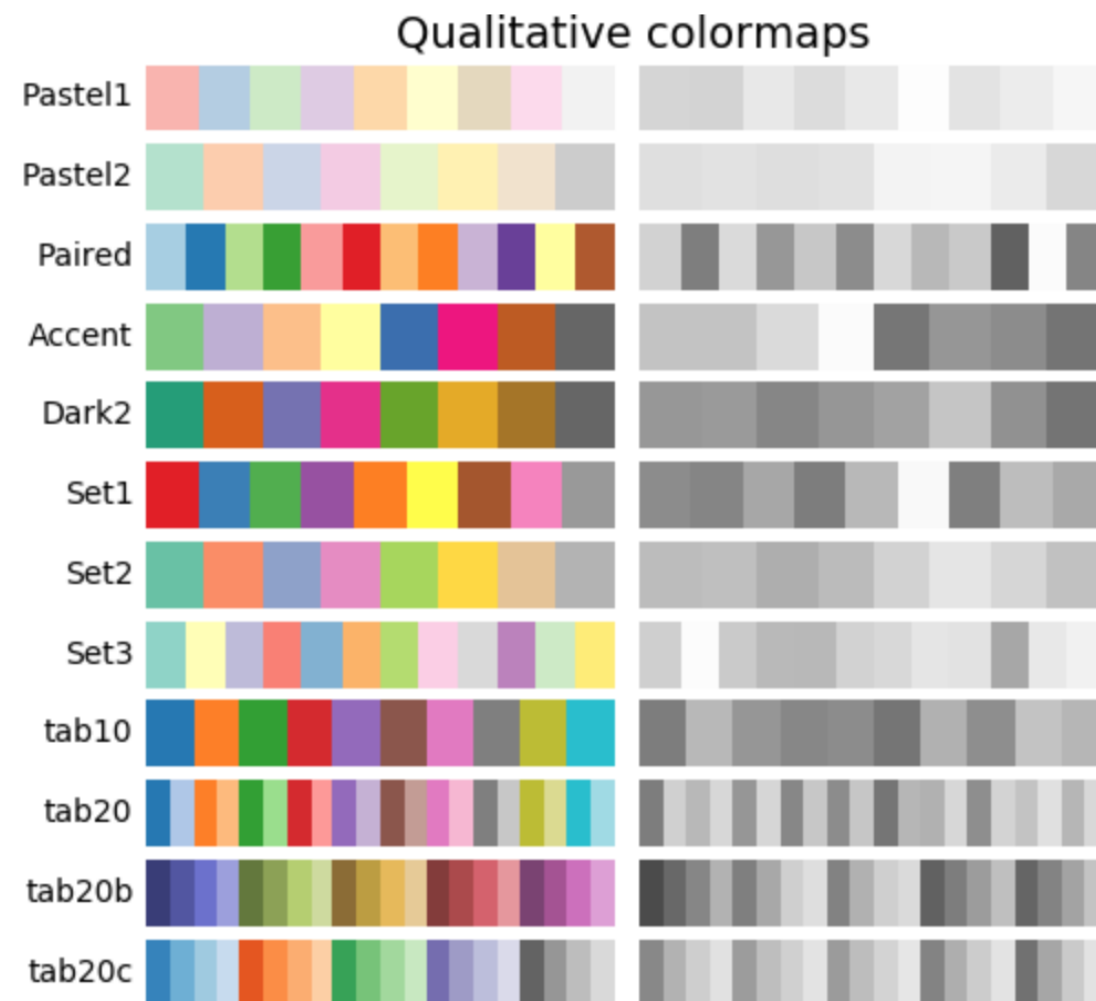


vector image of Corfu





# Colormaps

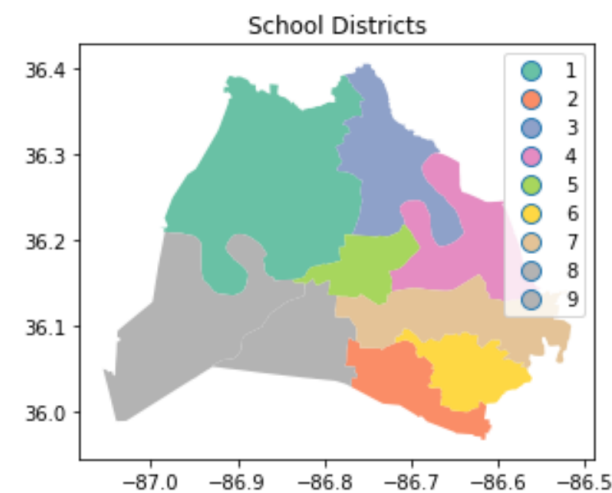
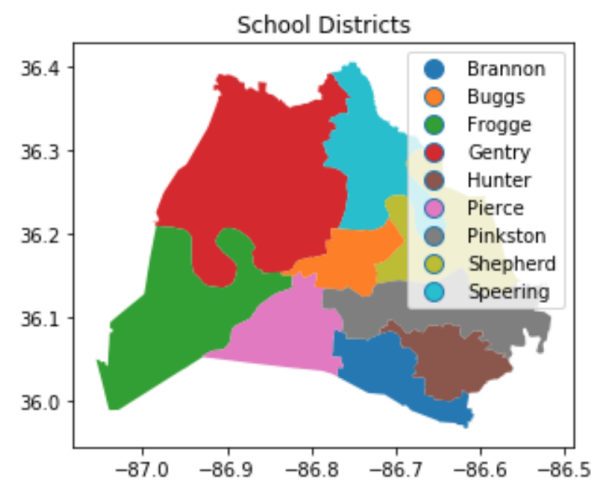


<https://matplotlib.org/users/colormaps.html>

# Plotting with color

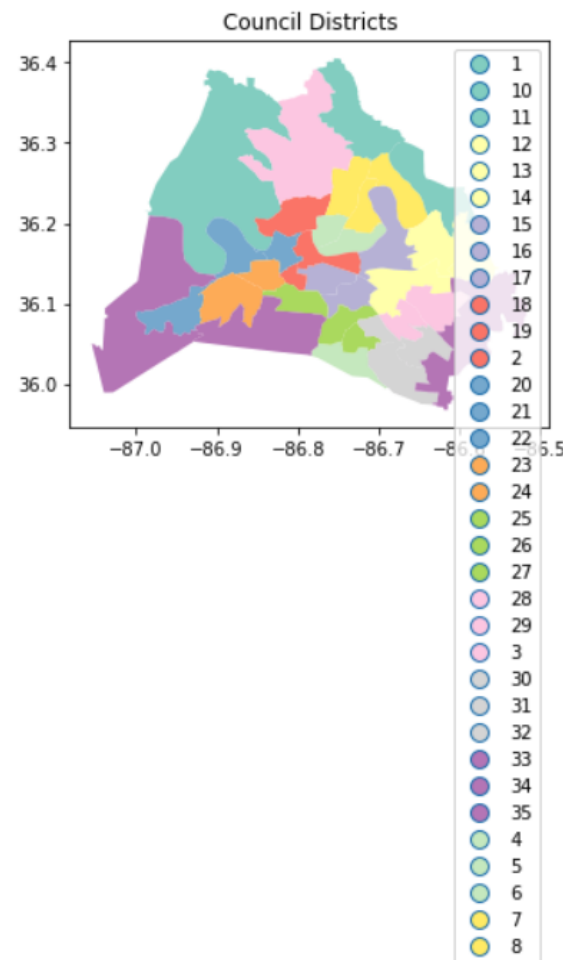
```
school_districts.head(3)
```

first_name	last_name	position	district	geometry
Sharon	Gentry	Member	1	(POLYGON ((-86.771 36.383)))
Jill	Speering	Vice-Chair	3	(POLYGON ((-86.753 36.404)))
Jo Ann	Brannon	Member	2	(POLYGON ((-86.766 36.083)))

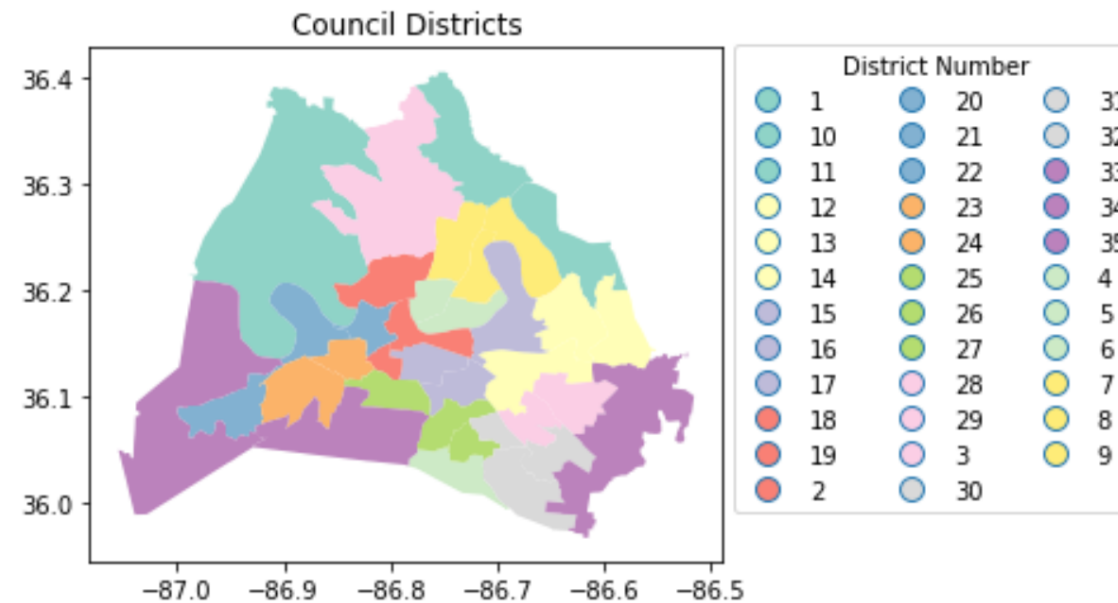


# The legend\_kwds argument to .plot()

```
council_dists.plot(
    column='district',
    cmap='Set3',
    legend=True)
plt.title('Council Districts')
plt.show();
```



```
leg_kwds={ 'title': 'District Number',
            'loc': 'upper left',
            'bbox_to_anchor': (1, 1.03),
            'ncol': 3}
council_dists.plot(column='district',
    cmap='Set3',
    legend=True,
    legend_kwds=leg_kwds)
plt.title('Council Districts')
plt.show();
```





## VISUALIZING GEOSPATIAL DATA IN PYTHON

**Let's practice!**



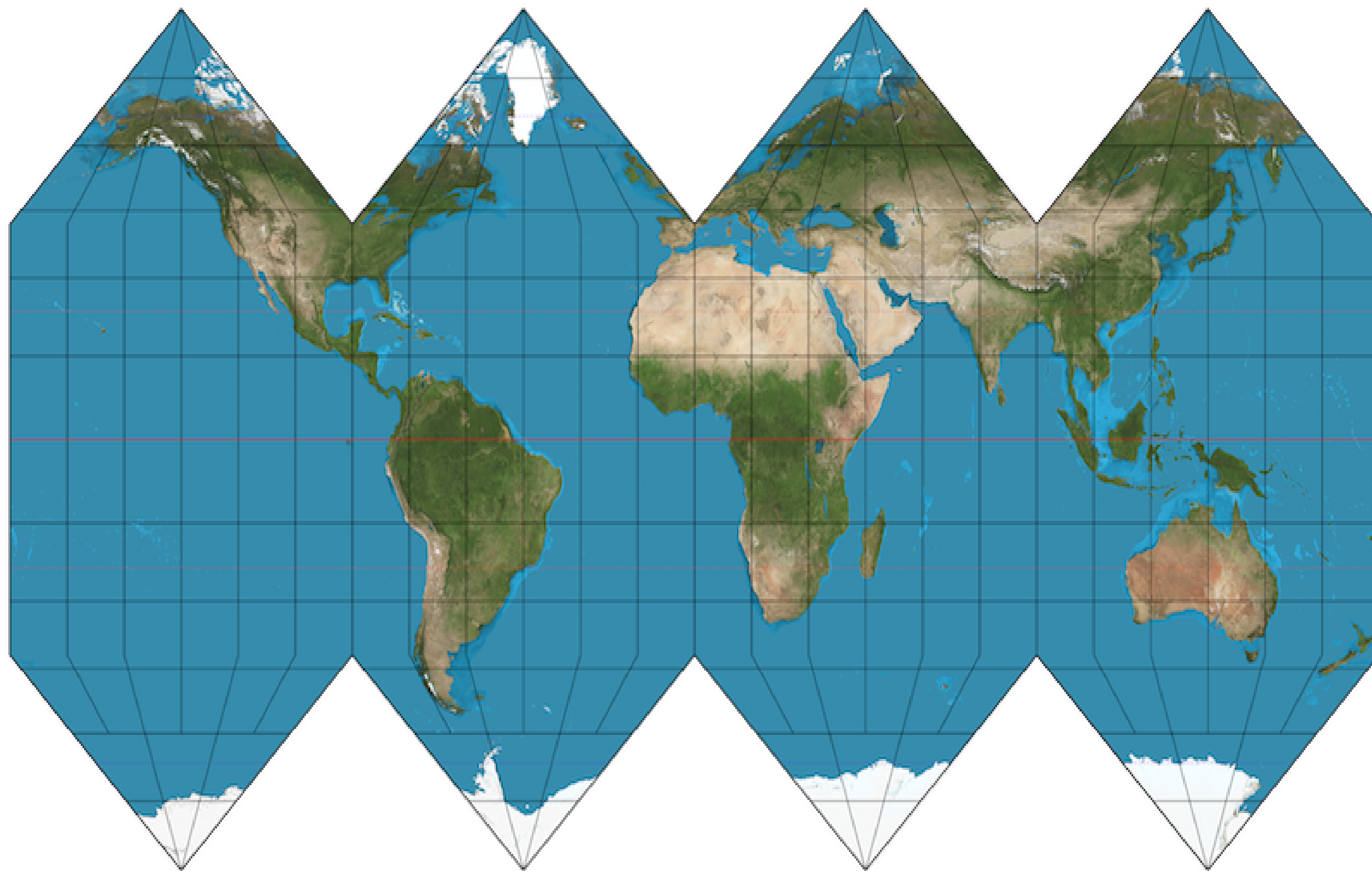


VISUALIZING GEOSPATIAL DATA IN PYTHON

# Projections and Coordinate Reference Systems

Mary van Valkenburg

Data Science Program Manager, Nashville Software School





# WHAT YOUR FAVORITE MAP PROJECTION SAYS ABOUT YOU

MERCATOR



YOU'RE NOT REALLY INTO MAPS.

ROBINSON



YOU HAVE A COMFORTABLE PAIR OF RUNNING SHOES THAT YOU WEAR EVERYWHERE. YOU LIKE COFFEE AND ENJOY THE BEATLES. YOU THINK THE ROBINSON IS THE BEST-LOOKING PROJECTION, HANDS DOWN.

WINKEL-TRIPLE



NATIONAL GEOGRAPHIC ADOPTED THE WINKEL-TRIPLEL IN 1998, BUT YOU'VE BEEN A WIT FAN SINCE LONG BEFORE "NAT GEO" SHOWED UP. YOU'RE WORRIED IT'S GETTING PLAYED OUT, AND ARE THINKING OF SWITCHING TO THE KAVRANSKY. YOU ONCE LEFT A PARTY IN DISGUST WHEN A GUEST SHOWED UP WEARING SHOES WITH TOES. YOUR FAVORITE MUSICAL GENRE IS "POST-".

VAN DER GRINTEN



YOU'RE NOT A COMPLICATED PERSON. YOU LOVE THE MERCATOR PROJECTION; YOU JUST WISH IT WEREN'T SQUARE. THE EARTH'S NOT A SQUARE, IT'S A CIRCLE. YOU LIKE CIRCLES. TODAY IS GONNA BE A GOOD DAY!

DYMATION



YOU LIKE ISAAC ASIMOV, XML, AND SHOES WITH TOES. YOU THINK THE SEGWAY GOT A BAD RAP. YOU OWN 3D GOGGLES, WHICH YOU USE TO VIEW ROTATING MODELS OF BETTER 3D GOGGLES. YOU TYPE IN DVORAK.

GOODE HOMOLoSINE



THEY SAY MAPPING THE EARTH ON A 2D SURFACE IS LIKE FLATTENING AN ORANGE PEEL, WHICH SEEMS EASY ENOUGH TO YOU. YOU LIKE EASY SOLUTIONS. YOU THINK WE WOULDN'T HAVE SO MANY PROBLEMS IF WE'D JUST ELECT *MORPHIC* PEOPLE TO CONGRESS INSTEAD OF POLITICIANS. YOU THINK AIRLINES SHOULD JUST BUY FOOD FROM THE RESTAURANTS NEAR THE GATES AND SERVE *THAT* ON BOARD. YOU CHANGE YOUR CAR'S OIL, BUT SECRETLY WONDER IF YOU REALLY *NEED* TO.

HOB0-DYER



YOU WANT TO AVOID CULTURAL IMPERIALISM, BUT YOU'VE HEARD BAD THINGS ABOUT GALL-PETERS. YOU'RE CONFLICT-AVERSE AND BUY ORGANIC. YOU USE A RECENTLY-INVENTED SET OF GENDER-NEUTRAL PRONOUNS AND THINK THAT WHAT THE WORLD NEEDS IS A REVOLUTION IN CONSCIOUSNESS.

A GLOBE!



YES, YOU'RE VERY CLEVER.

PEIRCE QUINCUNCIAL



YOU THINK THAT WHEN WE LOOK AT A MAP, WHAT WE REALLY SEE IS OURSELVES. AFTER YOU FIRST SAW *INCEPTION*, YOU SAT SILENT IN THE THEATER FOR SIX HOURS. IT BREAKS YOU OUT TO REALIZE THAT EVERYONE AROUND YOU HAS A SKELETON INSIDE THEM. YOU *HAVE* REALLY LOOKED AT YOUR HANDS.

PLATE CARRÉE  
(EQUIRECTANGULAR)

YOU THINK THIS ONE IS FINE. YOU LIKE HOW X AND Y MAP TO LATITUDE AND LONGITUDE. THE OTHER PROJECTIONS OVERCOMPLICATE THINGS. YOU WANT ME TO STOP ASKING ABOUT MAPS SO YOU CAN ENJOY DINNER.

WATERMAN BUTTERFLY



REALLY? YOU KNOW THE WATERMAN? HAVE YOU SEEN THE 1909 CAHILL MAP IT'S BASED — ...YOU HAVE A FRAMED REPRODUCTION AT HOME?! WHOA. ...LISTEN, FORGET THESE QUESTIONS. ARE YOU DOING ANYTHING TONIGHT?

GALL-PETERS



I HATE YOU.

What's that? You think I don't like the Peters map because I'm uncomfortable with having my cultural assumptions challenged? Are you sure you're not...

::puts on sunglasses:: ... **projecting?**

<http://xkcd.com/977/> - <http://bit.ly/explainxkcd-977>



# Coordinate Reference Systems

## **EPSG:4326**

- used by Google Earth
- units are decimal degrees

## **EPSG:3857**

- used by Google Maps, Bing Maps, Open Street Maps
- units are meters

# Creating a geometry column

School Name	Latitude	Longitude
A. Z. Kelley Elementary	36.021	-86.658
Alex Green Elementary	36.252	-86.832
Amqui Elementary	36.273	-86.703
Andrew Jackson Elementary	36.231	-86.623

```
# create a point geometry column
from shapely.geometry import Point

schools['geometry'] = schools.apply(
    lambda x: Point((x.Longitude, x.Latitude)),
    axis = 1)

schools.head()
```

School Name	Latitude	Longitude	geometry
A. Z. Kelley Elementary	36.021	-86.658	POINT (-86.658 36.021)
Alex Green Elementary	36.252	-86.832	POINT (-86.832 36.252)
Amqui Elementary	36.273	-86.703	POINT (-86.703 36.273)
Andrew Jackson Elementary	36.231	-86.623	POINT (-86.623 36.231)



# Creating a GeoDataFrame from a DataFrame

```
import geopandas as gpd

schools_crs = {'init': 'epsg:4326'}
schools_geo = gpd.GeoDataFrame(schools,
                               crs = schools_crs,
                               geometry = schools.geometry)
```

```
schools_geo.head(4)
```

School Name	Latitude	Longitude	geometry
A. Z. Kelley Elementary	36.021	-86.658	POINT (-86.658 36.021)
Alex Green Elementary	36.252	-86.832	POINT (-86.832 36.252)
Amqui Elementary	36.273	-86.703	POINT (-86.703 36.273)
Andrew Jackson Elementary	36.231	-86.623	POINT (-86.623 36.231)

# Changing from one CRS to another

```
schools_geo.head(2)
```

School Name	Latitude	Longitude	geometry
A. Z. Kelley Elementary	36.021	-86.658	POINT (-86.658 36.021)
Alex Green Elementary	36.252	-86.832	POINT (-86.832 36.252)

```
# convert geometry from decimal degrees to meters  
schools_geo.geometry = schools_geo.geometry.to_crs(epsg = 3857)  
schools_geo.head(2)
```

School Name	Latitude	Longitude	geometry
A. Z. Kelley Elementary	36.021	-86.658	POINT (-9646818.8 4303623.8)
Alex Green Elementary	36.252	-86.832	POINT (-9666119.5 4335484.4)



## VISUALIZING GEOSPATIAL DATA IN PYTHON

**Let's practice!**





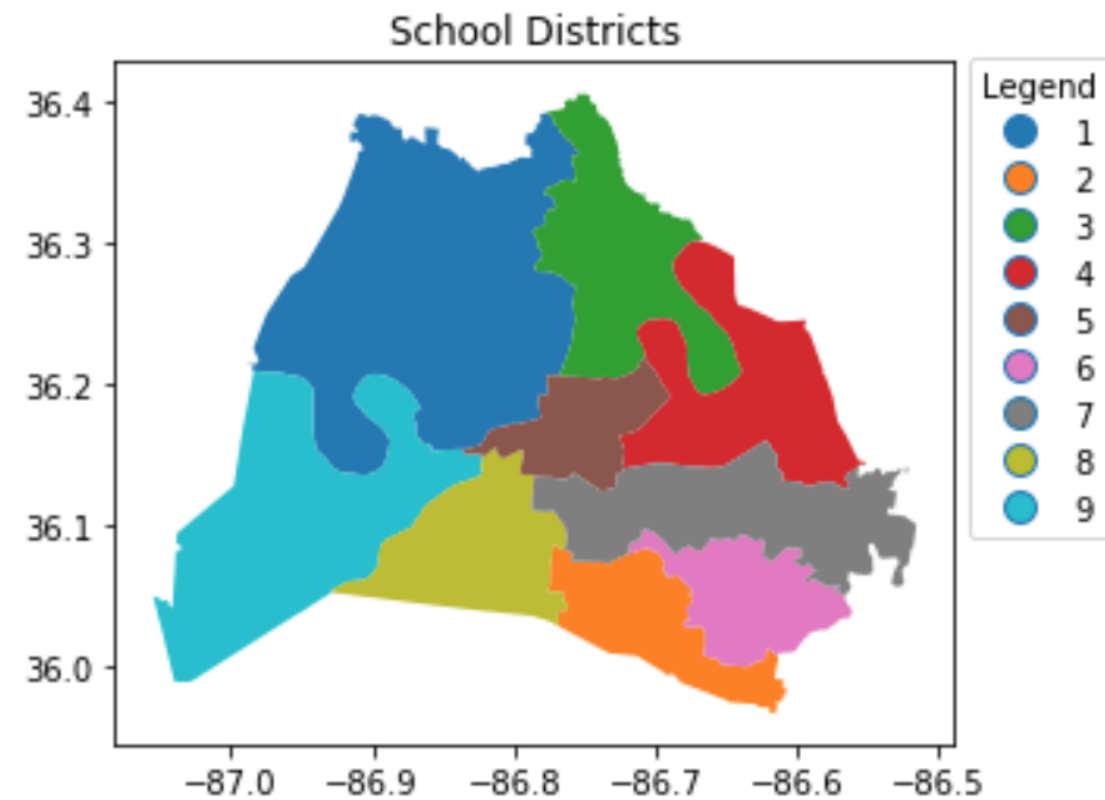
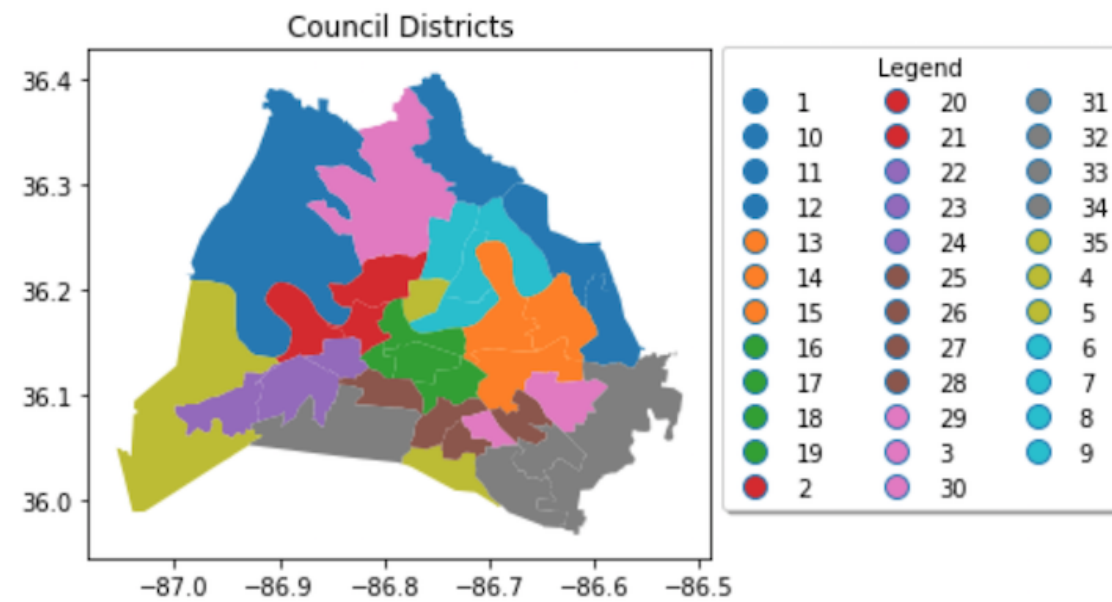
## VISUALIZING GEOSPATIAL DATA IN PYTHON

# Spatial joins

Mary van Valkenburg

Data Science Program Manager, Nashville Software School

# Council districts and school districts





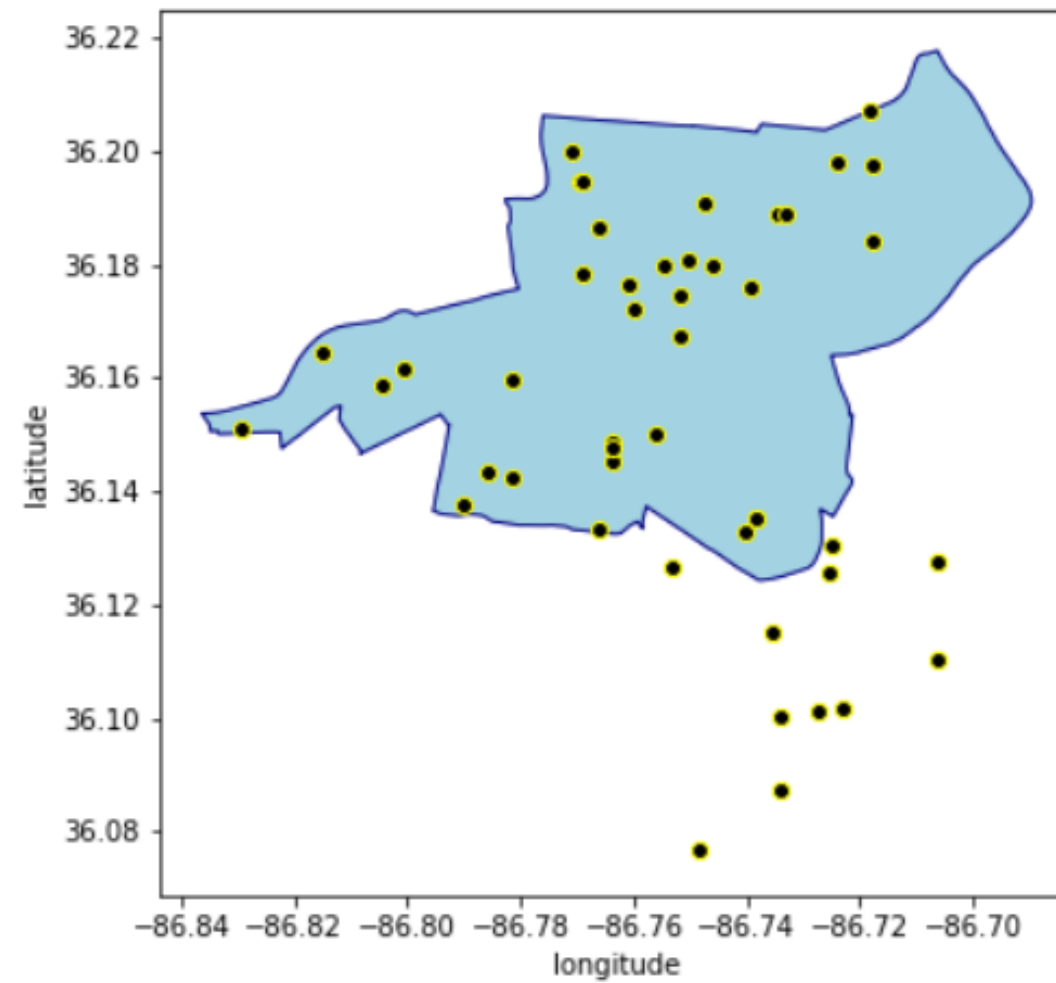
# The .sjoin() op argument

```
import geopandas as gpd  
  
gpd.sjoin(blue_region_gdf, black_point_gdf, op = <operation>)
```

operation can be ***intersects***, ***contains***, or ***within***

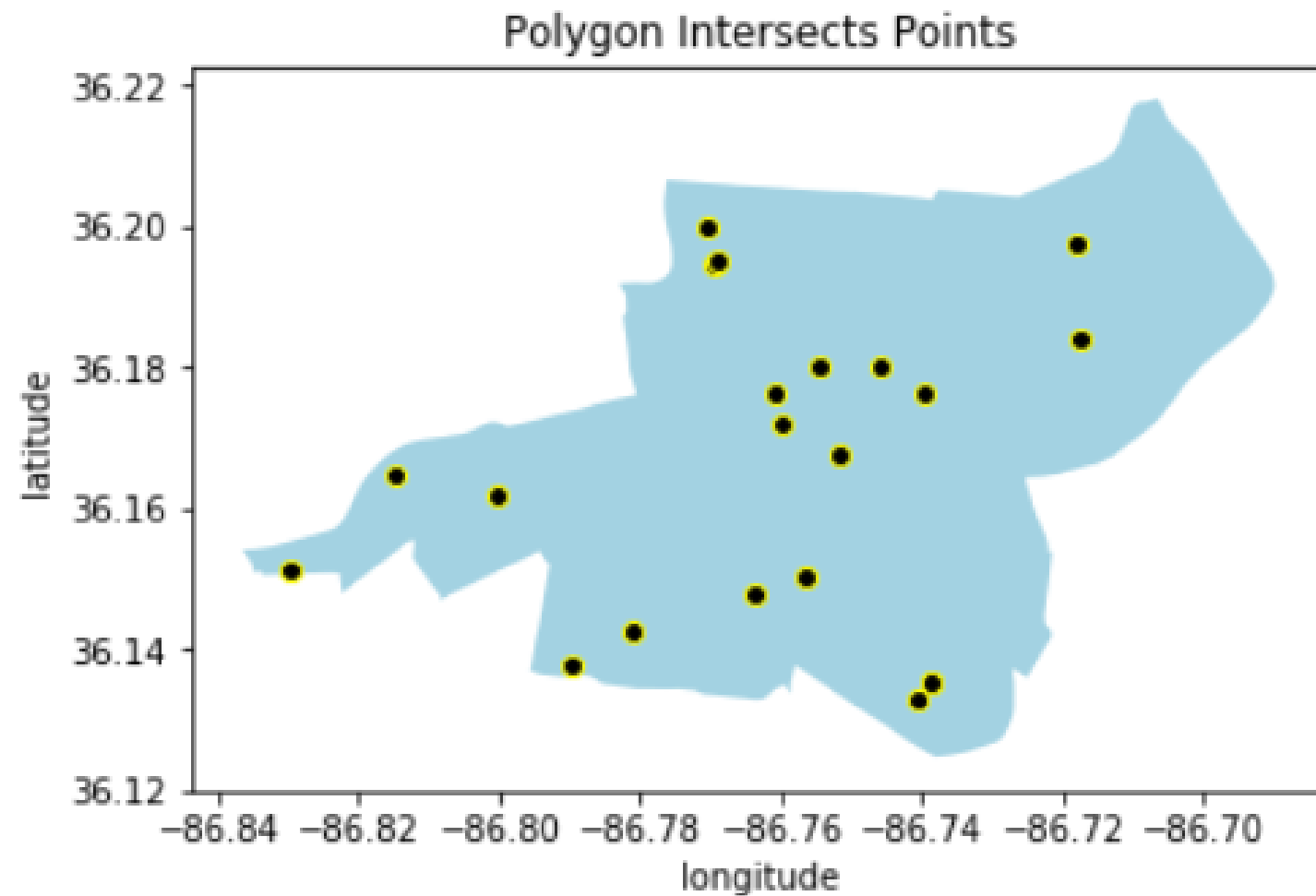


# Using `.sjoin()`



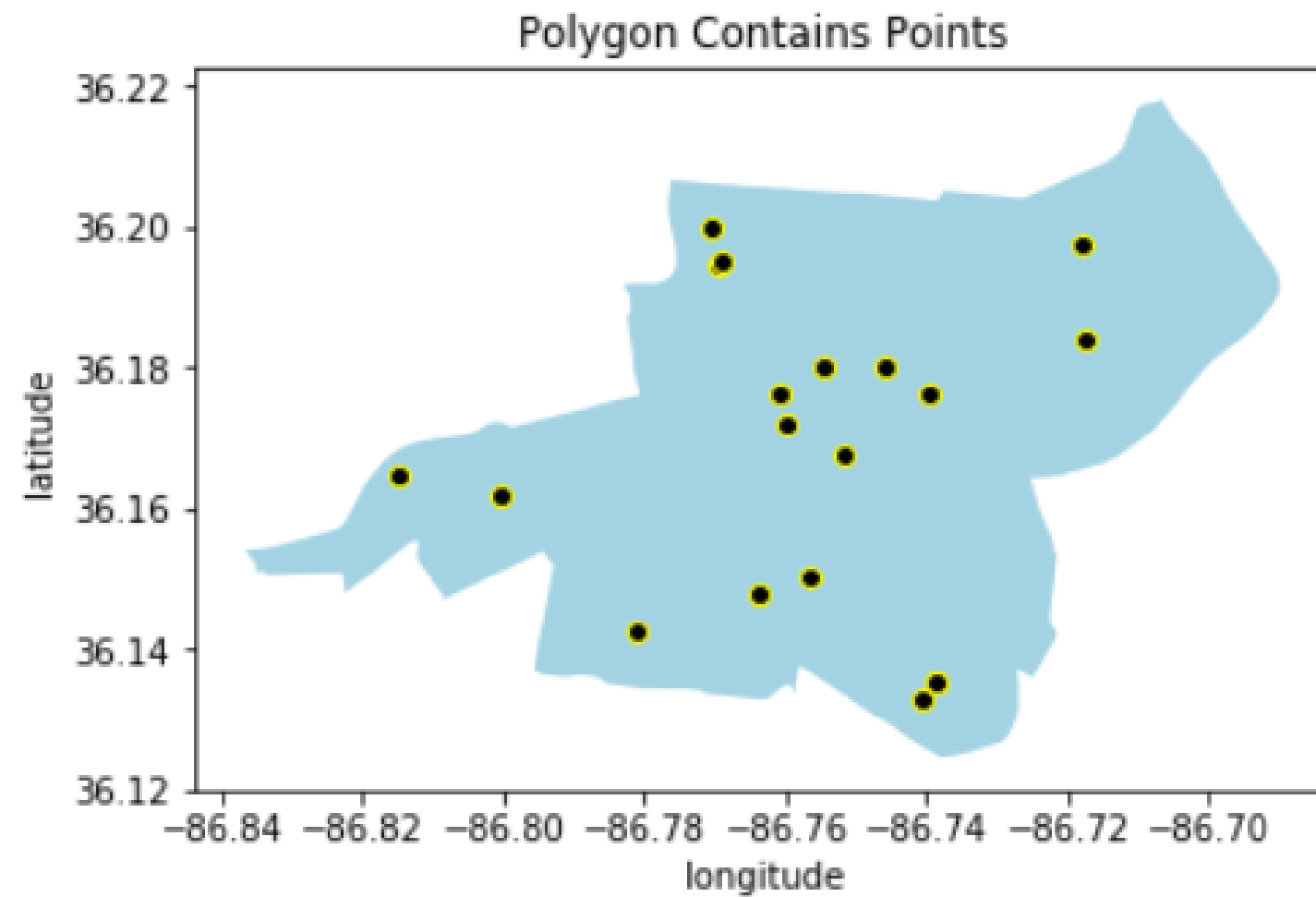
# op = 'intersects'

```
gpd.sjoin(blue_region_gdf, black_point_gdf, op = 'intersects')
```



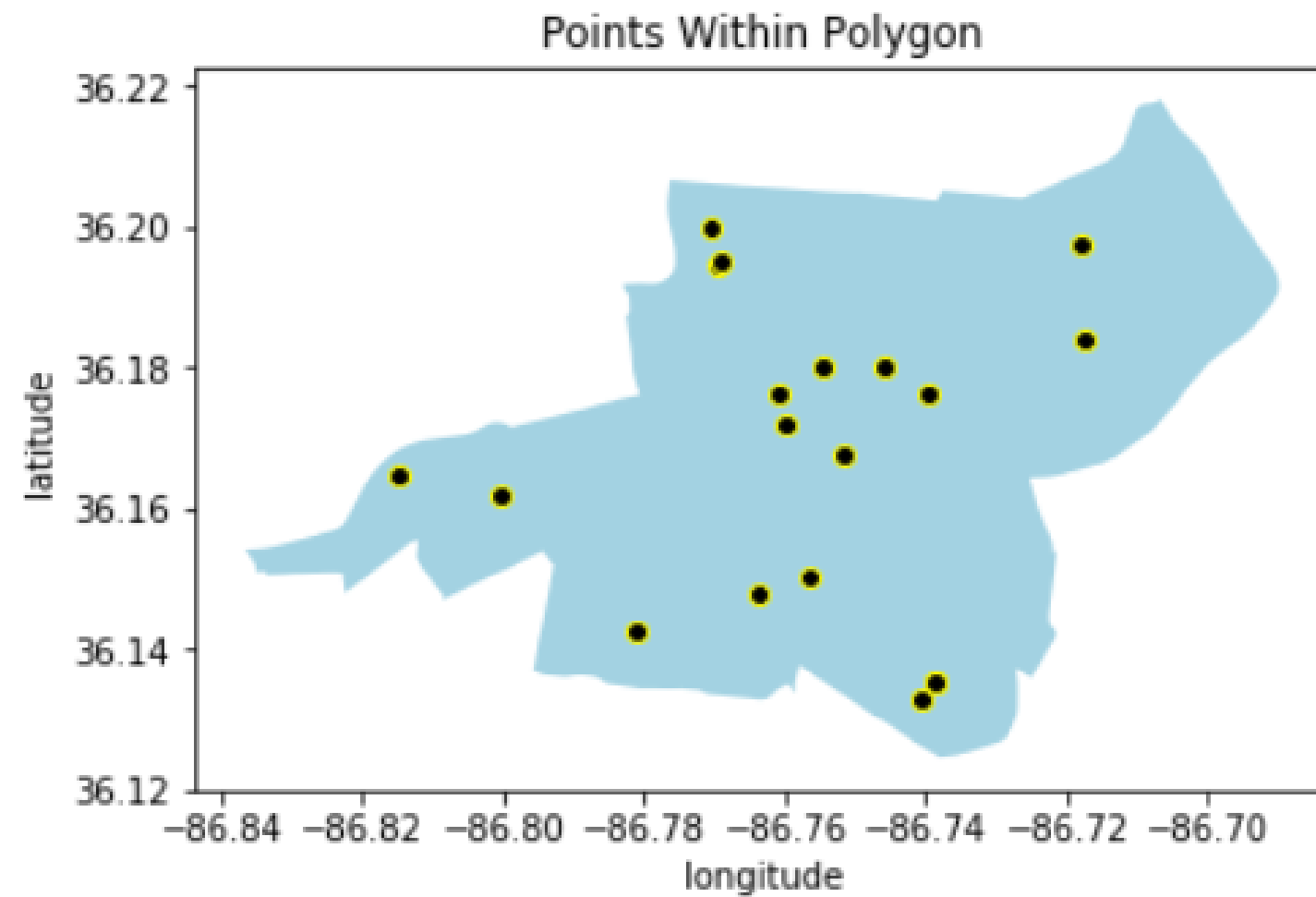
# op = 'contains'

```
gpd.sjoin(blue_region_gdf, black_point_gdf, op = 'contains')
```



# op = 'within'

```
gpd.sjoin(black_point_gdf, blue_region_gdf, op = 'within')
```





# The sjoin.() op argument - within

```
# find council districts within school districts

within_gdf = gpd.sjoin(council_districts, school_districts, op='within')
print('council districts within school districts: ', within_gdf.shape[0])
```

```
council districts within school districts: 11
```





# The sjoin.() op argument - contains

```
# find school districts that contain council districts

contains_gdf=pd.sjoin(school_districts, council_districts, op='contains')
print('school districts contain council districts: ', contains_gdf.shape[0])
```

```
school districts contain council districts: 11
```



# The sjoin.() op argument - intersects

```
# find council districts that intersect with school districts

intersect_gdf=gpd.sjoin(council_districts, school_districts, op='intersects')
print('council districts intersect school districts: ', intersect.shape[0])
```

```
council districts intersect school districts: 100
```



# Columns in a spatially joined GeoDataFrame

```
within_gdf=gpd.sjoin(council_districts, school_districts, op = 'within')  
within_gdf.head()
```

	first_name_left	last_name_left	district_left	index_right
0	Nick	Leonardo	1	0
1	DeCosta	Hastings	2	0
2	Nancy	VanReece	8	1
3	Bill	Pridemore	9	1
9	Doug	Pardue	10	1



# Aggregating spatially joined data

```
# Aggregate council districts by school district
# to see how many council districts are within each school district.

# first rename district_left and district_right
within_gdf.district_left = council_district
within_gdf.district_right = school_district

within_gdf[['council_district', 'school_district']]
    .groupby('school_district')
    .agg('count')
    .sort_values('council_district', ascending = False)
```

school_district	council_district
3	3
1	2
9	2
2	1
5	1
6	1
8	1



## VISUALIZING GEOSPATIAL DATA IN PYTHON

**Let's Practice!**