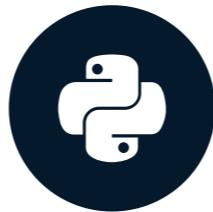


# Introduction

VISUALIZING GEOSPATIAL DATA IN PYTHON

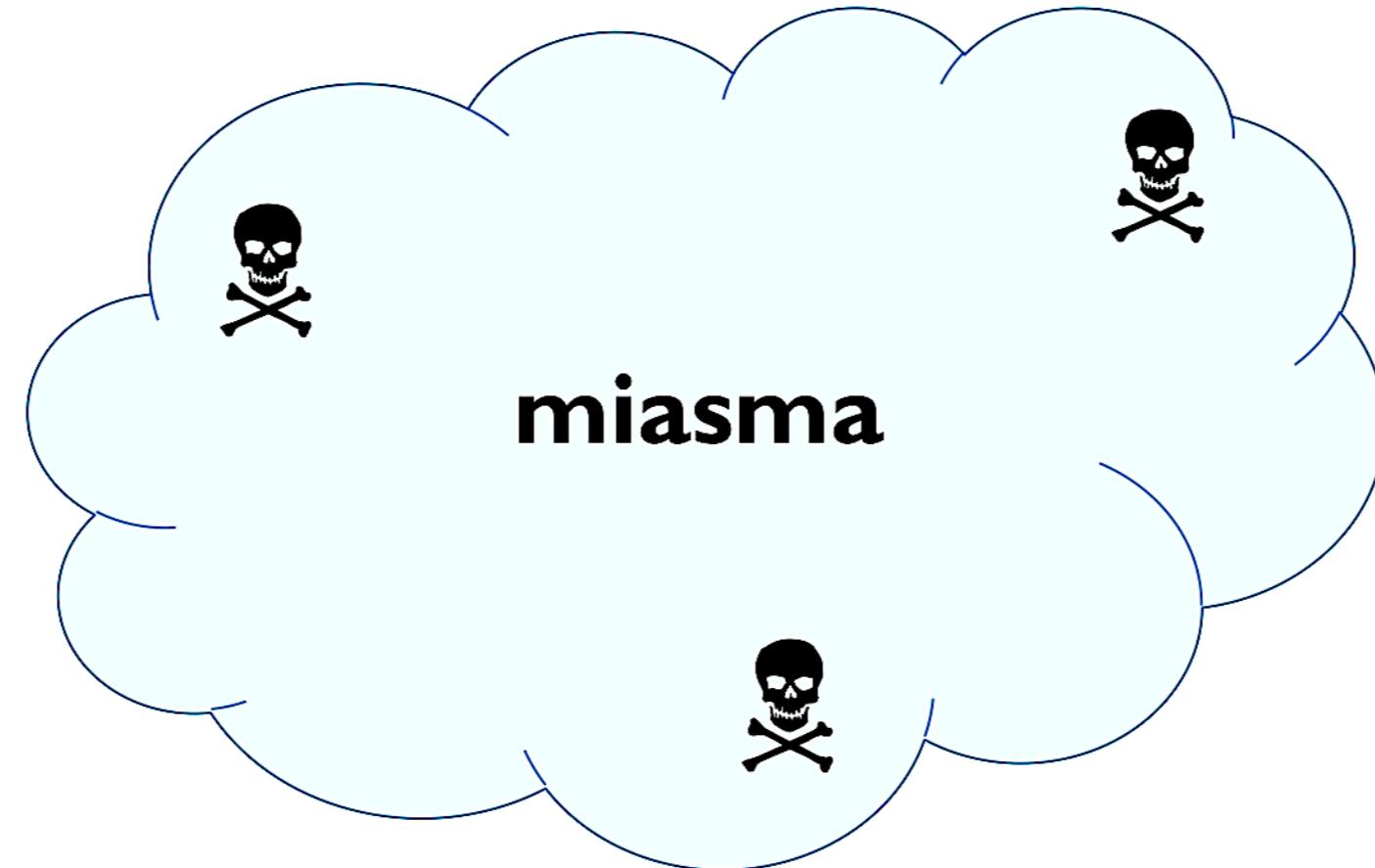


Mary van Valkenburg

Data Science Program Manager,  
Nashville Software School

# Location

- 1854 cholera outbreak in London
- 600+ deaths



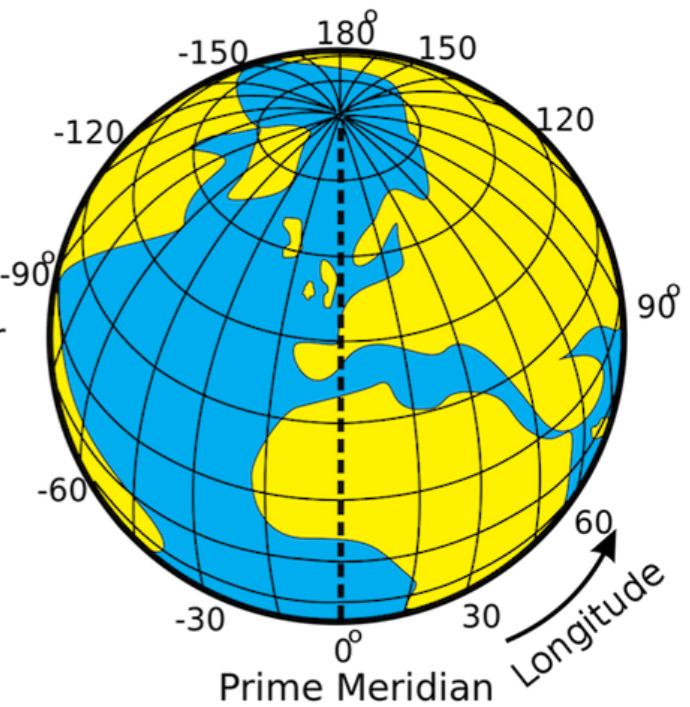
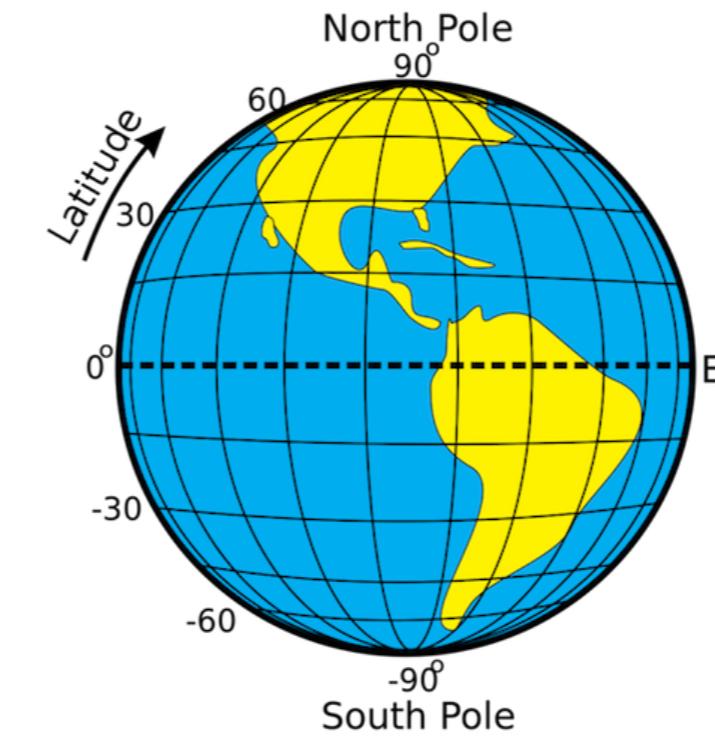
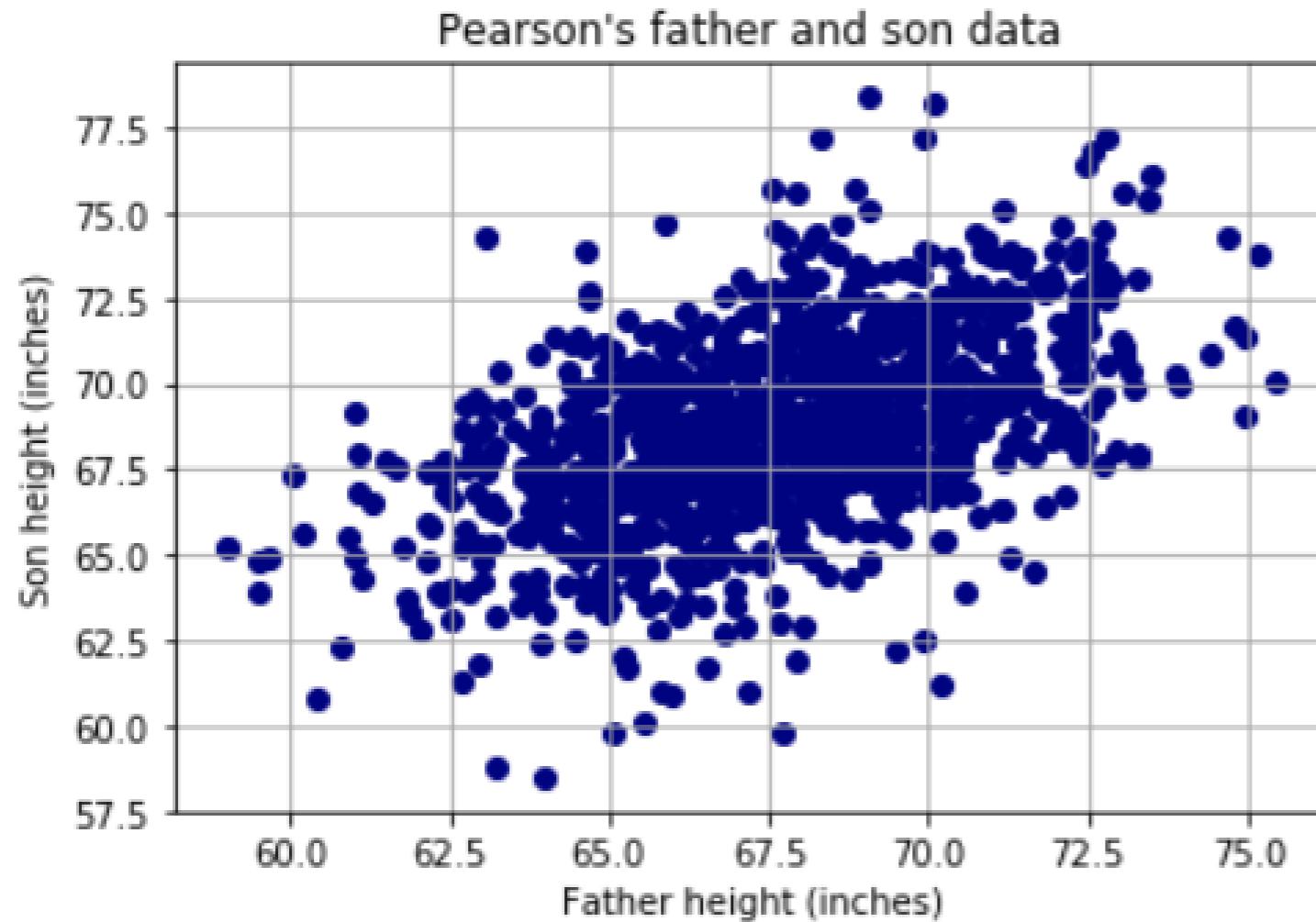
# Snow's dot map



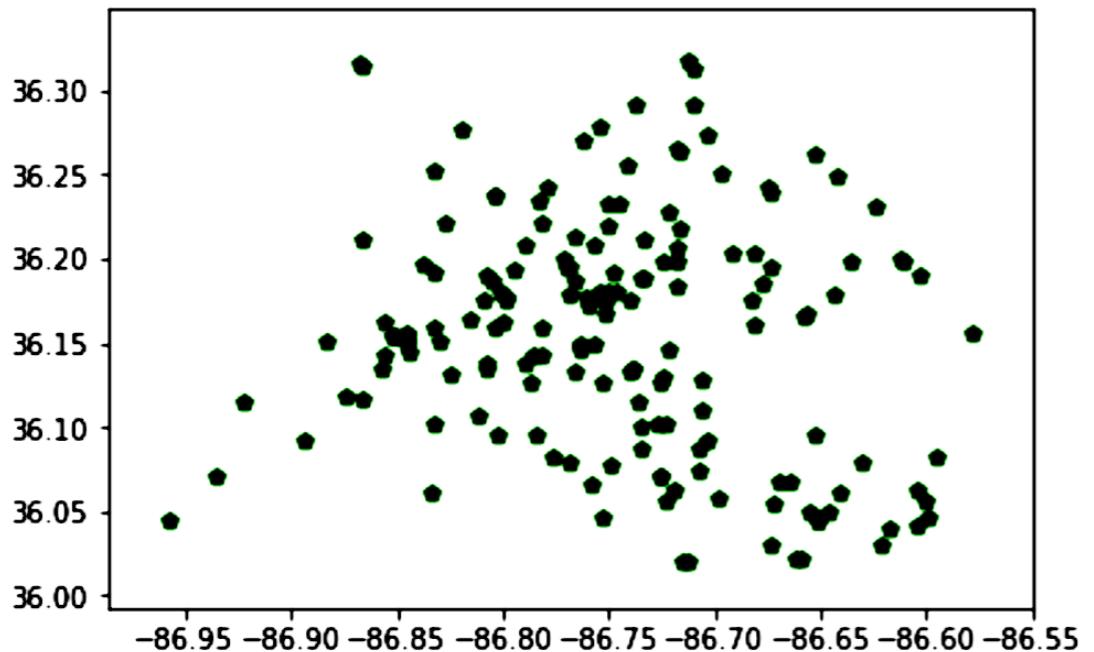
# What you will learn in this course

- How to plot geospatial points as scatterplots
- How to plot geometries using `geopandas`
- How to construct a `GeoDataFrame` from a `DataFrame`
- How to spatially join datasets
- How to add a street map to your plots
- When and how to create a choropleth

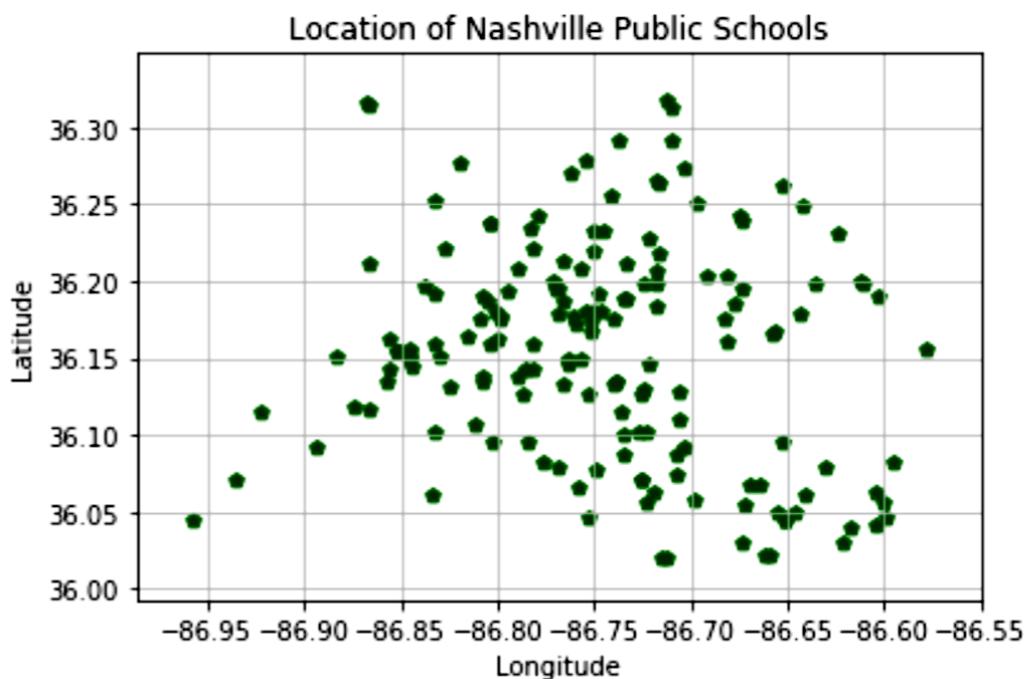
# Longitude and latitude



```
plt.scatter(schools.Longitude,  
           schools.Latitude,  
           c = 'darkgreen',  
           marker = 'p')  
  
plt.show()
```



```
plt.scatter(schools.Longitude, schools.Latitude,  
           c = 'darkgreen', marker = 'p')  
  
plt.xlabel('Longitude')  
plt.ylabel('Latitude')  
plt.title('Nashville Public Schools')  
  
plt.grid()  
  
plt.show()
```



# Extracting longitude and latitude

```
bus_stops.head()
```

Stop ID	StopName	Location
4431	MCC5_11	(36.16659, -86.781996)
588	CHA7AWN	(36.165, -86.78406)
590	CHA8AWN	(36.164393, -86.785451)
541	CXONGULC	(36.162249, -86.790464)
5231	7AVUNISM	(36.163822, -86.783791)

# Extracting longitude and latitude

```
bus_stops['lat'] = [loc[0] for loc in bus_stops.Location]
bus_stops['lng'] = [loc[1] for loc in bus_stops.Location]
bus_stops.head()
```

Stop ID	StopName	Location	lat	lng
4431	MCC5_11	(36.16659, -86.781996)	36.16659	-86.781996
588	CHA7AWN	(36.165, -86.78406)	36.165	-86.78406
590	CHA8AWN	(36.164393, -86.785451)	36.164393	-86.785451
541	CXONGULC	(36.162249, -86.790464)	36.162249	-86.790464
5231	7AVUNISM	(36.163822, -86.783791)	36.163822	-86.783791

# Extracting `lng` and `lat` with regular expressions

```
bus_stops2.head()
```

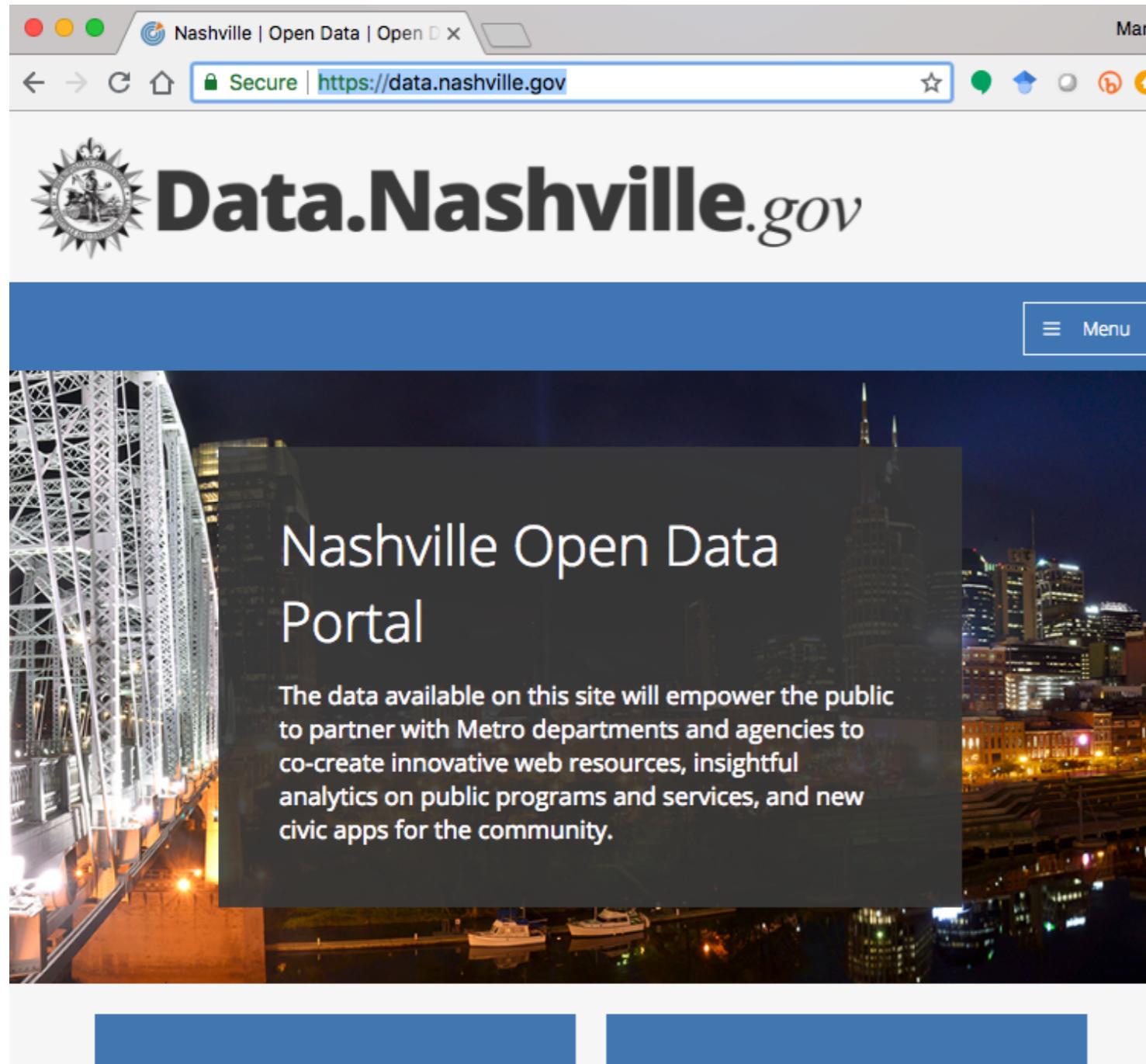
Stop ID	Location
4431	MCC - BAY 11\nNashville, TN\n(36.16659, -86.78199)
588	CHARLOTTE AVE\nNashville, TN\n(36.165, -86.78406)
590	CHARLOTTE AV\nNashville, TN\n(36.164393, -86.785451)
541	CHARLOTTE\nNashville, TN\n(36.162249, -86.790464)
5231	Nashville, TN\n(36.163822, -86.783791)

# Extracting lng and lat with regular expressions

```
lat_lng_pattern = re.compile(r'\((.*),\s*(.*))', flags=re.MULTILINE)
def extract_lat_lng(address):
    try:
        lat_lng_match = lat_lng_pattern.search(address)
        lat = float(lat_lng_match.group(1))
        lng = float(lat_lng_match.group(2))
        return (lat, lng)
    except:
        return (np.NaN, np.NaN)
```

```
lat_lngs = [extract_lat_lng(location) for location in \
            bus_stops2.loc[:, 'Location']]
bus_stops2['lat'] = [lat for lat, lng in lat_lngs]
bus_stops2['lng'] = [lng for lat, lng in lat_lngs]
```

# Nashville open data

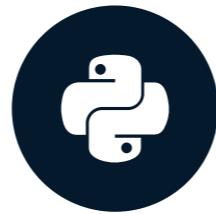


# **Let's practice!**

**VISUALIZING GEOSPATIAL DATA IN PYTHON**

# Geometries and shapefiles

VISUALIZING GEOSPATIAL DATA IN PYTHON

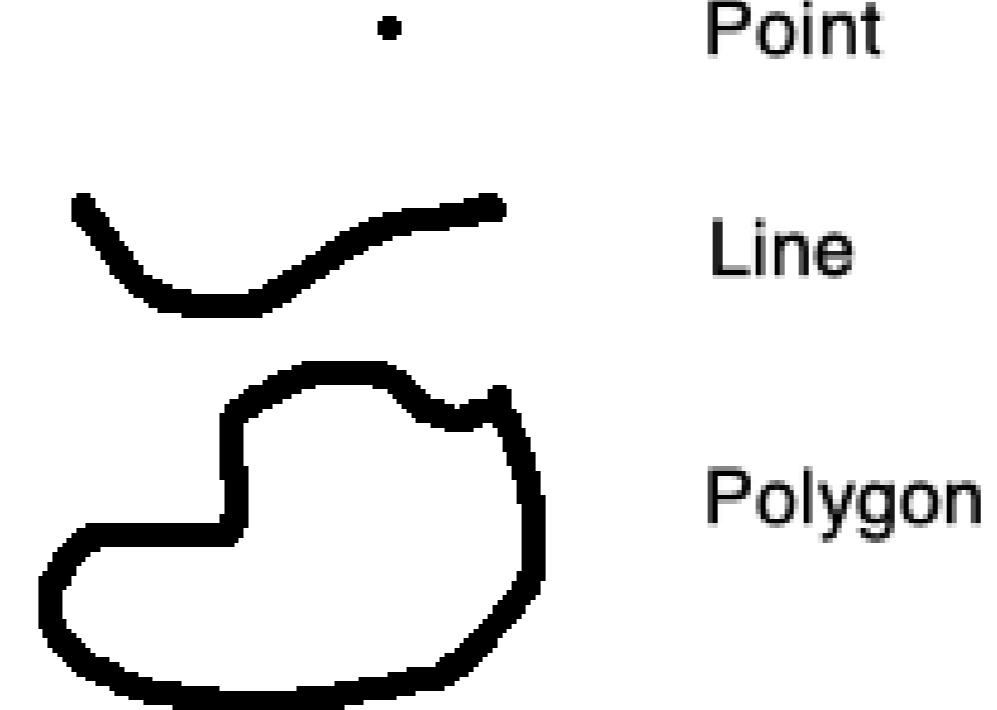


Mary van Valkenburg

Data Science Program Manager,  
Nashville Software School

# Shapefiles

Shapefiles store a special type of data known as *geometry*.



# Shapefile components

KEEP ALL THE FILES TOGETHER!

```
$ ls my_map_files/
```

```
my_map.dbf
```

```
my_map.shp
```

```
my_map.shx
```

- `my_map.shp` (contains the geometry)
- `my_map.dbf` (holds attributes for each geometry)
- `my_map.shx` (links the attributes to the geometry)

# geopandas

This code reads a shapefile into a GeoDataFrame and looks at the first few rows.

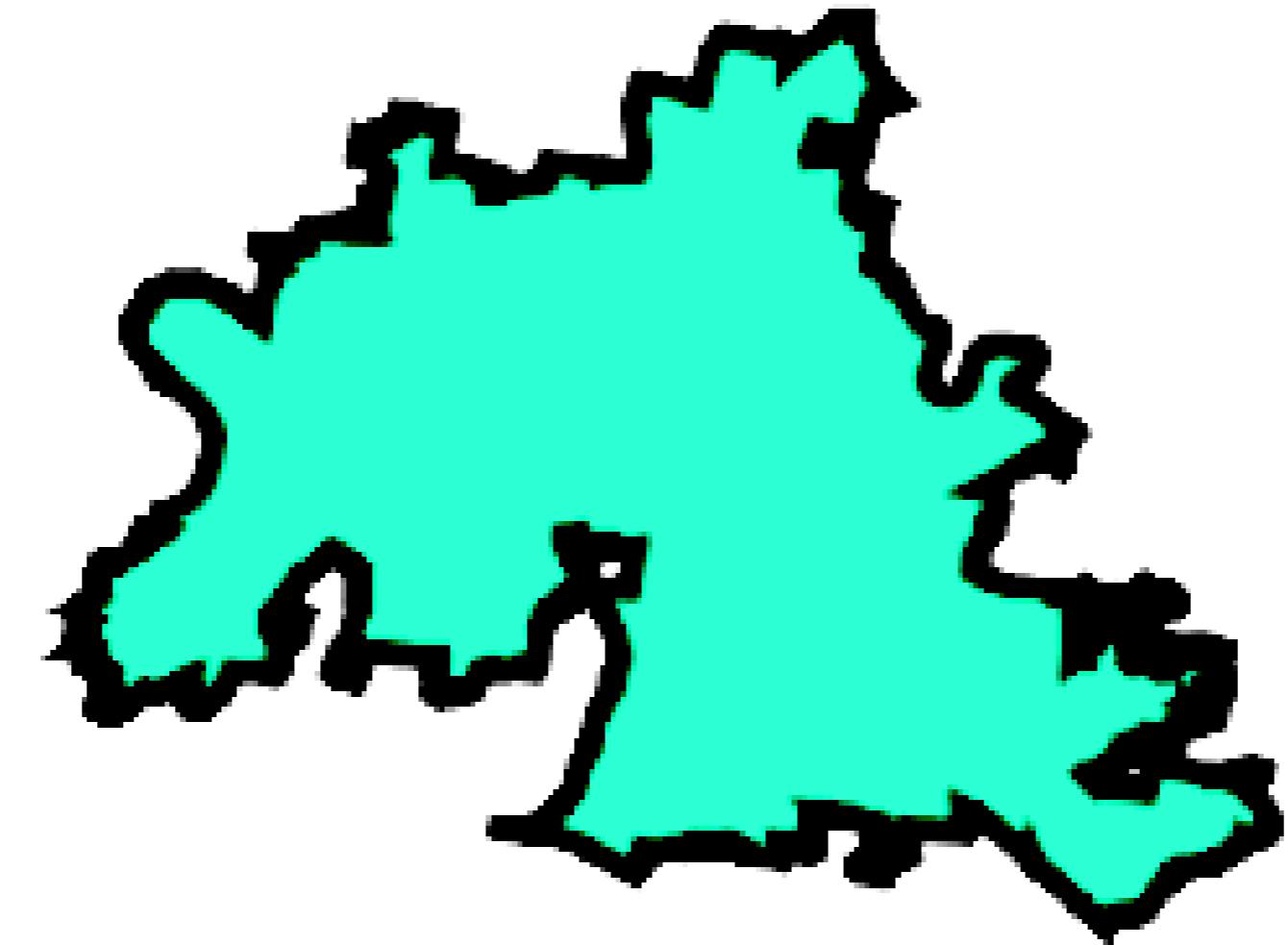
```
import geopandas as gpd

geo_df = gpd.read_file('My_Map_Files/my_map.shp')
geo_df.head()
```



# Viewing a geometry

```
service_district.loc[0, 'geometry']
```



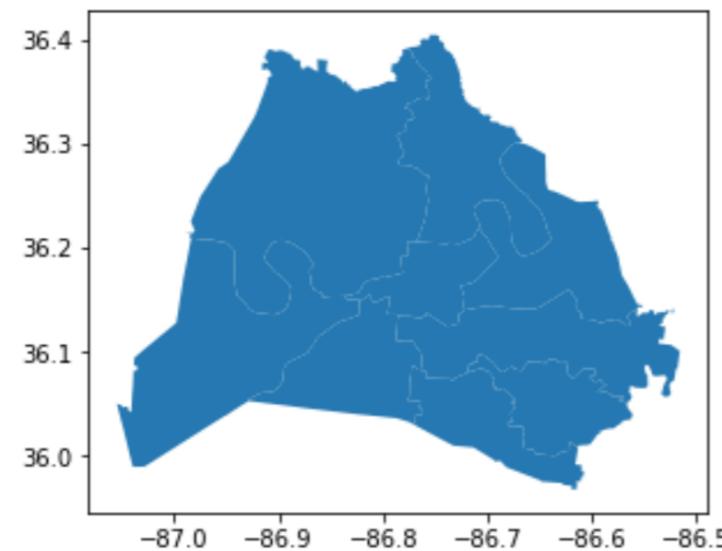
# Printing a geometry

```
print(service_district.loc[0, 'geometry'])
```

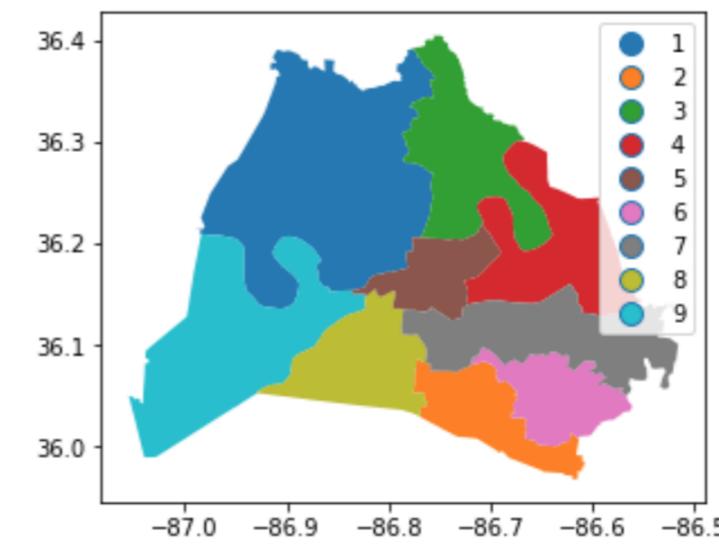
```
POLYGON ((-86.68680500011935 36.28670500013504,  
-86.68706099969657 36.28550299967364, -86.68709498823965 36.28511683351293,  
-86.68712691935902 36.28475404474551, -86.6871549990252 36.28443499969863,  
-86.68715025108719 36.28438104319917, -86.68708600011215 36.2836510002216,  
-86.6870599998375 36.28335400009232, -86.68683200030846 36.28073200026927,  
-86.68678671280243 36.2804916722591, -86.68668199966068 36.27993600019391,  
-86.686543000303 36.27920000021985, -86.68641799989246 36.27853199938513,  
-86.68600744248923 36.27759483150202, -86.68579942352289 36.27711998225582,  
-86.68482299948184 36.2748910007355, -86.68476799897849 36.27478700083996,  
-86.68372700043393 36.27281799971492, -86.6832880000829 36.27208000018629,  
-86.68313199902317 36.27181700012145, -86.68278700024624 36.27108100075766,  
-86.68257822861736 36.27077209799597, -86.68177585777893 36.2694062861527....)
```

# Plotting a GeoDataFrame

```
school_districts.plot()  
plt.show()
```



```
school_districts.plot(column =  
                      'district',  
                      legend = True)  
plt.show()
```

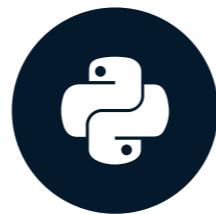


# **Let's practice!**

**VISUALIZING GEOSPATIAL DATA IN PYTHON**

# Putting it all together

VISUALIZING GEOSPATIAL DATA IN PYTHON



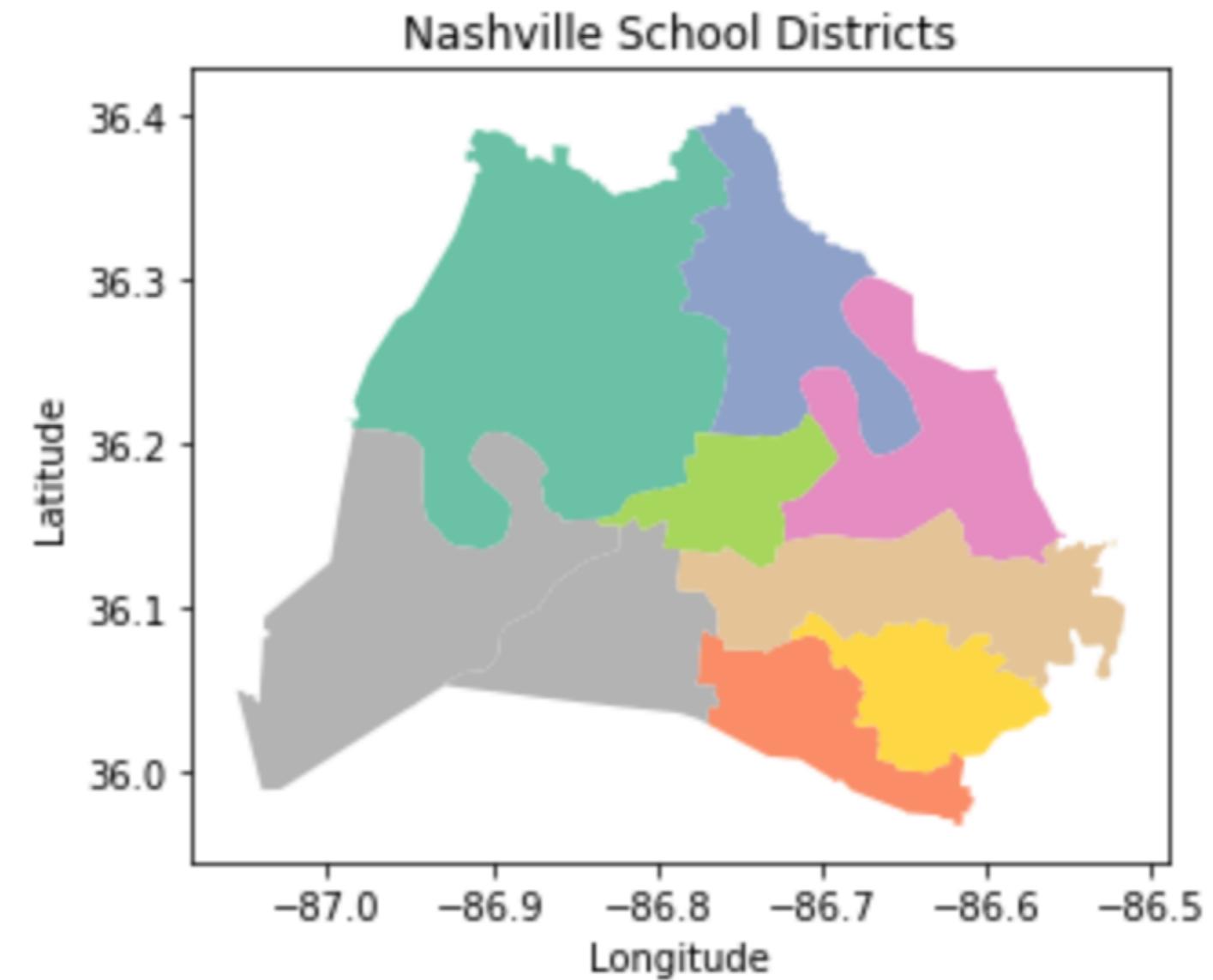
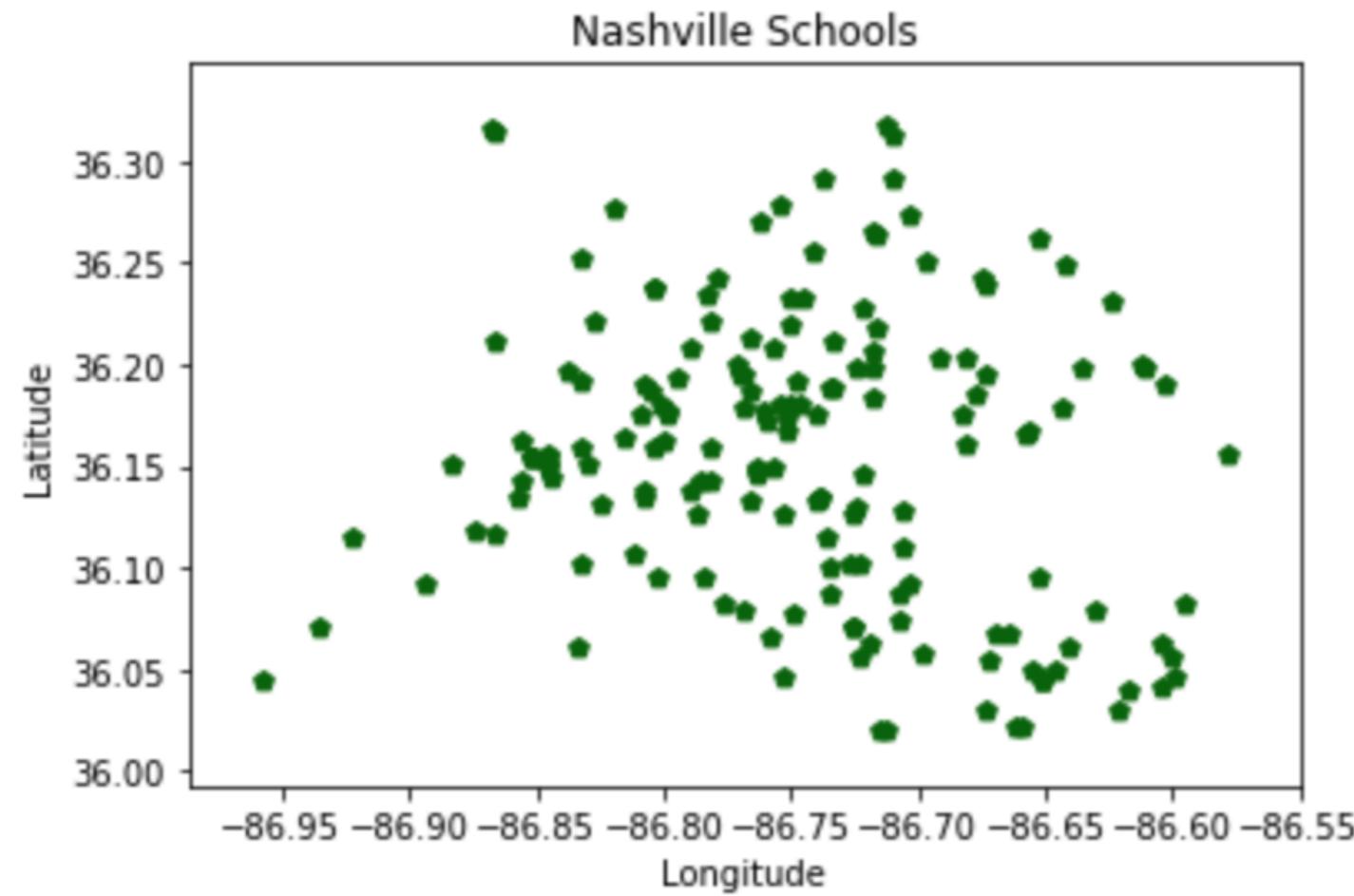
Mary van Valkenburg

Data Science Program Manager,  
Nashville Software School

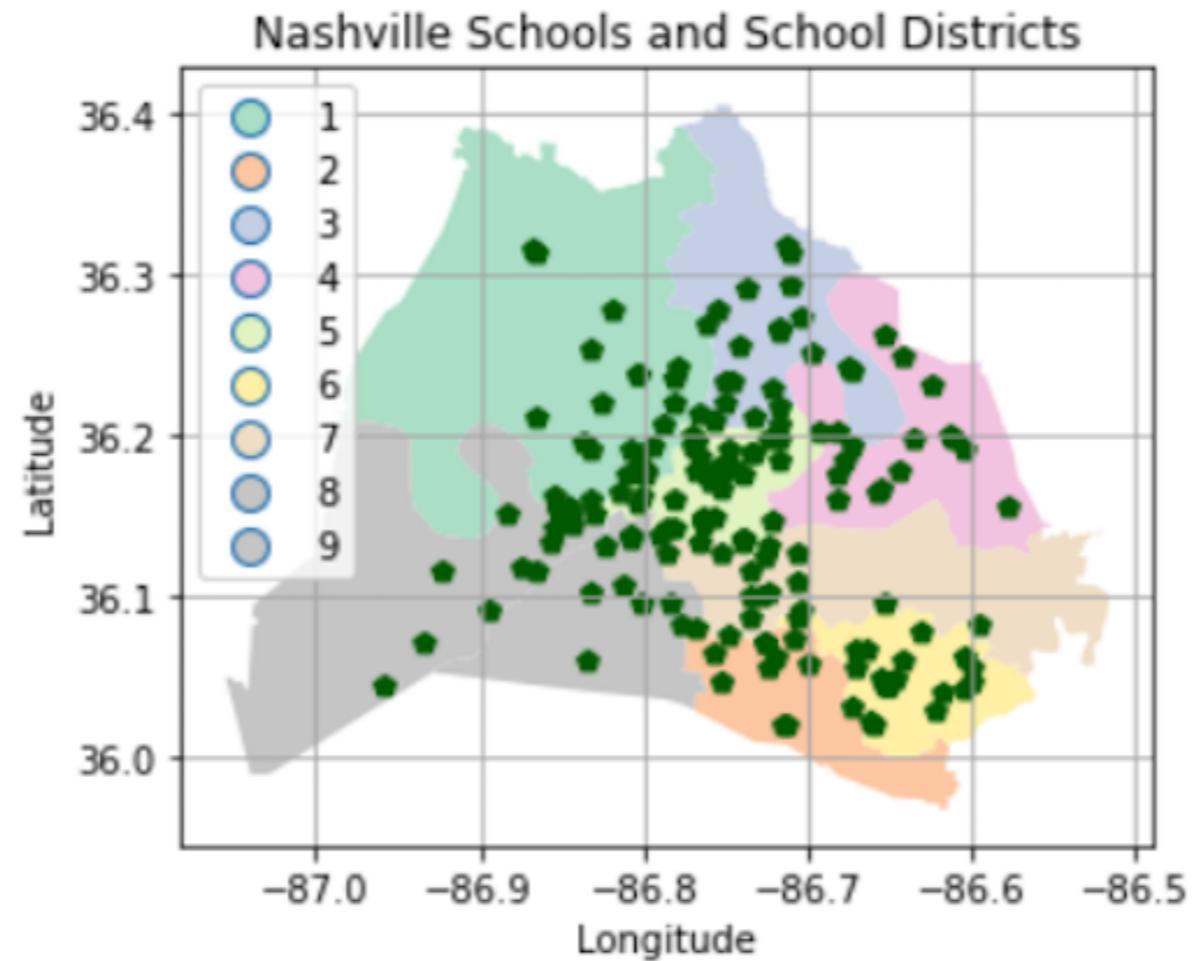
# Skills list

- Understanding longitude and latitude
- Extracting longitude and latitude
- Plotting points on scatterplot using longitude and latitude
- Styling scatterplots for better aesthetics and insight
- Plotting polygons from shapefiles

# Combining scatterplots and polygons



```
school_districts.plot(column = 'district', legend = True, cmap = 'Set2')
plt.scatter(schools.lng, schools.lat, marker = 'p', c = 'darkgreen')
plt.title('Nashville Schools and School Districts')
plt.show();
```



# **Let's practice!**

**VISUALIZING GEOSPATIAL DATA IN PYTHON**