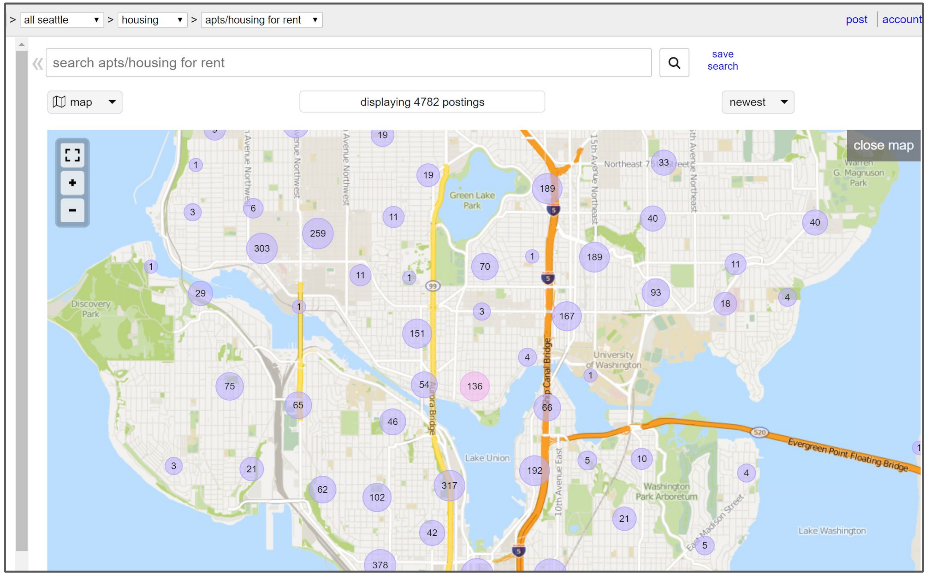
Bokeh and Folium

Python GIS Visualization

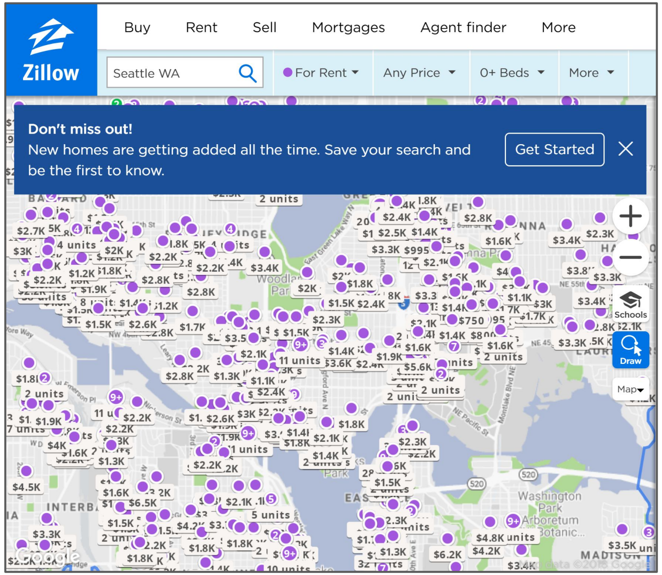
Graduate School Housing Search

Craigslist



displaying 4782 postings

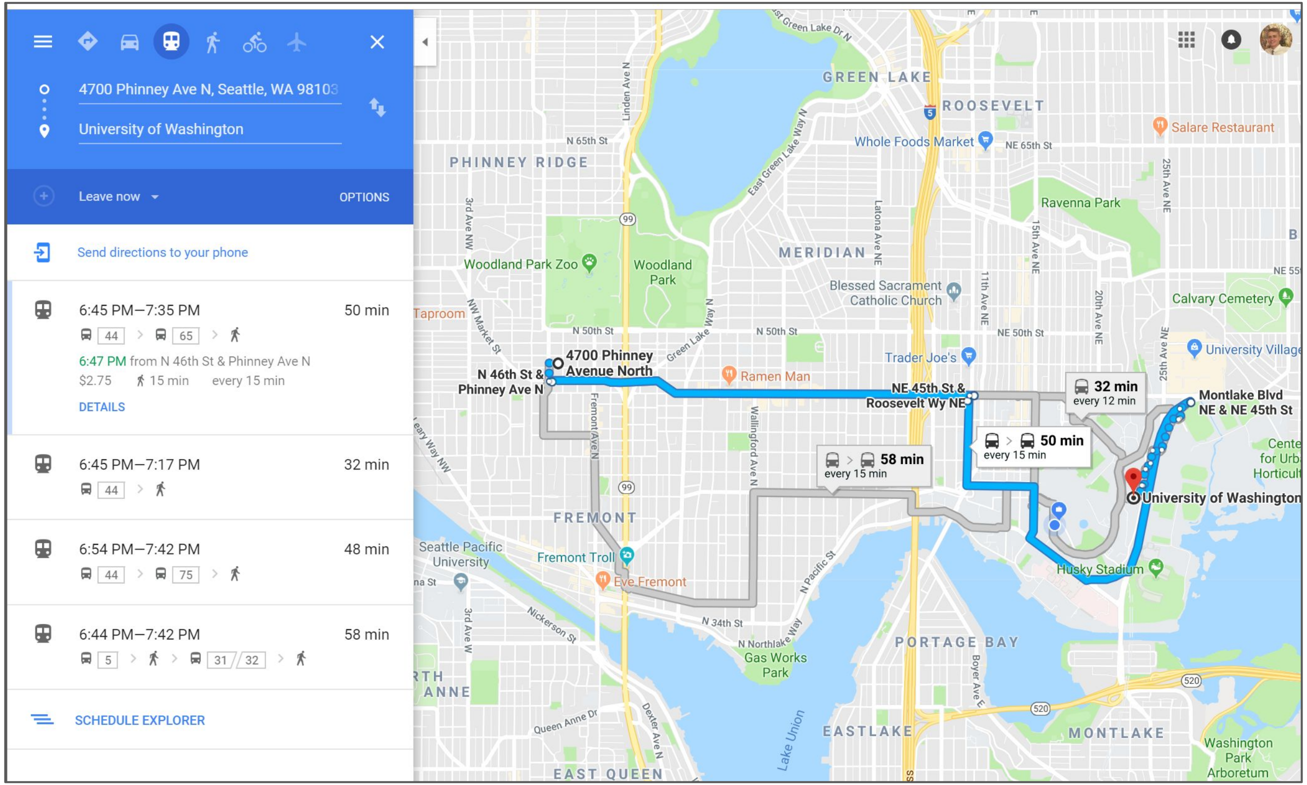
Zillow



Don't miss out!

New homes are getting added all the time. Save your search and be the first to know.

Graduate School Housing Search



4700 Phinney Ave N, Seattle, WA 98103

University of Washington

Leave now ▾ OPTIONS

Send directions to your phone

6:45 PM—7:35 PM 50 min
44 > 65 >
6:47 PM from N 46th St & Phinney Ave N
$2.75 15 min every 15 min
DETAILS

6:45 PM—7:17 PM 32 min
44 >

6:54 PM—7:42 PM 48 min
44 > 75 >

6:44 PM—7:42 PM 58 min
5 > > 31 / 32 >

SCHEDULE EXPLORER

GradPads

- Goal: Develop an interactive web tool to help graduate students find housing that aggregates information besides pricing

- User: Incoming/current graduate student

- Example use case: User browses neighborhoods to look for parks and crime data

GradPads - Visualization Goals

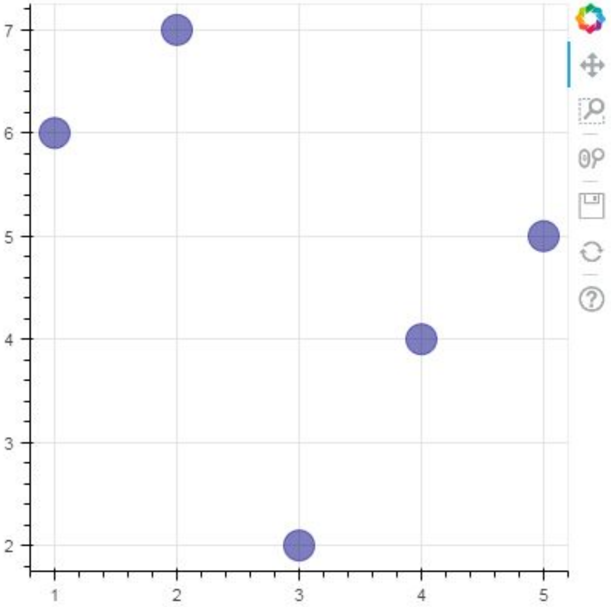
- Zip-code- to address-level specificity
- Interactive
    - Can toggle layers of prices on and off
- No user python
    - HTML output

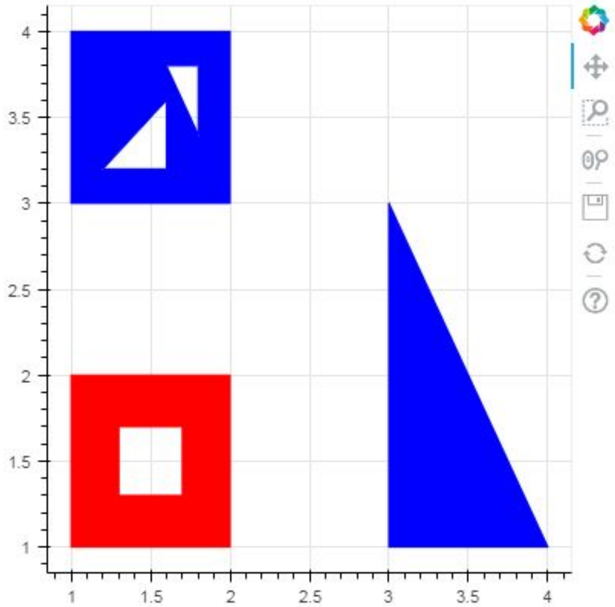
Bokeh

- Made up of two library components
    - Python library that generates JavaScript Object Notation (JSON)
    - Javascript library
        - Runs in the browser
        - Responsible for rendering and user interaction
        - “We write the JavaScript so you don’t have to!”

- Offers different levels of control
    - Bokeh.models - low-level
    - Bokeh.plotting - high-level

Bokeh.plotting - glyphs

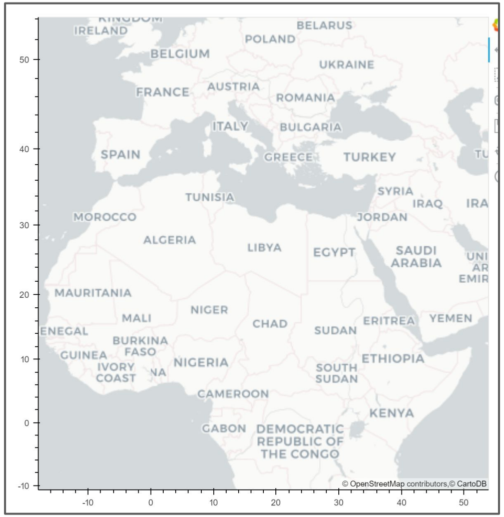




Bokeh - Mapping Geo Data

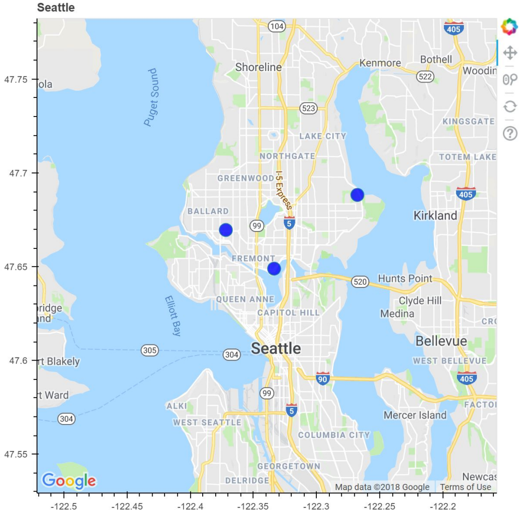
Tile provider maps

- Web Mercator Projection



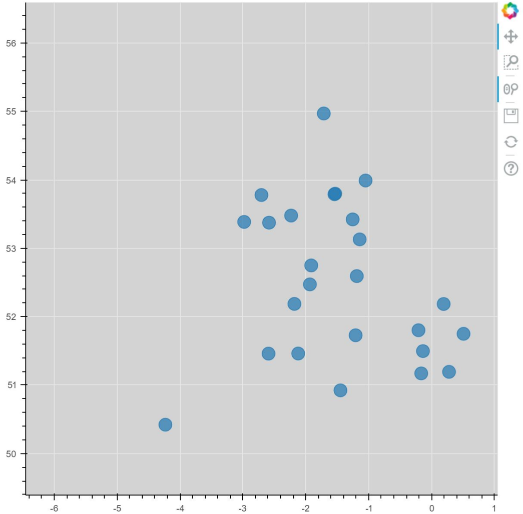
Google Maps

- Plots glyphs over Google Map

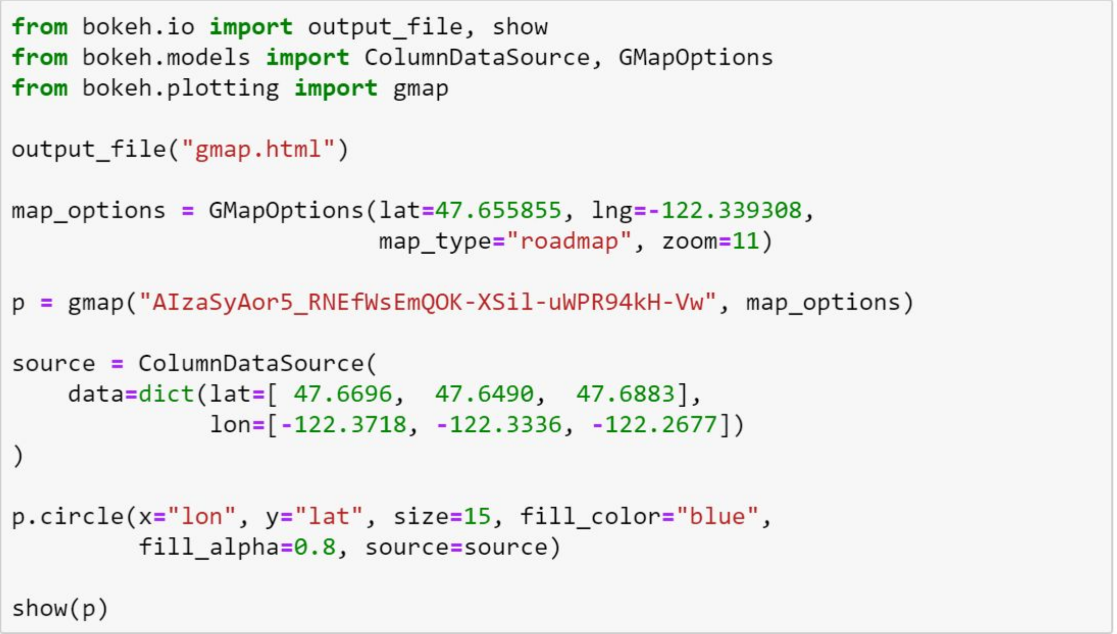


GeoJSON

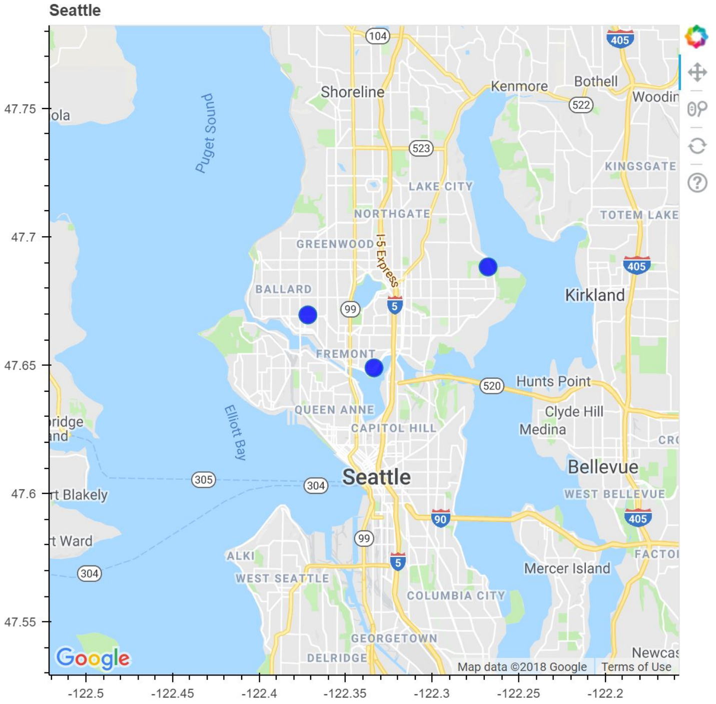
- Popular way to represent geographical features



Bokeh - Mapping Geo Data with Google Maps

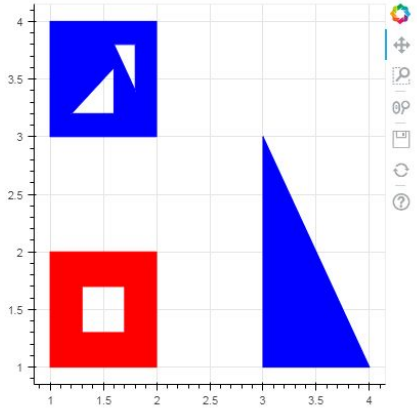


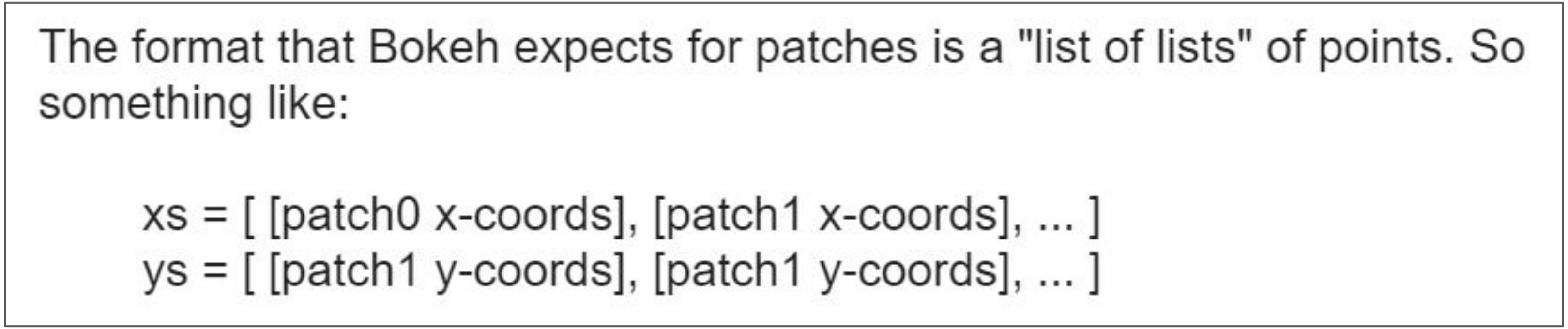
```
from bokeh.io import output_file, show
from bokeh.models import ColumnDataSource, GMapOptions
from bokeh.plotting import gmap

output_file("gmap.html")

map_options = GMapOptions(lat=47.655855, lng=-122.339308,
                          map_type="roadmap", zoom=11)

p = gmap("AIzaSyAor5_RNEfWsEmQOK-XSil-uWPR94kH-Vw", map_options)

source = ColumnDataSource(
    data=dict(lat=[ 47.6696,  47.6490,  47.6883],
              lon=[-122.3718, -122.3336, -122.2677])
)

p.circle(x="lon", y="lat", size=15, fill_color="blue",
         fill_alpha=0.8, source=source)

show(p)
```



Bokeh - Zip code polygons

- Zip code polygons available as shapefiles (.shp)
- No built-in functionality for shapefiles in Bokeh
- From a Google discussion with one of the developers:

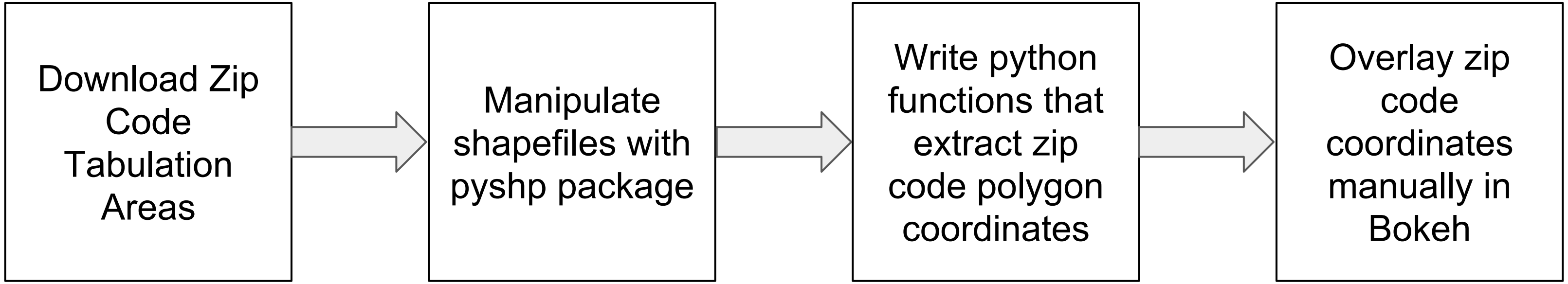




The format that Bokeh expects for patches is a "list of lists" of points. So something like:

```
    xs = [ [patch0 x-coords], [patch1 x-coords], ... ]
    ys = [ [patch1 y-coords], [patch1 y-coords], ... ]
```

Bokeh - Zip Code Workflow



Download Zip Code Tabulation Areas

Manipulate shapefiles with pyshp package

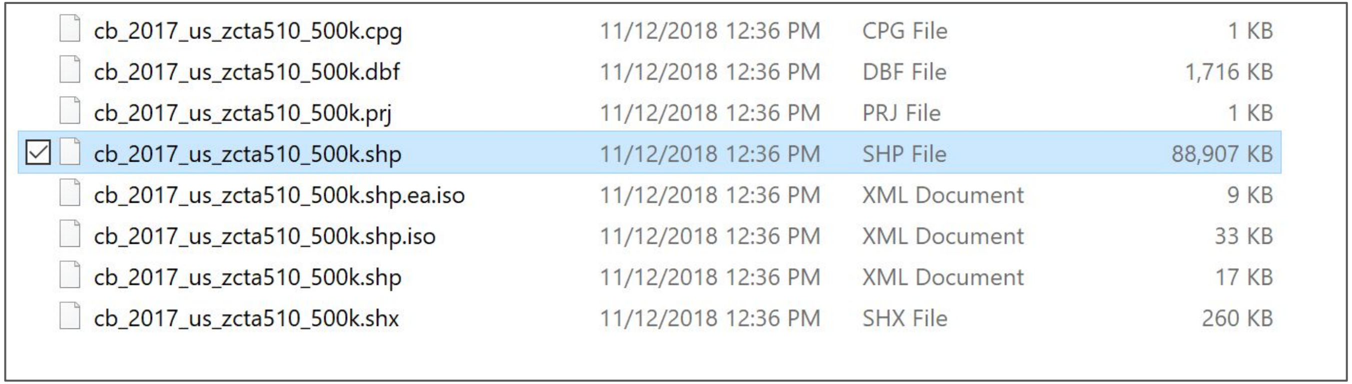
Write python functions that extract zip code polygon coordinates

Overlay zip code coordinates manually in Bokeh

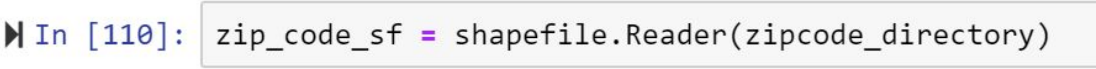
Bokeh - Zip Code Workflow

- Pyshp package
  - ”reads and writes ESRI Shapefiles in pure Python”
  - Reads in whole directory
    - .shp file contains points; .dbf contains record information

PyShp

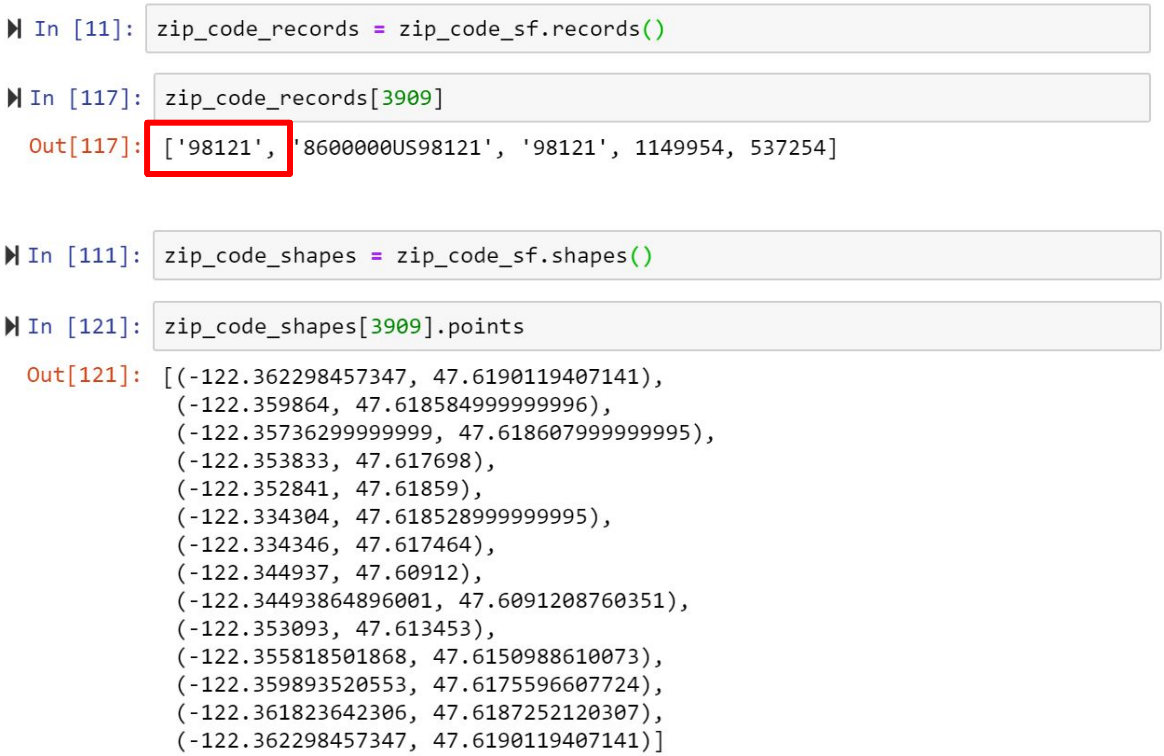


cb_2017_us_zcta510_500k.cpg	11/12/2018 12:36 PM	CPG File	1 KB
cb_2017_us_zcta510_500k.dbf	11/12/2018 12:36 PM	DBF File	1,716 KB
cb_2017_us_zcta510_500k.prj	11/12/2018 12:36 PM	PRJ File	1 KB
☑ cb_2017_us_zcta510_500k.shp	11/12/2018 12:36 PM	SHP File	88,907 KB
cb_2017_us_zcta510_500k.shp.ea.iso	11/12/2018 12:36 PM	XML Document	9 KB
cb_2017_us_zcta510_500k.shp.iso	11/12/2018 12:36 PM	XML Document	33 KB
cb_2017_us_zcta510_500k.shp	11/12/2018 12:36 PM	XML Document	17 KB
cb_2017_us_zcta510_500k.shx	11/12/2018 12:36 PM	SHX File	260 KB



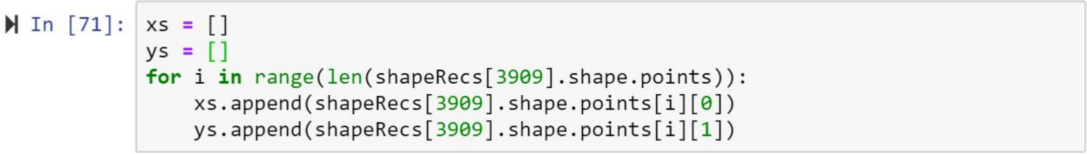
```
In [110]: zip_code_sf = shapefile.Reader(zipcode_directory)
```

Bokeh - Zip Code workflow



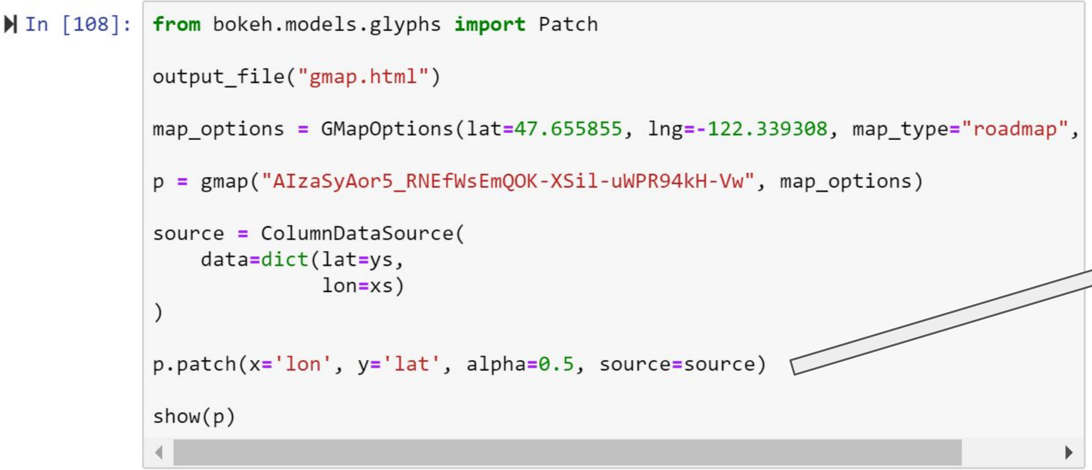
```
In [11]: zip_code_records = zip_code_sf.records()

In [117]: zip_code_records[3909]

Out[117]: ['98121', '8600000US98121', '98121', 1149954, 537254]

In [111]: zip_code_shapes = zip_code_sf.shapes()

In [121]: zip_code_shapes[3909].points

Out[121]: [(-122.362298457347, 47.6190119407141),
           (-122.359864, 47.618584999999996),
           (-122.35736299999999, 47.618607999999995),
           (-122.353833, 47.617698),
           (-122.352841, 47.61859),
           (-122.334304, 47.618528999999995),
           (-122.334346, 47.617464),
           (-122.344937, 47.60912),
           (-122.34493864896001, 47.6091208760351),
           (-122.353093, 47.613453),
           (-122.355818501868, 47.6150988610073),
           (-122.359893520553, 47.6175596607724),
           (-122.361823642306, 47.6187252120307),
           (-122.362298457347, 47.6190119407141)]
```

Bokeh - Zip Code Polygon Workflow



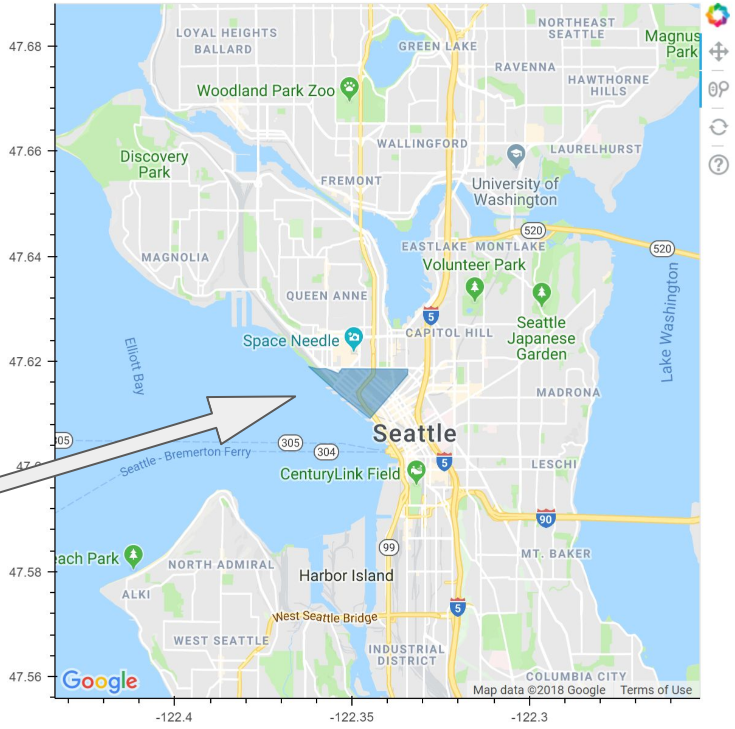
In [71]:

```
xs = []
ys = []
for i in range(len(shapeRecs[3909].shape.points)):
    xs.append(shapeRecs[3909].shape.points[i][0])
    ys.append(shapeRecs[3909].shape.points[i][1])
```



In [108]:

```
from bokeh.models.glyphs import Patch

output_file("gmap.html")

map_options = GMapOptions(lat=47.655855, lng=-122.339308, map_type="roadmap",

p = gmap("AIzaSyAor5_RNEfWsEmQOK-XSil-uWPR94kH-Vw", map_options)

source = ColumnDataSource(
    data=dict(lat=ys,
              lon=xs)
)

p.patch(x='lon', y='lat', alpha=0.5, source=source)

show(p)
```



Bokeh - Summary

- Pros
  - Easily interactive
  - HTML output
  - Extensive control over glyphs

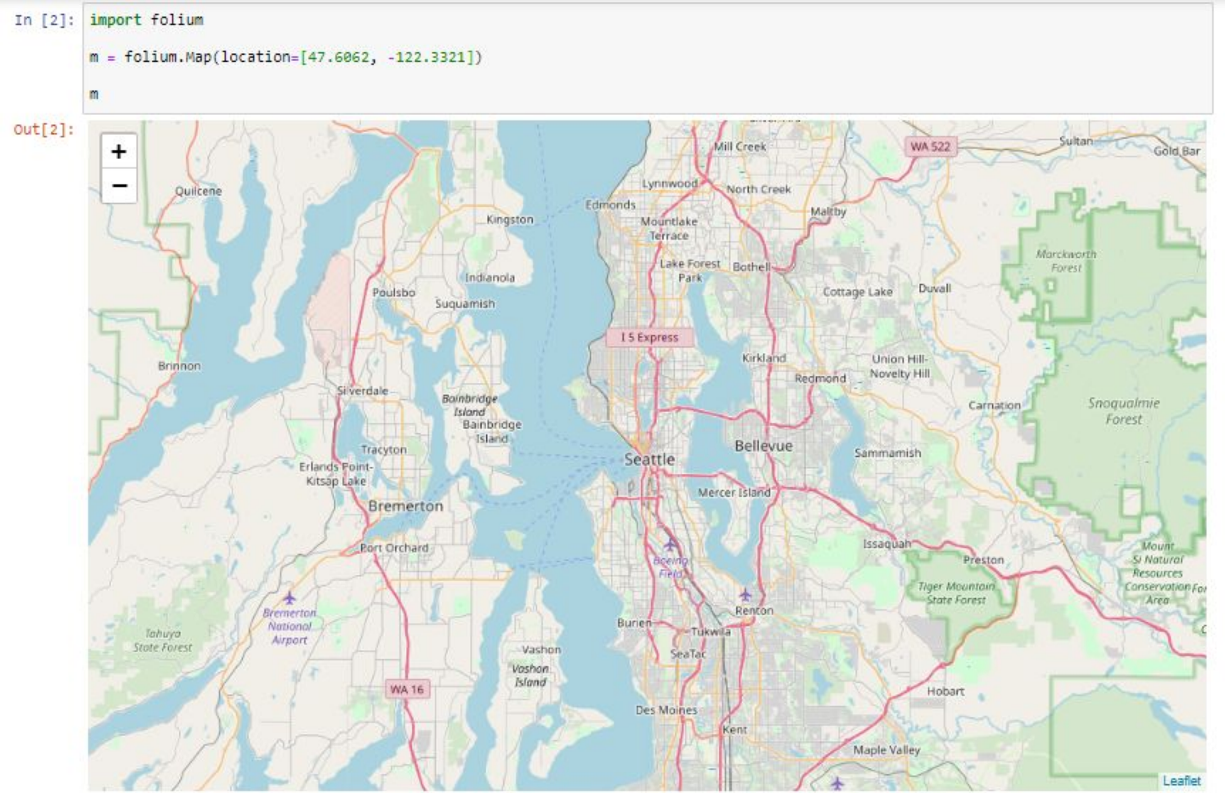
- Cons
  - Geographical data support is not very developed
  - Requires download of more packages for geo data

Folium

- Python library that combines Python’s data manipulation features with Leaflet.js’s mapping ability
    - Highly specific purpose that aligns with GradPads’s goals
    - Support for image overlays as well as GeoJSON and TopoJSON formats

- Can run from jupyter notebook
    - Ability to create interactive and customizeable maps
    - Can also render and export as HTML depending on users’ level of experience / need

Folium - Using Maps as Base Layers

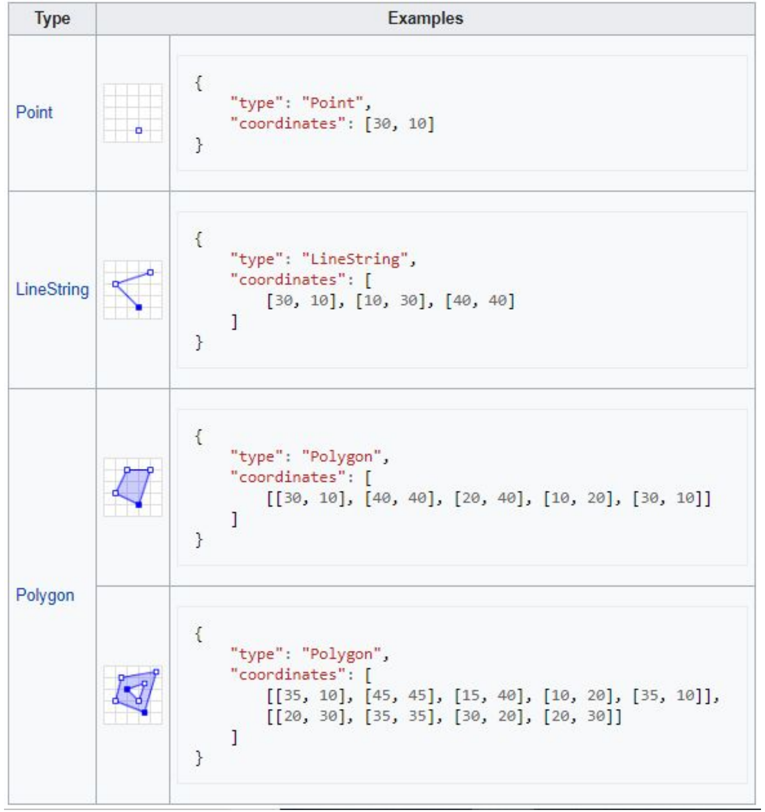


```
In [2]: import folium

        m = folium.Map(location=[47.6062, -122.3321])

        m
```

Out[2]:

Folium - GeoJSON Data Overlays

- GeoJSON - format based on JSON (JavaScript Object Notation)
  - Designed to represent geographical features along with non-spatial attributes
  - Contains points, LineStrings, and polygons



Type	Examples	

Point		`{`
		`    "type": "Point",`
		`    "coordinates": [30, 10]`
		`}`

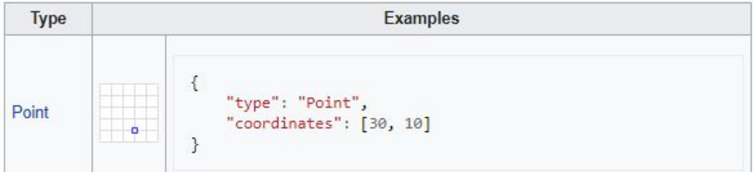
LineString		`{`
		`    "type": "LineString",`
		`    "coordinates": [`
		`        [30, 10], [10, 30], [40, 40]`
		`    ]`
		`}`

Polygon		`{`
		`    "type": "Polygon",`
		`    "coordinates": [`
		`        [[30, 10], [40, 40], [20, 40], [10, 20], [30, 10]]`
		`    ]`
		`}`

		`{`
		`    "type": "Polygon",`
		`    "coordinates": [`
		`        [[35, 10], [45, 45], [15, 40], [10, 20], [35, 10]],`
		`        [[20, 30], [35, 35], [30, 20], [20, 30]]`
		`    ]`
		`}`

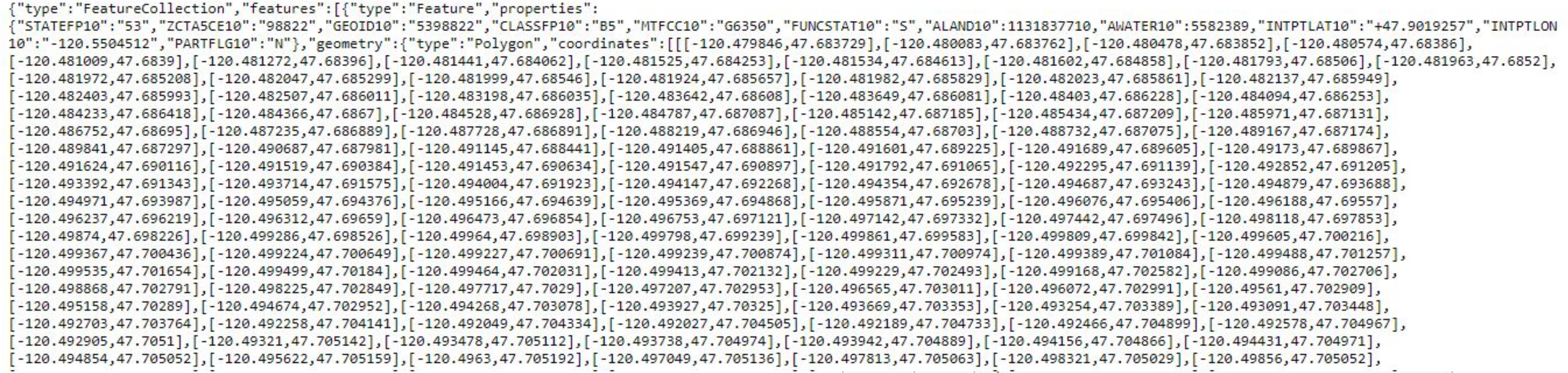
Folium - GeoJSON Data Overlays

- GeoJSON - format based on JSON (JavaScript Object Notation)
    - Designed to represent geographical

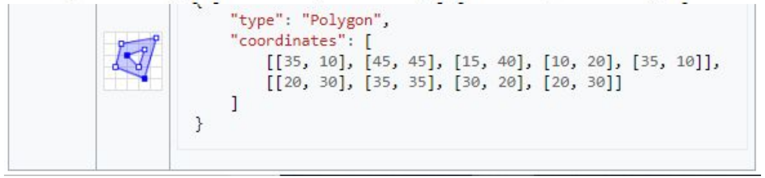


Type	Examples

Point	{
	    "type": "Point",
	    "coordinates": [30, 10]
	}

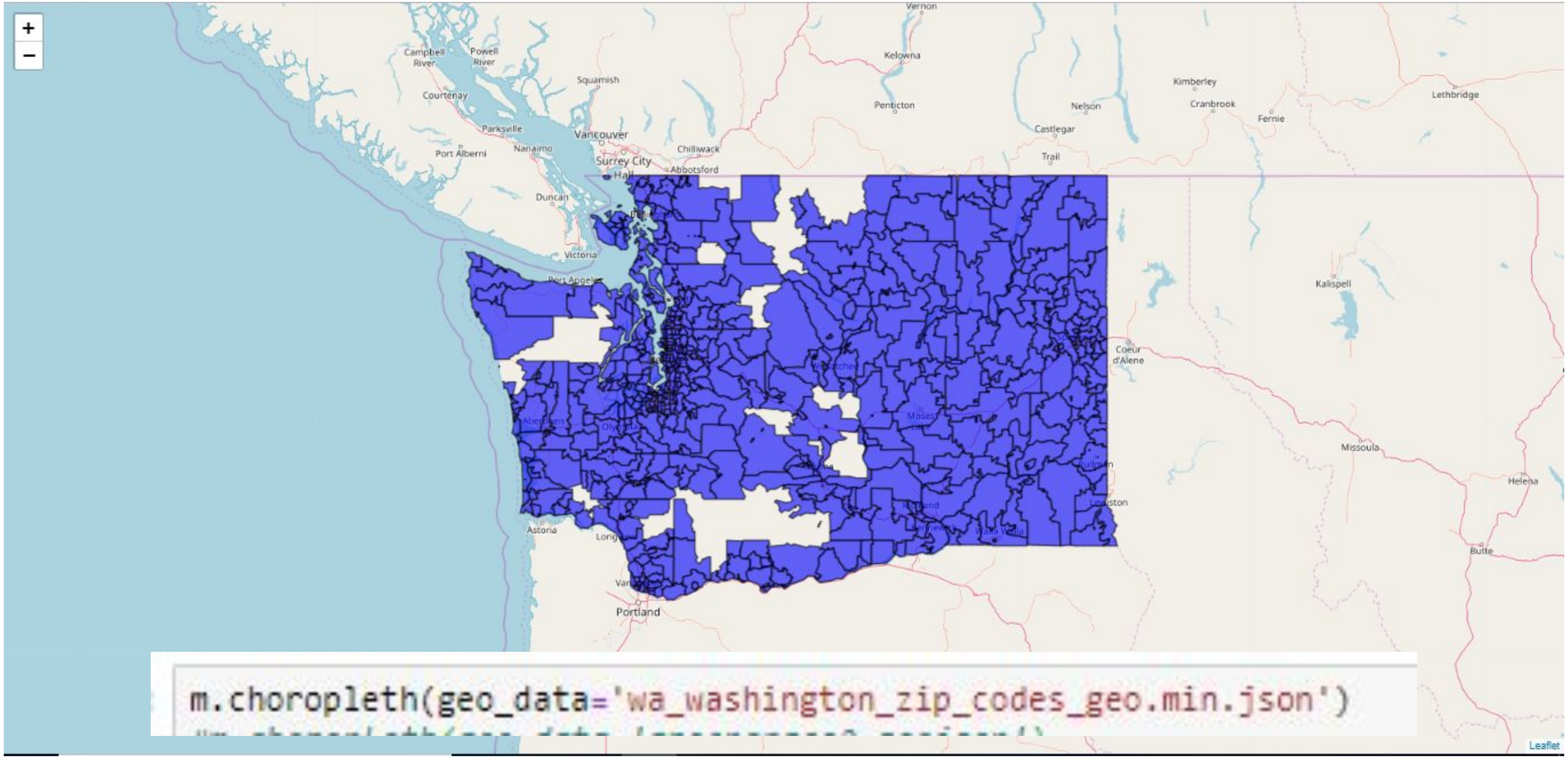


```
{"type":"FeatureCollection","features":[{"type":"Feature","properties":
{"STATEFP10":"53","ZCTA5CE10":"98822","GEOID10":"5398822","CLASSFP10":"B5","MTFCC10":"G6350","FUNCSTAT10":"S","ALAND10":1131837710,"AWATER10":5582389,"INTPTLAT10":"+47.9019257","INTPTLON
10":"-120.5504512","PARTFLG10":"N"},"geometry":{"type":"Polygon","coordinates":[[[-120.479846,47.683729],[-120.480083,47.683762],[-120.480478,47.683852],[-120.480574,47.68386],
[-120.481009,47.6839],[-120.481272,47.68396],[-120.481441,47.684062],[-120.481525,47.684253],[-120.481534,47.684613],[-120.481602,47.684858],[-120.481793,47.68506],[-120.481963,47.6852],
[-120.481972,47.685208],[-120.482047,47.685299],[-120.481999,47.68546],[-120.481924,47.685657],[-120.481982,47.685829],[-120.482023,47.685861],[-120.482137,47.685949],
[-120.482403,47.685993],[-120.482507,47.686011],[-120.483198,47.686035],[-120.483642,47.68608],[-120.483649,47.686081],[-120.48403,47.686228],[-120.484094,47.686253],
[-120.484233,47.686418],[-120.484366,47.6867],[-120.484528,47.686928],[-120.484787,47.687087],[-120.485142,47.687185],[-120.485434,47.687209],[-120.485971,47.687131],
[-120.486752,47.68695],[-120.487235,47.686889],[-120.487728,47.686891],[-120.488219,47.686946],[-120.488554,47.68703],[-120.488732,47.687075],[-120.489167,47.687174],
[-120.489841,47.687297],[-120.490687,47.687981],[-120.491145,47.688441],[-120.491405,47.688861],[-120.491601,47.689225],[-120.491689,47.689605],[-120.49173,47.689867],
[-120.491624,47.690116],[-120.491519,47.690384],[-120.491453,47.690634],[-120.491547,47.690897],[-120.491792,47.691065],[-120.492295,47.691139],[-120.492852,47.691205],
[-120.493392,47.691343],[-120.493714,47.691575],[-120.494004,47.691923],[-120.494147,47.692268],[-120.494354,47.692678],[-120.494687,47.693243],[-120.494879,47.693688],
[-120.494971,47.693987],[-120.495059,47.694376],[-120.495166,47.694639],[-120.495369,47.694868],[-120.495871,47.695239],[-120.496076,47.695406],[-120.496188,47.69557],
[-120.496237,47.696219],[-120.496312,47.69659],[-120.496473,47.696854],[-120.496753,47.697121],[-120.497142,47.697332],[-120.497442,47.697496],[-120.498118,47.697853],
[-120.49874,47.698226],[-120.499286,47.698526],[-120.49964,47.698903],[-120.499798,47.699239],[-120.499861,47.699583],[-120.499809,47.699842],[-120.499605,47.700216],
[-120.499367,47.700436],[-120.499224,47.700649],[-120.499227,47.700691],[-120.499239,47.700874],[-120.499311,47.700974],[-120.499389,47.701084],[-120.499488,47.701257],
[-120.499535,47.701654],[-120.499499,47.70184],[-120.499464,47.702031],[-120.499413,47.702132],[-120.499229,47.702493],[-120.499168,47.702582],[-120.499086,47.702706],
[-120.498868,47.702791],[-120.498225,47.702849],[-120.497717,47.7029],[-120.497207,47.702953],[-120.496565,47.703011],[-120.496072,47.702991],[-120.49561,47.702909],
[-120.495158,47.70289],[-120.494674,47.702952],[-120.494268,47.703078],[-120.493927,47.70325],[-120.493669,47.703353],[-120.493254,47.703389],[-120.493091,47.703448],
[-120.492703,47.703764],[-120.492258,47.704141],[-120.492049,47.704334],[-120.492027,47.704505],[-120.492189,47.704733],[-120.492466,47.704899],[-120.492578,47.704967],
[-120.492905,47.7051],[-120.49321,47.705142],[-120.493478,47.705112],[-120.493738,47.704974],[-120.493942,47.704889],[-120.494156,47.704866],[-120.494431,47.704971],
[-120.494854,47.705052],[-120.495622,47.705159],[-120.4963,47.705192],[-120.497049,47.705136],[-120.497813,47.705063],[-120.498321,47.705029],[-120.49856,47.705052],
```



	"type": "Polygon",
	"coordinates": [
	    [[35, 10], [45, 45], [15, 40], [10, 20], [35, 10]],
	    [[20, 30], [35, 35], [30, 20], [20, 30]]
	]
	}

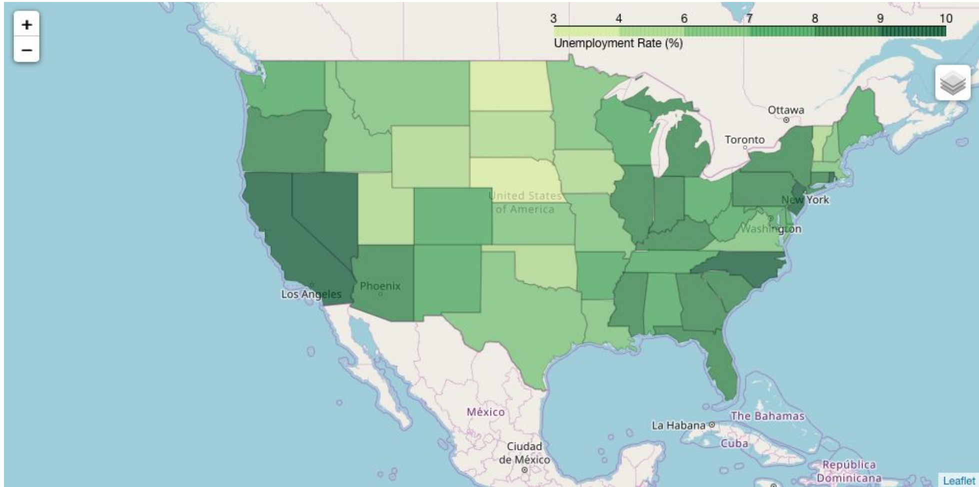
Folium - GeoJSON Data Overlays



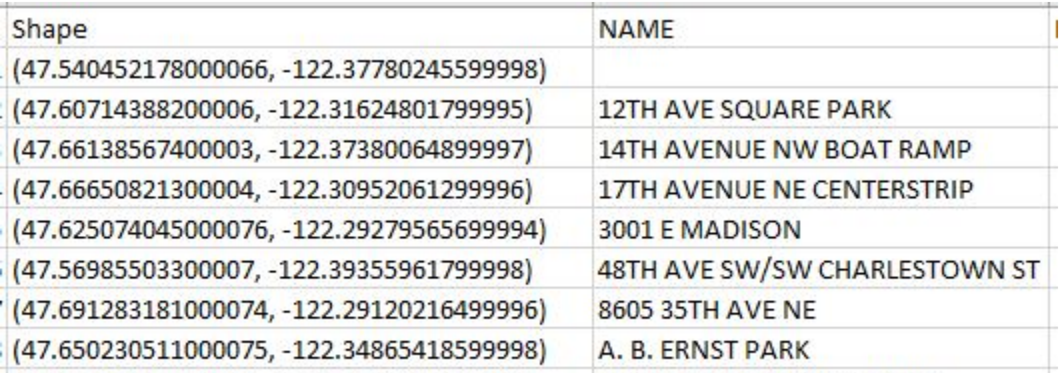
```
m.choropleth(geo_data='wa_washington_zip_codes_geo.min.json')
```

Folium - Choropleth Maps

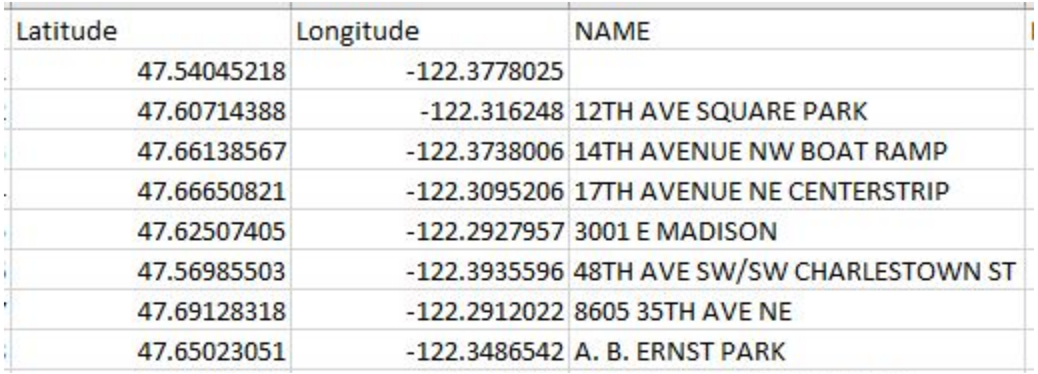
- Choropleth - a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map
- Folium creates choropleth by joining data in a dataframe to geographical data (like GeoJSON data) and overlaying heat map-type layers onto the map base layer



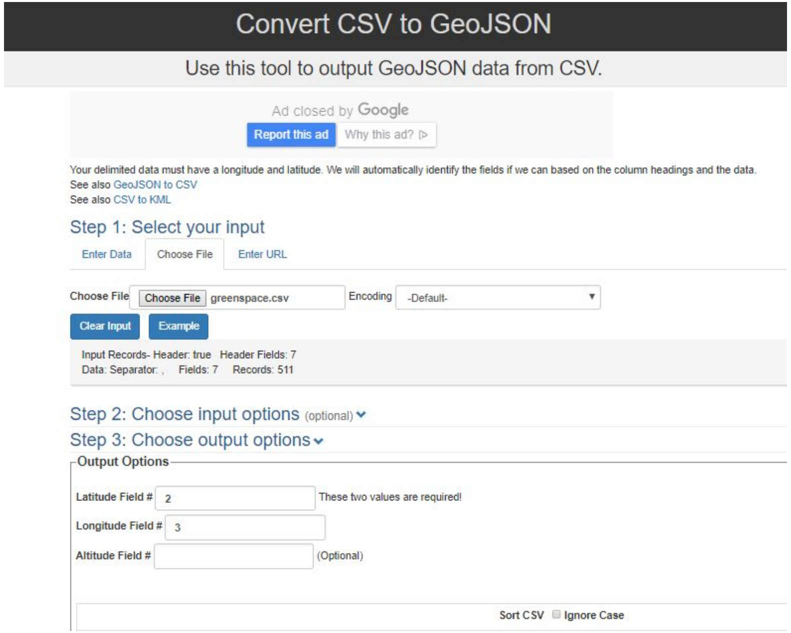
Folium - Data Formatting



Shape	NAME
(47.54045217800066, -122.37780245599998)	
(47.60714388200006, -122.31624801799995)	12TH AVE SQUARE PARK
(47.66138567400003, -122.37380064899997)	14TH AVENUE NW BOAT RAMP
(47.66650821300004, -122.30952061299996)	17TH AVENUE NE CENTERSTRIP
(47.625074045000076, -122.29279565699994)	3001 E MADISON
(47.56985503300007, -122.39355961799998)	48TH AVE SW/SW CHARLESTOWN ST
(47.691283181000074, -122.29120216499996)	8605 35TH AVE NE
(47.650230511000075, -122.34865418599998)	A. B. ERNST PARK



Latitude	Longitude	NAME
47.54045218	-122.3778025	
47.60714388	-122.316248	12TH AVE SQUARE PARK
47.66138567	-122.3738006	14TH AVENUE NW BOAT RAMP
47.66650821	-122.3095206	17TH AVENUE NE CENTERSTRIP
47.62507405	-122.2927957	3001 E MADISON
47.56985503	-122.3935596	48TH AVE SW/SW CHARLESTOWN ST
47.69128318	-122.2912022	8605 35TH AVE NE
47.65023051	-122.3486542	A. B. ERNST PARK



Convert CSV to GeoJSON

Use this tool to output GeoJSON data from CSV.

Your delimited data must have a longitude and latitude. We will automatically identify the fields if we can based on the column headings and the data.
See also GeoJSON to CSV
See also CSV to KML

Step 1: Select your input

Choose File [Choose File] greenspace.csv Encoding -Default-

Input Records- Header: true Header Fields: 7
Data: Separator: , Fields: 7 Records: 511

Step 2: Choose input options (optional)

Step 3: Choose output options

Output Options

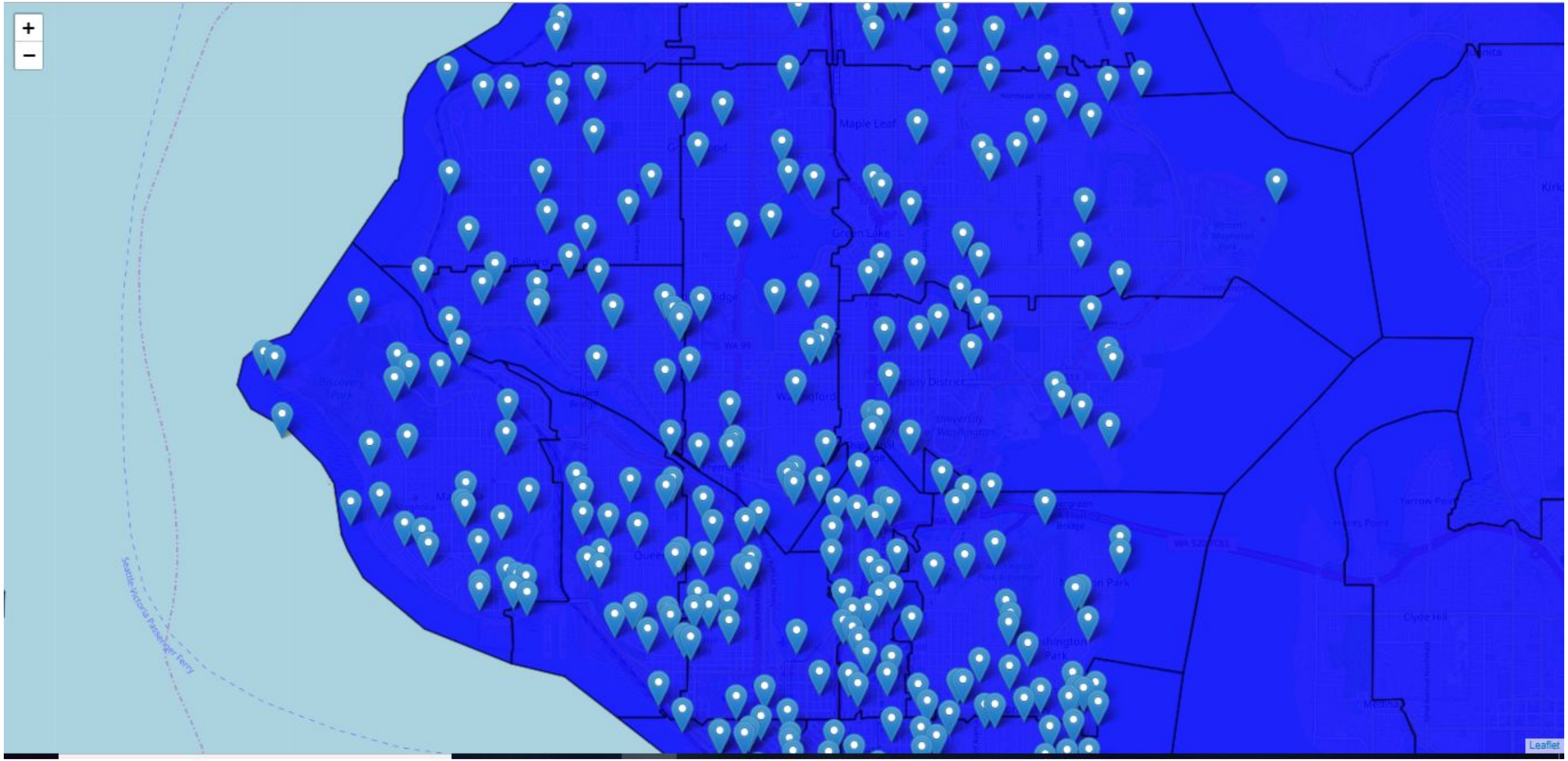
Latitude Field # 2 These two values are required!

Longitude Field # 3

Altitude Field # (Optional)

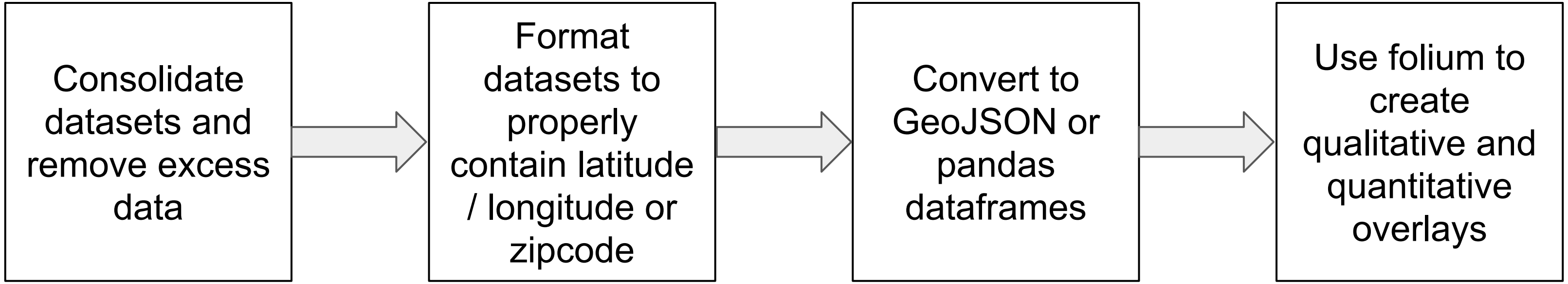
Sort CSV □ Ignore Case

Folium - More Data Overlays



Folium - Workflow

- Primarily formatting / manipulating datasets
  - Most available in CSV format
  - Also need to package data into dataframes with zipcode data vs. variable of interest
    - Need to make sure uniform set of zipcodes are used, remove unnecessary data
  - Will require some conversion of CSV, etc. type data into GeoJSON format for qualitative overlays



Consolidate datasets and remove excess data

Format datasets to properly contain latitude / longitude or zipcode

Convert to GeoJSON or pandas dataframes

Use folium to create qualitative and quantitative overlays

Folium - Summary

- Pros
  - Interactive interface
  - HTML output
  - Relatively easy to interface with our datasets and types
  - Good support for geographical data

- Cons
  - Somewhat limited in scope - only map functionality
  - Doesn’t interface seamlessly with jupyter? Browser? (Can’t display complex maps in the notebook)
  - Does require some manipulation of data to comply with several different formats - GeoJSON, pandas dataframes, etc.

http://www.convertcsv.com/csv-to-geojson.htm

Greenspace data
https://catalog.data.gov/dataset?organization_type=City+Government&publisher=data.seattle.gov&tags=green-space

JSON Seattle zip codes

https://catalog.data.gov/dataset/seattle-zip-codes-ebab5/resource/5bb72c5f-b9ee-4dc5-852b-0f3a51bda638

GeoJSON WA zip codes

https://raw.githubusercontent.com/OpenDataDE/State-zip-code-GeoJSON/master/