Chapter 11. 지도 시각화

- json.load(open()): open 또는 load에 빈칸 주의, GEOjson 파일을 읽을 땐 먼저 open()으로 파일을 열고, json.load()를 해줘야 함
- astype(): 강제 캐스팅 ~ 지도에 활용하기 위해서는 int와 같은 숫자 타입이 아닌 문자 타입인 str로 강제 형변환을 해줘야 함
- folium.Map()
- 304p ~ tiles 파라미터 빈칸
- 305p ~ .add_to() 함수, key_on, columns, data, geo_data의 차이
- 305p ~ 단계 구분도 함수: Choropleth()

- 단계 구분도(choropleth map): 지역별 통계치를 색깔 차이로 표현한 지도
  - 단계 구분도를 만들면 통계치가 지역별로 어떻게 다른지 쉽게 이해할 수 있음
  - 단계 구분도를 만들면 최적의 장소를 정하는 데 유용하게 활용할 수 있음
  - 단계 구분도를 만들기 위해서는 folium 패키지를 이용하여 만들 수 있음
    - 1) 아나콘다 프롬프트 환경에서 pip install folium 을 실행하여 패키지를 설치한다.
    - 2) 주피터 노트북을 실행하고 노트북 파일에서 import folium 을 통해 해당 패키지를 불러온다.
    - 3) folium.Choropleth() 과 같은 식으로 해당 패키지 내의 함수를 사용한다.
  - 단계 구분도 만드는 순서
    - 1) 배경 지도 만들기
    - 2) 단계 구분도 만들기
    - 3) 계급 구간 정하기
    - 4) 디자인 수정하기(+ 구 경계선 추가하기)

- folium.Map(): 배경 지도를 만드는 함수
  - location: 지도의 중심 위도, 경도 좌표를 입력 (e.g. location = [35.95, 127.7])
  - zoom_start: 지도를 확대할 정도를 입력 (e.g. zoom_start = 8)
  - tiles: 지도의 종류를 입력 (e.g. tiles = 'cartodbpositron')
  - 참고로 width와 height를 이용하여 지도 크기를 조절할 수 있음 (e.g. width = '80%', height = '80%')

- folium.Choropleth(): 단계 구분도를 만드는 함수
  - geo_data: 지도 데이터
  - data: 색깔로 표현할 통계 데이터
  - columns: 행정 구역 코드 변수, 색깔로 표현할 변수
  - key_on: 지도 데이터의 행정 구역 코드
  - bins: 계급 구간(통계치가 가장 작은 값부터 큰 값까지 일정한 간격으로 계급 구간을 만들어 지역 색깔을 정함)
  - fill_color: 컬러맵 (e.g. fill_color = 'YlGnBu' 또는 fill_color = 'Blues)
    - 통계치가 클수록 파란색, 적을수록 노란색에 가깝게 표현
    - 통계치가 클수록 진한 파란색, 적을수록 연한 파란색
  - nan_fill_color: 결측치 색깔 (e.g. nan_fill_color = 'White')
  - fill_opacity: 투명도 (e.g. fill_opacity = 1) ~ 1은 불투명, 0은 투명
  - line_opacity: 경계선 투명도 (e. g. line_opacity = 1) ~ 1은 불투명, 0은 투명
  - line_weight: 선 두께(지역 구의 경계선을 추가할 때 사용, 이땐 fill_opacity = 0을 주자!, e.g line_weight = 4)
  - folium.Choropleth()에 .add_to(map_sig)를 추가하면 배경 지도에 단계 구분도를 덧씌움

- 계급 구간(bins) 정하기
  - 단계 구분도를 만들어서 배경 지도에 덧씌우기만 하면 지역이 색깔별로 표현되지 않음(계급 구간이 적당하지 않기 때문)
  - 따라서, 분위수(quantile)를 이용해 지역을 나누는 계급 구간을 정할 수 있음
  - e.g. bins = list(df_pop["pop"].quantile([0, 0.2, 0.4, 0.6, 0.8, 1]))
    - quantile(): 값을 크기순으로 나열한 다음 입력한 비율에 해당하는 값인 '분위수'를 구하는 함수
    - list()를 해주는 이유는 계급 구간을 folium.Choropleth()에 활용하려면 리스트로 되어 있어야 하기 때문

11_1. 시군구별 인구 단계 구분도 만들기

In [3]:
```
import json
import folium
import pandas as pd
```

In [4]:
```
# 1. 시군구 경계 지도 데이터 SIG.geojson 불러오기

geo = json.load(open('../data/SIG.geojson', encoding = 'UTF-8')) # UTF-8: 한국어가 지원되는 방식으로 열기

# SIG.geojson 파일: 대한민국의 시군구별 경계좌표가 들어 있는 GeoJson 파일, json 패키지의 json.load()를 이용하여 불러오
# json.load()로 파일을 읽으려면, 먼저 open()을 이용해 파일을 열어야 함(운영체제 과목에서 파일을 Open하고 난 후 Read 하는
# GeoJSON: 위치 정보를 JSON 포맷으로 저장한 표준 지리 데이터 포맷
```

In [37]:
```
# geo # Dictionary 타입
# geo의 properties에 들어 있는 SIG_CD에 지역을 나타내는 행정 구역 코드가 담겨 있고
# geo의 geometry에 시군구의 경계를 나타낸 위도, 경도 좌표가 들어 있음
```

In [6]:
```
# 행정 구역 코드 출력
geo['features'][0]['properties']
```

Out[6]:
```
{'SIG_CD': '42110', 'SIG_ENG_NM': 'Chuncheon-si', 'SIG_KOR_NM': '춘천시'}
```

In [35]:
```
# 위도, 경도 좌표 출력
# geo['features'][0]['geometry']
```

In [8]:
```
# 2. 지도에 표현할 시군구별 인구 통계 데이터 Population_Sig.csv 파일 불러오기

df_pop = pd.read_csv('../data/Population_Sig.csv')
df_pop.head()
```

Out[8]:

	code	region	pop
0	11	서울특별시	9509458
1	11110	종로구	144683
2	11140	중구	122499
3	11170	용산구	222953
4	11200	성동구	285990

In [9]:
```
df_pop.info()

# 행정 구역 코드를 나타내는 df_pop 데이터 프레임의 code는 int64 타입이라는 것을 확인할 수 있음
# 하지만, 지도를 만드는데 활용하고 싶으면 문자 타입으로 바꿔야 함
# 따라서 df.astype()을 이용하여 강제 캐스팅을 해줘야 함
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278 entries, 0 to 277
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   code    278 non-null    int64 
 1   region  278 non-null    object
 2   pop     278 non-null    int64 
dtypes: int64(2), object(1)
memory usage: 6.6+ KB
```

In [11]:
```
# 강제 형변환 ~ int64 → str

df_pop['code'] = df_pop['code'].astype(str) # 타입 캐스팅
```

In [12]:
```
df_pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278 entries, 0 to 277
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   code    278 non-null    object
 1   region  278 non-null    object
 2   pop     278 non-null    int64 
dtypes: int64(1), object(2)
memory usage: 6.6+ KB
```

단계 구분도 만들기

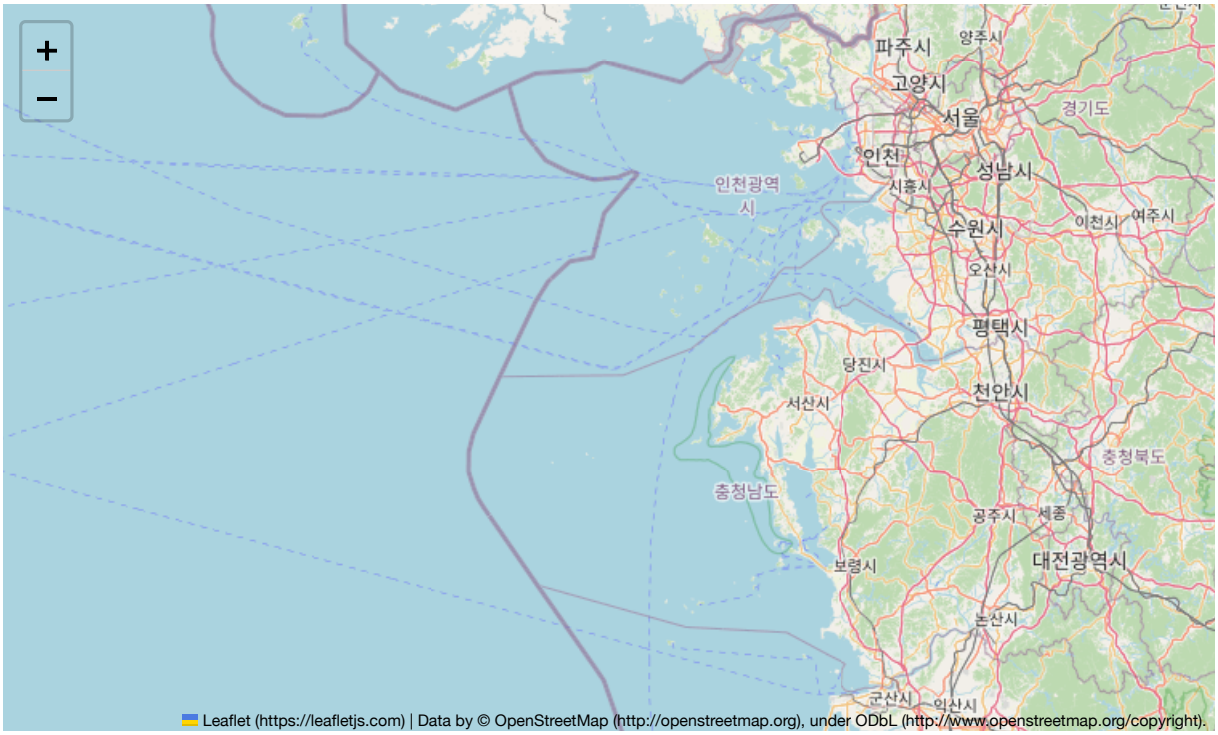
- folium.Map(): 배경 지도 만들기 (Map()의 M이 대문자라는 것을 주의해야 함)
- folium.Choropleth().add_to(): 단계 구분도를 만들고 배경 지도에 덧씌우기

In [13]:
```
# 1. 배경 지도 만들기

folium.Map(location = [35.95, 127.7],  # 지도 중심 좌표
           zoom_start = 8)             # 확대 단계

# width = '80%', height = '80%'와 같이 width와 height를 이용해 지도 크기를 조절할 수 있음
```

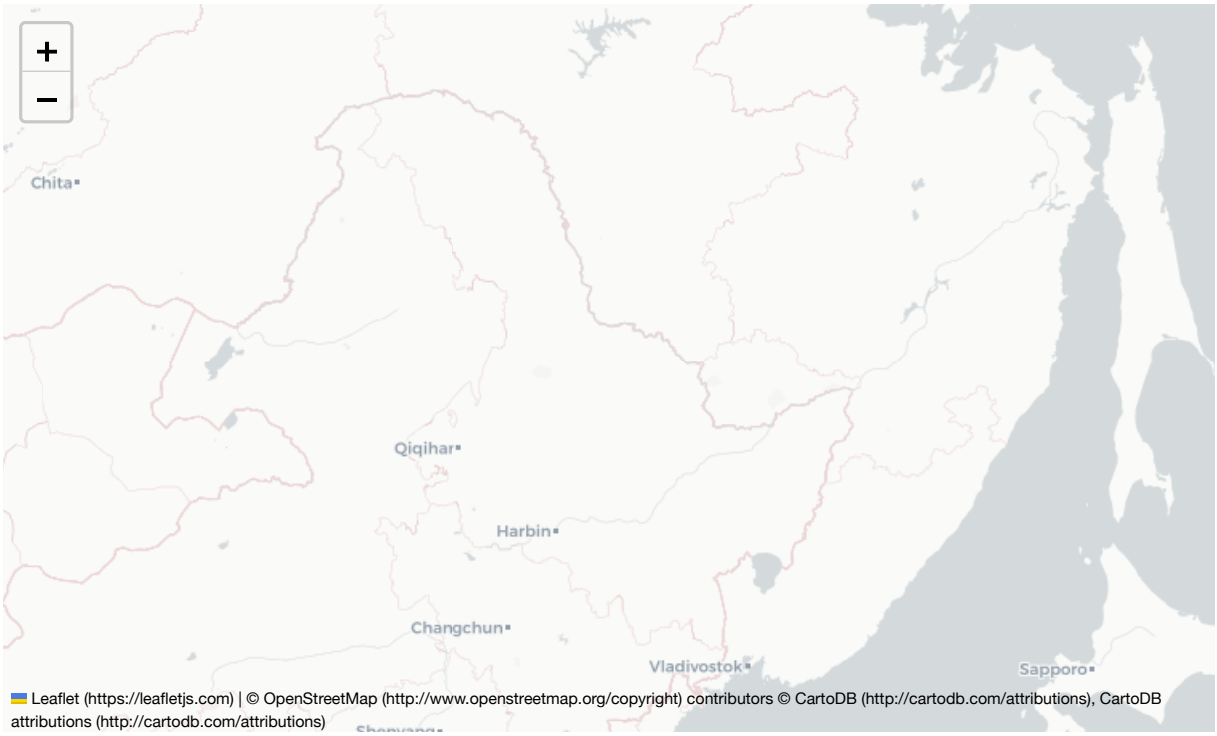
Out[13]:



In [14]:
```
map_sig = folium.Map(location = [35.95, 127.7],  # 지도 중심 좌표
                     zoom_start = 8,             # 확대 단계
                     tiles = 'cartodbpositron')  # 지도 종류 ~ 중요

# 단계 구분도가 잘 표현되도록 지도 종류를 cartodbpositron으로 변경
map_sig
```

Out[14]:



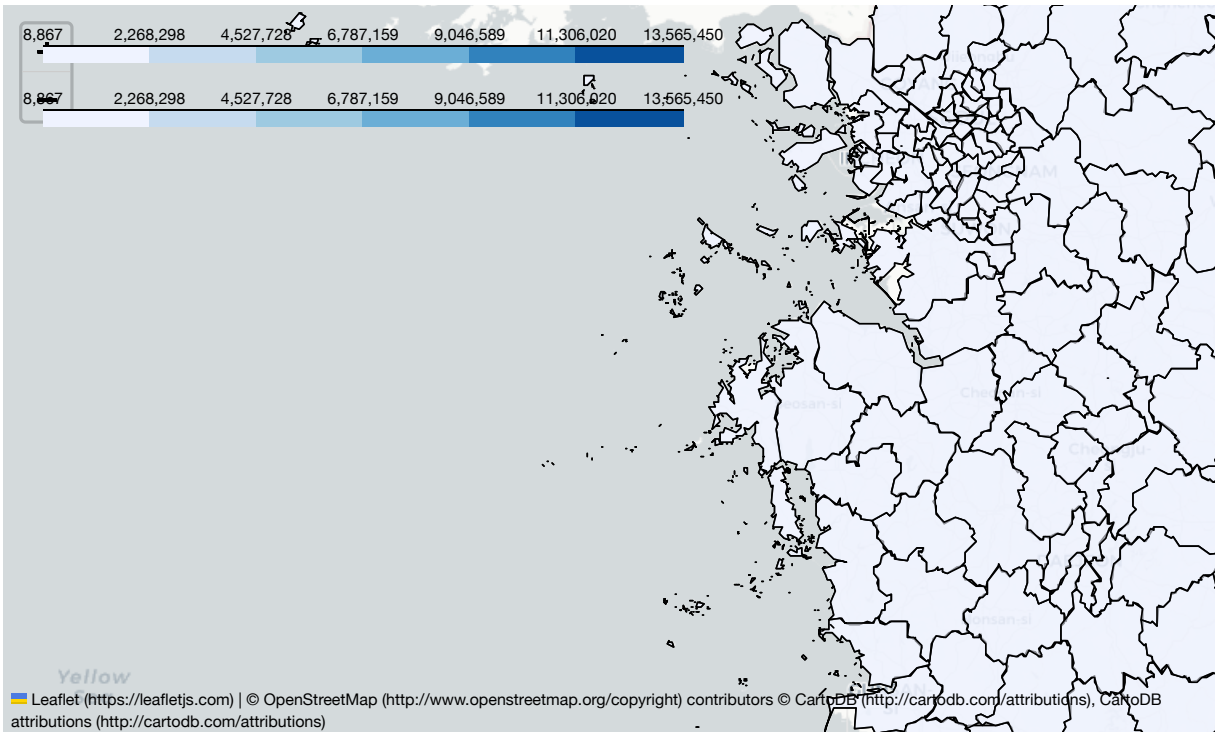
In [16]:
```
# 단계 구분도(Choropleth) 만들기
# geo_data: 지도 데이터
# data: 색깔로 표현할 통계 데이터
# columns: 통계 데이터의 행정 구역 코드 변수, 색깔로 표현할 변수
```

```
# key_on: 지도 데이터의 행정 구역 코드

folium.Choropleth(
    geo_data = geo,
    data = df_pop,
    columns = ('code', 'pop'),
    key_on = 'feature.properties.SIG_CD') \
        .add_to(map_sig) # 배경 지도에 단계 구분도를 덧씌운다는 뜻

map_sig
```

Out[16]:



In [17]:

```
# 3. 계급 구간 정하기
# 6개의 일정한 범위의 분위 수를 구하여 리스트형으로 바꾼 값을 계급 구간으로 사용
# 6개로 하는 이유는 folium의 단계 구분도는 최솟값부터 최댓값까지 6개의 일정한 간격으로 계급 구간을 만들어 색깔을 정하기 때문

bins = list(df_pop['pop'].quantile([0, 0.2, 0.4, 0.6, 0.8, 1]))
bins
```

Out[17]:

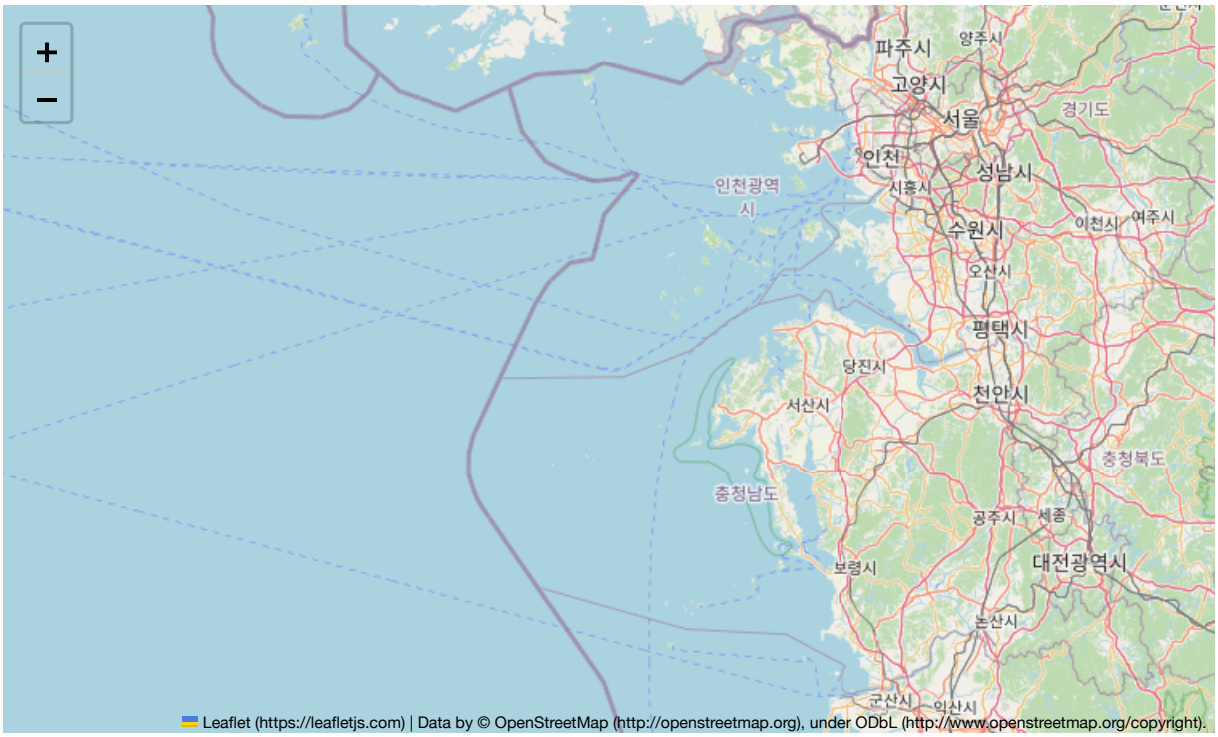
```
[8867.0, 50539.6, 142382.20000000004, 266978.6, 423107.20000000024, 13565450.0]
```

In [22]:

```
# 4. 디자인 수정하기

map_sig = folium.Map(location = [35.95, 127.7],
                     zoom_start = 8,
                     titles = 'cartodbpositron')
map_sig
```

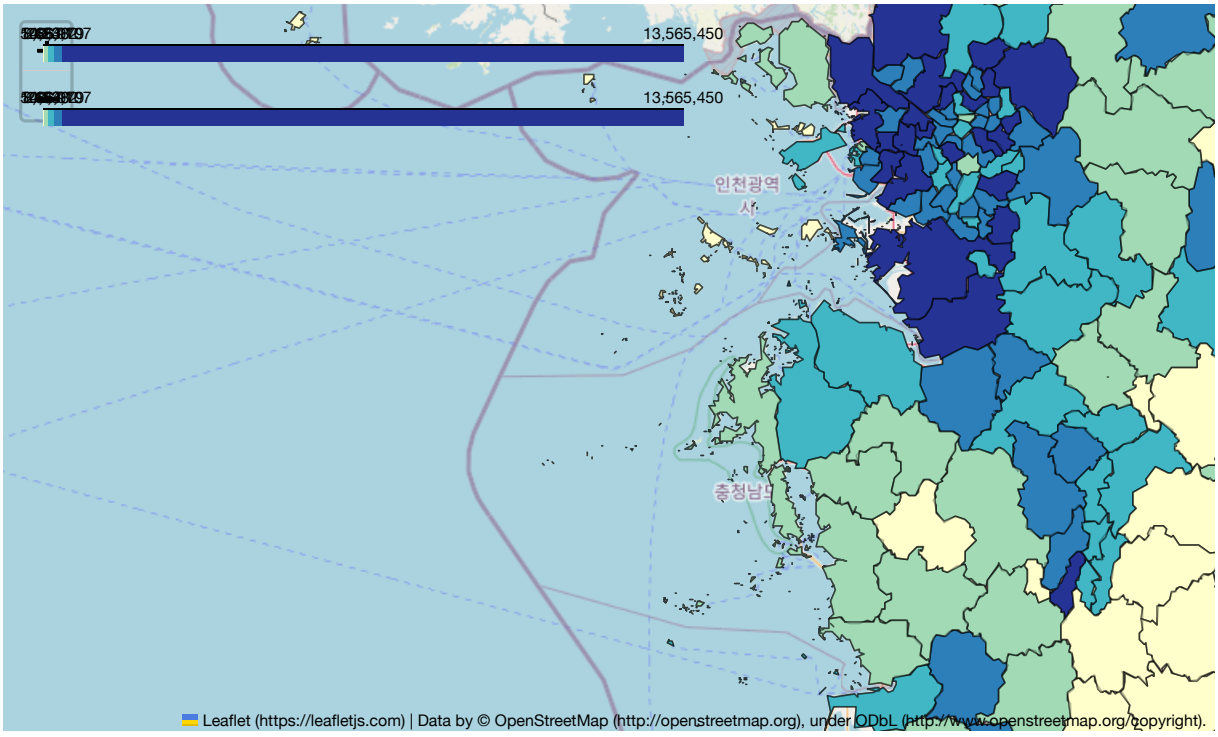
Out[22]:



In [24]:

```
folium.Choropleth(
    geo_data = geo, data = df_pop, columns = ('code', 'pop'),
    key_on = 'feature.properties.SIG_CD',
    fill_color = 'YlGnBu',   #Yellow, Green, Blue
    fill_opacity = 1,
    line_opacity = 0.5,
    bins = bins) \
        .add_to(map_sig)
map_sig
```

Out[24]:



11-2. 서울시 동별 외국인 인구 단계 구분도 만들기

In [30]:

```
# 1. 서울시의 동 경계 좌표가 들어 있는 파일 불러오기

geo_seoul = json.load(open('../data/EMD_Seoul.geojson', encoding='UTF-8'))
# geo_seoul
```

In [58]:

```
# 행정 구역 코드 출력
geo_seoul['features'][0]['properties']
```

Out[58]:
```
{'BASE_DATE': '20200630',
 'ADM_DR_CD': '1101053',
 'ADM_DR_NM': '사직동',
 'OBJECTID': '1'}
```

In [31]:
```
# 위도, 경도 좌표 출력
# geo_seoul['features'][0]['geometry']
```

In [64]:
```
# 2. 서울시 동별 외국인 인구 통계가 들어 있는 파일 불러오기
foreigner = pd.read_csv('../Data/Foreigner_EMD_Seoul.csv')
foreigner.head()
```

Out[64]:

	code	region	pop
0	1101053	사직동	418.0
1	1101054	삼청동	112.0
2	1101055	부암동	458.0
3	1101056	평창동	429.0
4	1101057	무악동	102.0

In [65]:
```
foreigner.info() # 행정 구역 코드인 code가 int형으로 되어 있음
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3490 entries, 0 to 3489
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   code    3490 non-null   int64
 1   region  3490 non-null   object
 2   pop     3486 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 81.9+ KB
```

In [67]:
```
foreigner['code'] = foreigner['code'].astype(str) # 지도에 활용할 수 있도록 행정 구역 코드를 str 타입으로 변환
```

In [69]:
```
# 계급 구간 만들기
bins = list(foreigner['pop'].quantile([0, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]))
bins
```

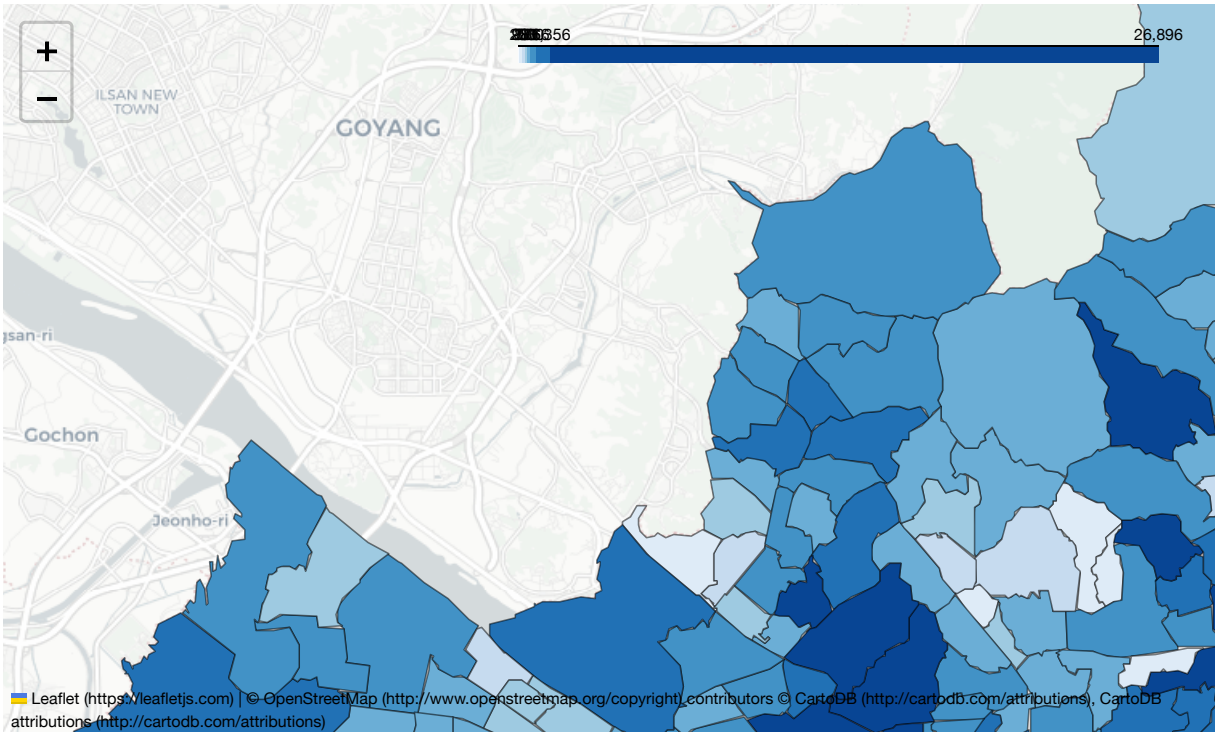
Out[69]:
```
[7.0, 98.0, 200.0, 280.0, 386.0, 529.5, 766.0, 1355.5, 26896.0]
```

In [71]:
```
# 3. 단계 구분도 만들기
map_seoul = folium.Map(location = [37.56, 127],
                       zoom_start = 12,
                       tiles = 'cartodbpositron')

folium.Choropleth(
    geo_data = geo_seoul,
    data = foreigner,
    columns = ('code', 'pop'),
    key_on = 'feature.properties.ADM_DR_CD',
    fill_color = 'Blues',
    nan_fill_color = 'White',
    fill_opacity = 1,
    line_opacity = 0.5,
    bins = bins) \
        .add_to(map_seoul)

map_seoul
```

Out[71]:



In [72]:

```
# 4_1. 구 경계선을 추가하기 위해, 구 경계 좌표를 담은 GEOJson 파일 불러오기
geo_seoul_sig = json.load(open('../data/SIG_Seoul.geojson', encoding='UTF-8'))
```

In [73]:

```
# 4_2. 서울 구 라인 추가
# 서울시 구 경계선을 이용해서 단계 구분도를 만든 후, .add_to()를 이용해 앞에서 만든 지도에 덮어쓰기
# 이때 색 투명도를 0을 입력하여 색깔을 칠하지 않도록 하고, line_weight = 4를 주어 구 경계선을 두껍게 나타내도록 함
folium.Choropleth(geo_data = geo_seoul_sig,
                  fill_opacity = 0,
                  line_weight = 4) \
     .add_to(map_seoul)
map_seoul
```

Out[73]: