Elixirr

파이썬으로 배우는 따릉이 데이터 분석과 시각화

9회차 수업

데이터 분석 & 시각화 및 고찰

수업 일정

전체 수업은 13회로 구성된다.

- 따릉이 이용현황 파악
- 문제 정의
- 파이썬 및 사용할 라이브러리 소개

- 비주얼 스튜디오 코드 설치
- 따릉이 데이터 수집

- 파이썬 라이브러리
- 따릉이 데이터프레임 만들기

- 따릉이 데이터프레임 관찰하기

- 시간 개념에 따른 데이터 분석을 위한 컬럼 추가

- 장소적 특징에 따른 데이터 분석을 위한 컬럼 추가

- 시간 개념에 따른 데이터 분석 및 시각화-(1)

- 시간 개념에 따른 데이터 분석 및 시각화-(2)

- 장소 특징에 따른 데이터 분석 및 시각화-(1)

- 장소 특징에 따른 데이터 분석 및 시각화-(2)

수업 일정

전체 수업은 13회로 구성된다.

- 시간 개념 X 장소 특징에 따른 데이터 분석 및 시각화

- 주말과 평일에 이용건수가 많은 대여소 데이터 분석 및 시각화

- 문제 정의에 맞춘 해결방안 도출
- 총정리

1. 문제정의
2. 데이터 수집
3. 데이터 가공

4. 데이터 분석

5. 시각화 및 탐색

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 현재 따릉이가 이용되고 있는 현황 조사
- 미래에 개선될 점에 대해서 분석주제 정의
- 파이썬과 판다스 소개

- 환경 설정 비주얼 스튜디오 코드(Visual Studio Code) 설치
- 따릉이 데이터 수집

- 확보한 따릉이 데이터 관찰 및 기초 데이터 분석
- 분석에 필요한 데이터 컬럼 추가 및 가공

- 시간 개념에 따른 데이터 분석
- **장속적 특징에 따른 데이터 분석**
- 두 개념을 동시에 고려한 데이터 분석
- 인기 대여점 Top50

- **4단계 데이터 분석을 가장 잘 반영할 수 있는 시각화**
- 문제 정의에 맞춘 해결방안 도출

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

단계 1 : 시간 개념에 따른 따릉이 이용패턴 분석 및 시각화

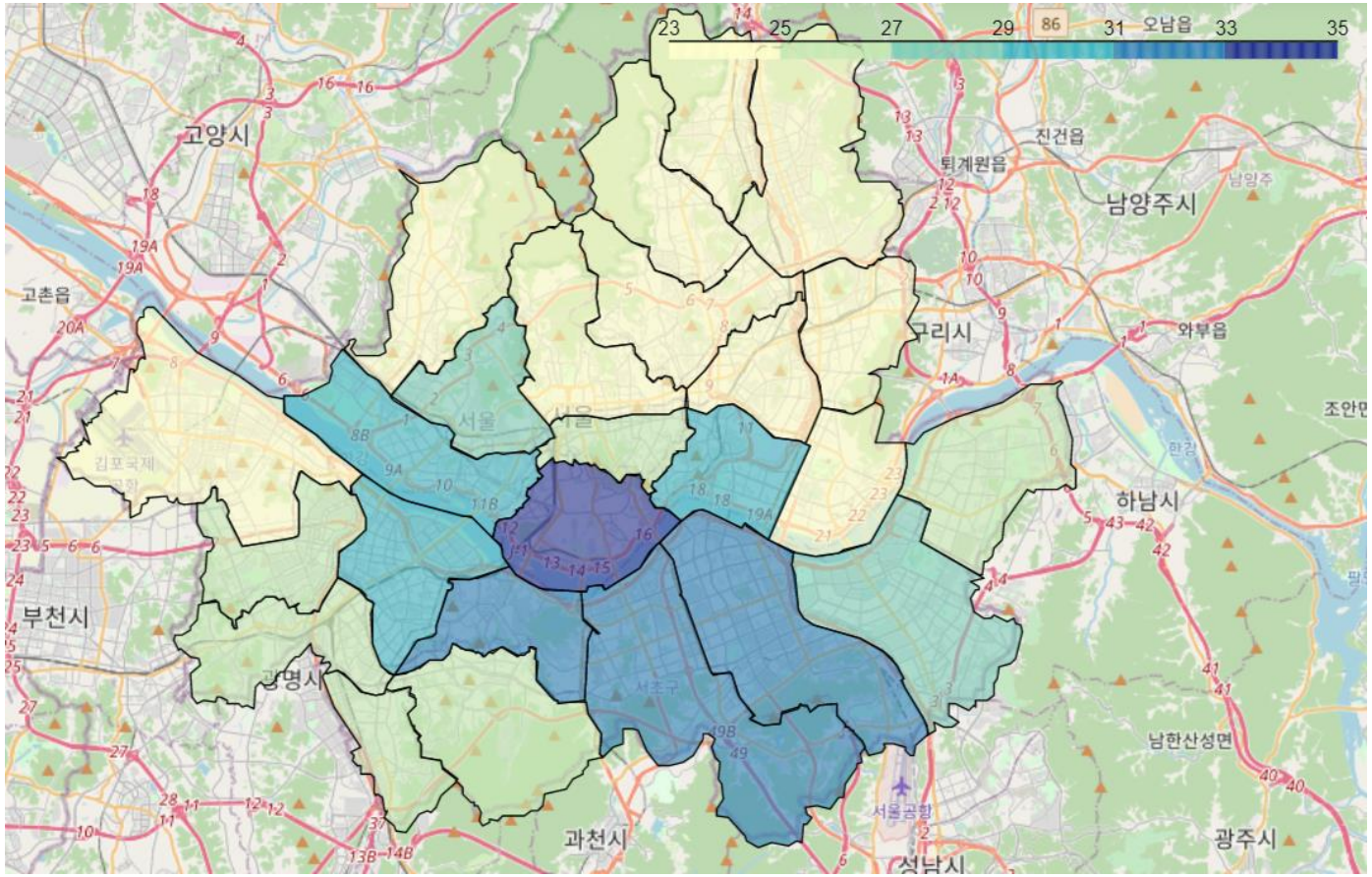
단계 2 : 장소적 특징에 따른 따릉이 이용패턴 분석 및 지도 시각화

단계 3 : 시간 개념 x 장소적 특징 연관 분석 후 시각화

단계 3 : 주말과 평일에 인기 있는 대여소 상위 50개 지도에 표시해보기

구별 따릉이 이용시간 평균 지도 표시

지역구별로 따릉이 이용시간 평균을 구해서 지도에 표시하고 싶다! 무엇부터 알아야 할까?



해리

각각의 지역구와 해당 구의 따릉이 이용시간 평균을 구해야 할 것 같아.

제니

서울시의 지도 데이터가 필요할 것 같아.

론

이렇게 멋진 지도를 그리는 방법을 배워야 할 것 같아.

각각의 지역구와 해당 구의 따릉이 이용시간 평균을 구해야 할 것 같아.

피봇테이블pivot_table

피봇테이블에서 인덱스만 필요한 경우로서 피봇테이블 수행 후 결과는 컬럼이 하나인 데이터프레임이다.

bikes.pivot_table(index='대여구', values='이용시간', aggfunc='mean')

1 3 4

1. pivot_table의 인덱스로 정할 컬럼명 : '대여구'
2. pivot_table의 컬럼으로 정할 컬럼명 : 생략가능
3. pivot_table의 값으로 정할 컬럼명 : '이용시간'
4. 집계함수 : mean()

```
구별이용시간평균 = bikes.pivot_table(\
        index = '대여구', \
        values = '이용시간', \
        aggfunc = 'mean')
구별이용시간평균
```

대여구	이용시간
강남구	31.79
강동구	25.39
강북구	23.57
강서구	23.14
관악구	26.51
광진구	24.95
구로구	26.48
금천구	26.40
노원구	24.26
도봉구	24.99
동대문구	23.78

pivot_table에서 columns를 생략했으므로
완성된 데이터프레임은 컬럼이 1개이다.

각각의 지역구와 해당 구의 따릉이 이용시간 평균을 구해야 할 것 같아.

인덱스 재설정 : reset_index()

데이터프레임의 인덱스를 컬럼으로 전송하며 새로운 정수 인덱스를 세팅한다.

```
# 대여구별 이용시간 평균을 구한다.

구별이용시간평균 = bikes.pivot_table(\
        index = '대여구', \
        values = '이용시간', \
        aggfunc = 'mean')
구별이용시간평균
```

✓ 0.2s

대여구	이용시간
강남구	31.79
강동구	25.39
강북구	23.57
강서구	23.14
관악구	26.51
광진구	24.95
구로구	26.48
금천구	26.40

인덱스 → 컬럼

```
구별이용시간평균.reset_index(inplace=True)
구별이용시간평균
```

✓ 0.3s

	대여구	이용시간
0	강남구	31.79
1	강동구	25.39
2	강북구	23.57
3	강서구	23.14
4	관악구	26.51
5	광진구	24.95
6	구로구	26.48
7	금천구	26.40
8	노원구	24.26
9	도봉구	24.99
10	동대문구	23.78
11	동작구	31.66

각각의 지역구와 해당 구의 따릉이 이용시간 평균을 구해야 할 것 같아.

데이터 정렬 : sort_values()

sort_values( ) 명령은 데이터프레임의 특정 컬럼의 데이터 값을 기준으로 크기 순서로 나열한다. 오름차순(ascending)을 기본으로 적용한다.

df.sort_values(by='정렬 기준 컬럼명', ascending=True, inplace=True)

1 2 3 4

1. **df.sort_values :** 데이터프레임의 특정 컬럼값에 따라 순서대로 다시 배열
2. **by='정렬 기준 컬럼명' :** 정렬 기준으로 삼을 컬럼명
3. **ascending=True :** 오름차순, **ascending=False :** 내림차순
4. **inplace=True :** 데이터프레임에 변경사항을 반영한다.

```
구별이용시간평균.sort_values(by='이용시간', \
                              ascending=True, \
                              inplace=True)
구별이용시간평균
```

✓ 0.4s

	대여구	이용시간
3	강서구	23.14
16	성북구	23.17
22	종로구	23.32
2	강북구	23.57
10	동대문구	23.78
24	중랑구	24.24
8	노원구	24.26
5	광진구	24.95
9	도봉구	24.99
21	은평구	25.16

서울시의 지도 데이터가 필요할 것 같아.

JSON 파일

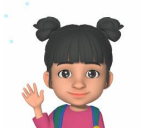
정의 : 파이썬의 딕셔너리 스타일로 데이터를 표현한 파일이다. 키-값으로 이루어져 있으며 사람이 쉽게 읽을 수 있도록 구성되어 있다. 시스템과 언어가 달라도 데이터를 교환하는데 편리하다.
실시간 버스 위치정보 같은 데이터를 간단하게 JSON으로 주고받는다.

만들기 : 객체 -> 모든 객체는 { }로 표현 데이터 -> { "키" : 값 }
예) { "name" : "홍길동" }, { "name" : "홍길동" , "age" : 30 , "weight" : 68.2 }

인코딩 : JSON의 내용에 한글이 들어갈 때는 파일을 UTF-8 인코딩으로 저장해야 한다.


```
{'type': 'FeatureCollection',
 'features': [{'type': 'Feature',
   'properties': {'code': '11250',
    'name': '강동구',
    'name_eng': 'Gangdong-gu',
    'base_year': '2013'},
   'geometry': {'type': 'Polygon',
    'coordinates': [[[127.11519584981606, 37.557533180704915],
      [127.11540412678968, 37.557491025257455],
      [127.1165206546129, 37.557268061772696],
      [127.1175954493625, 37.55705301284316],
      [127.11879551821994, 37.557222485451305],
      [127.11969651045837, 37.558176474822524],
```


type : FeatureCollection
features : type : Feature,
properties : code : 11250
name : 강동구
name_eng : Gangdong-gu
base_year : 2013
geometry : type : Polygon
coordinates : [127.11540412678968, 37.557491025257455], [127.1165206546129, 37.557268061772696], [127.1175954493625, 37.55705301284316], [127.11879551821994, 37.557222485451305], [127.11969651045837, 37.558176474822524],



서울시의 지도 데이터가 필요할 것 같아.

컬럼별 기초 통계

데이터 프레임의 각각의 컬럼의 최대, 최소, 평균, 중앙값 등을 집계함수를 사용해서 구할 수 있다..

최소값 min

```
bikes['대여점위도'].min()
✓ 0.3s
37.437271
```

- bikes['대여점위도'] 컬럼값들 중 최소값

최대값 max

```
bikes['대여점위도'].max()
✓ 0.5s
37.68972
```

- bikes['대여점위도'] 컬럼값들 중 최대값

평균 mean

```
bikes['대여점위도'].mean()
✓ 0.5s
37.547349884398024
```

- bikes['대여점위도'] 컬럼값들의 평균값

중앙값 median

```
bikes['대여점위도'].median()
✓ 0.4s
37.54459
```

- bikes['대여점위도'] 컬럼값들 중 중위값

이렇게 멋진 지도를 그리는 방법을 배워야 할 것 같아.

folium.Map()

map = folium.Map(location=지도중심위치, zoom_start=11)

1 **2** **3**

1. **folium.Map** : 중심위치와 배율을 조정해서 지도를 초기화해서 보여준다.
2. **location=지도중심위치 :** 중심위치의 위도와 경도를 입력한다.
3. **zoom_start=11 :** 배율을 입력한다. 값이 크면 더 좁은 지역을 크게 볼 수 있다.

```
lat = bikes['대여점위도'].mean()
lon = bikes['대여점경도'].mean()
center = [lat, lon]
map = folium.Map(location = center, zoom_start = 11)
map
```

center = [lat, lon] ← 리스트

평균위도 평균경도

이렇게 멋진 지도를 그리는 방법을 배워야 할 것 같아.

map.choropleth() - (1)

choropleth() 명령어는 판다스의 데이터프레임에 있는 지역 데이터와 그에 해당하는 수치 데이터분포를 지도에 색깔로 표시해 준다.

```
1 import  json

2 with open('./data/seoul.json', 'r', encoding='utf-8') as f:
      서울시지도 = json.load(f)
          4
3 folium.choropleth (
        5 geo_data = 서울시지도,
        6 data = 구별이용시간평균 ,
        7 columns = [ '대여구', '이용시간' ] ,
        8 fill_color = 'YlGnBu' ,
        9 key_on = 'feature.properties.name' ).add_to(map)

map
```


{'type': 'FeatureCollection',
 'features': [{'type': 'Feature',
   'properties': {'code': '11250',
    'name': '강동구',
    'name_eng': 'Gangdong-gu',
    'base_year': '2013'},
   'geometry': {'type': 'Polygon',
    'coordinates': [[[127.11519584981606, 37.557533180704915],
      [127.11540412678968, 37.55749102525745 5],
      [127.1165206546129, 37.55726806177269 6],
      [127.1175954493625, 37.55705301284316],
      [127.11879551821994, 37.55722248545130 5],
      [127.11969651045837, 37.55817647482252 4],

1. json 파일을 다루는 라이브러리를 임포트
2. 서울시 지도데이터를 변수 할당
3. folium 라이브러리에 속한다
4. map에 지역구와 해당 수치 데이터를 표시
5. 지역구를 표시할 지리 데이터
6. 지도에 표시될 데이터프레임
7. 데이터프레임에서 지역구와 수치데이터 컬럼
8. 표현할 색깔 : Yellow - Green - Blue
9. json파일에서 키값

이렇게 멋진 지도를 그리는 방법을 배워야 할 것 같아.

map.choropleth() - (2)

choropleth() 명령어는 판다스의 데이터프레임에 있는 지역 데이터와 그에 해당하는 수치 데이터분포를 지도에 색깔로 표시해 준다.

```
import  json

with open('./data/seoul.json', 'r', encoding='utf-8') as f:
    서울시지도 = json.load(f)

folium.choropleth (
            geo_data = 서울시지도,
            data = 구별이용시간평균 ,
            columns = [ '대여구', '이용시간' ] ,
            fill_color = 'YlGnBu' ,
            key_on = 'feature.properties.name' ).add_to(map)

map
```

```
구별이용시간평균.sort_values(by='이용시간', \
                              ascending=True, \
                              inplace=True)
구별이용시간평균
```

✓ 0.4s

	대여구	이용시간
3	강서구	23.14
16	성북구	23.17
22	종로구	23.32
2	강북구	23.57
10	동대문구	23.78
24	중랑구	24.24
8	노원구	24.26
5	광진구	24.95
9	도봉구	24.99
21	은평구	25.16


```
{'type': 'FeatureCollection',
'features': [{'type': 'Feature',
  'properties': {'code': '11250',
   'name': '강동구',
   'name_eng': 'Gangdong-gu',
   'base_year': '2013'},
  'geometry': {'type': 'Polygon',
   'coordinates': [[[127.11519584981606, 37.557533180704915],
     [127.11540412678968, 37.557491025257455],
     [127.1165206546129, 37.557268061772696],
     [127.1175954493625, 37.55705301284316],
     [127.11879551821994, 37.557222485451305],
     [127.11969651045837, 37.558176474822524],
```

이렇게 멋진 지도를 그리는 방법을 배워야 할 것 같아.

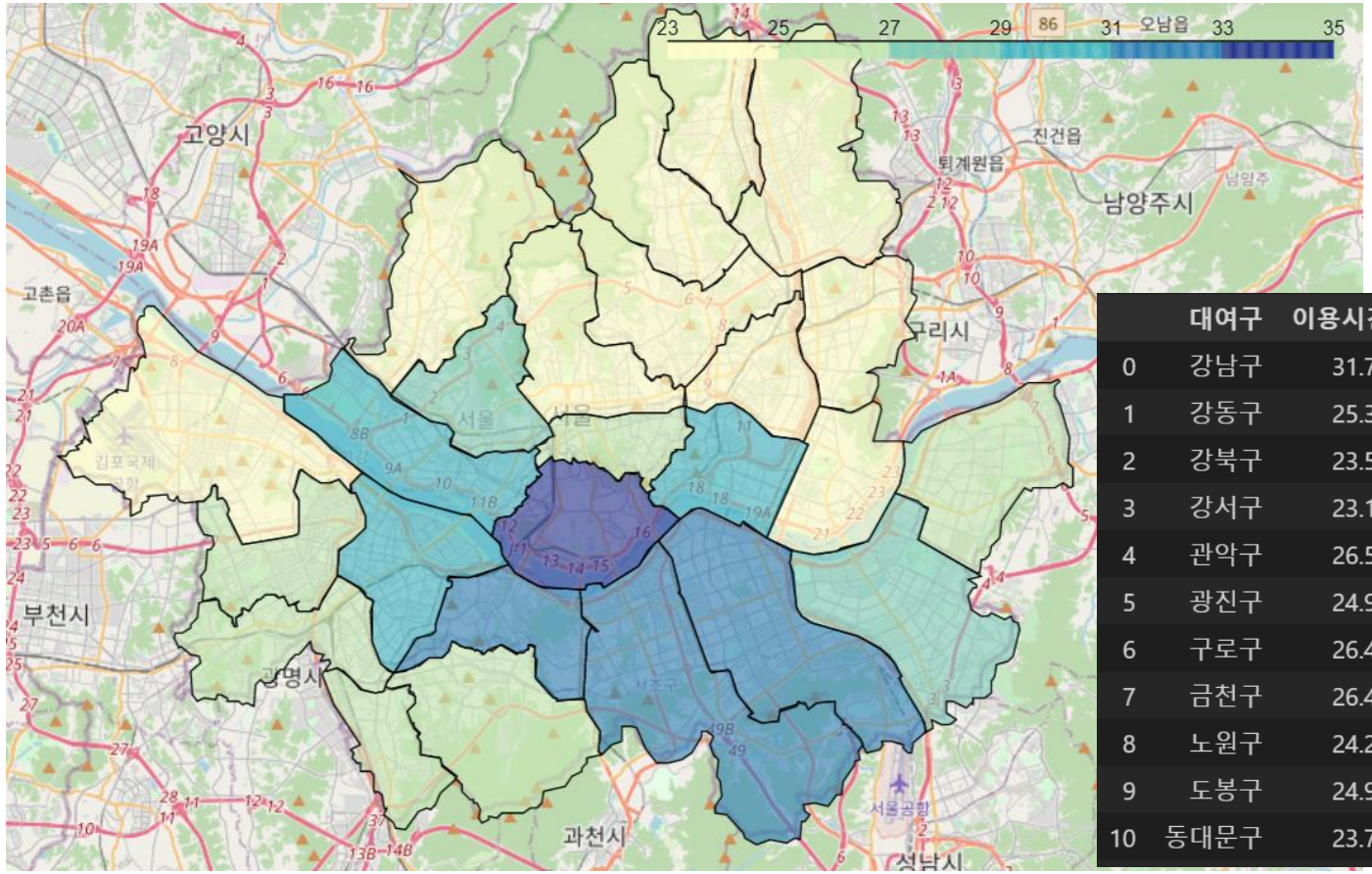
map.choropleth() - (3)

choropleth() 명령어는 판다스의 데이터프레임에 있는 지리 데이터와 그에 해당하는 수치 데이터분포를 지도에 색깔로 표시해 준다.

```
import json

with open('./data/seoul.json', 'r', encoding='utf-8') as f:
    서울시지도 = json.load(f)

서울시지도
```

```
# 지도의 중심위치를 정한다.
lat = bikes['대여점위도'].mean()
lon = bikes['대여점경도'].mean()
center = [lat, lon]
map1 = folium.Map(location = center, zoom_start = 11)
map1
```

```
folium.Choropleth(
    geo_data = 서울시지도,
    data = 구별이용시간평균,
    columns = ['대여구', '이용시간'],
    fill_color = 'YlGnBu',
    key_on = 'feature.properties.name').add_to(map1)

map1
```



	대여구	이용시간
0	강남구	31.79
1	강동구	25.39
2	강북구	23.57
3	강서구	23.14
4	관악구	26.51
5	광진구	24.95
6	구로구	26.48
7	금천구	26.40
8	노원구	24.26
9	도봉구	24.99
10	동대문구	23.78

나 지금 어느 단계를 공부하는 거지?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

단계 3 : 분석 명령어를 사용해서 장소적 특징에 따른 따릉이 이용패턴 분석 및 지도 시각화

데이터 집계 -> df.pivot_table()

인덱스 재설정 -> df.reset_index()

지도 초기화 -> folium.Map()

구단위 지도 시각화 -> map.choropleth()

퀴즈

퀴즈를
풀어봅시다

1. 중심위치와 배율을 조정해서 지도를 초기화해서 보여주는 명령어는 ?

2. 판다스의 데이터프레임에 있는 지역 데이터와 그에 해당하는 수치 데이터분포를 지도에 색깔로 표시해 주는 명령어는 ?

3. 파이썬의 딕셔너리 스타일로 데이터를 표현한 파일로 키-값으로 이루어져 있으며 사람이 쉽게 읽을 수 있도록 구성되어 있는 파일은 ?

4. 데이터프레임의 특정 컬럼의 데이터 값을 기준으로 크기 순서로 나열하고 오름차순 (ascending)을 기본으로 적용하는 명령어는 ?

실습 순서

장소 특징에 따른 데이터 분석

대여구 별 이용건수 분석

대여구 별 이용시간 분석

실습 시작

GD쌤

이제부터 Visual Studio Code 실습 환경에서 지금까지 배운 내용을 실습해 보겠습니다.

앞에서 배웠던 내용을 Visual Studio Code에서 직접 실습해보면 더욱 이해하기 편리할 것입니다.

수업 마무리

GD쌤

지금까지 9회차 수업내용을 배워 보았습니다.

다음 시간에는 10회차 수업내용으로 장소적 특징에 따른 데이터 분석 및 시각화로서 실습위주로 진행해 보겠습니다.

수고 많으셨어요. 다음 시간에 만나요.