

파이썬으로 시작하는 데이터 분석

03 데이터 시각화



목차

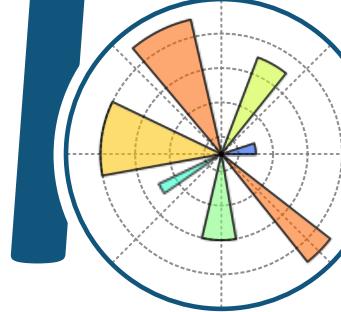
-
- 01 Matplotlib 기본 문법
 - 02 Seaborn 기본 문법
 - 03 선 그래프와 막대 그래프
 - 04 파이 차트와 히스토그램

01

Matplotlib 기본 문법

✓ Matplotlib

matplotlib



- Python의 시각화 패키지
- Seaborn, Pandas와 같은 패키지와 함께 사용하면 풍부한 시각화 기능들을 사용할 수 있음

① 패키지 불러오기

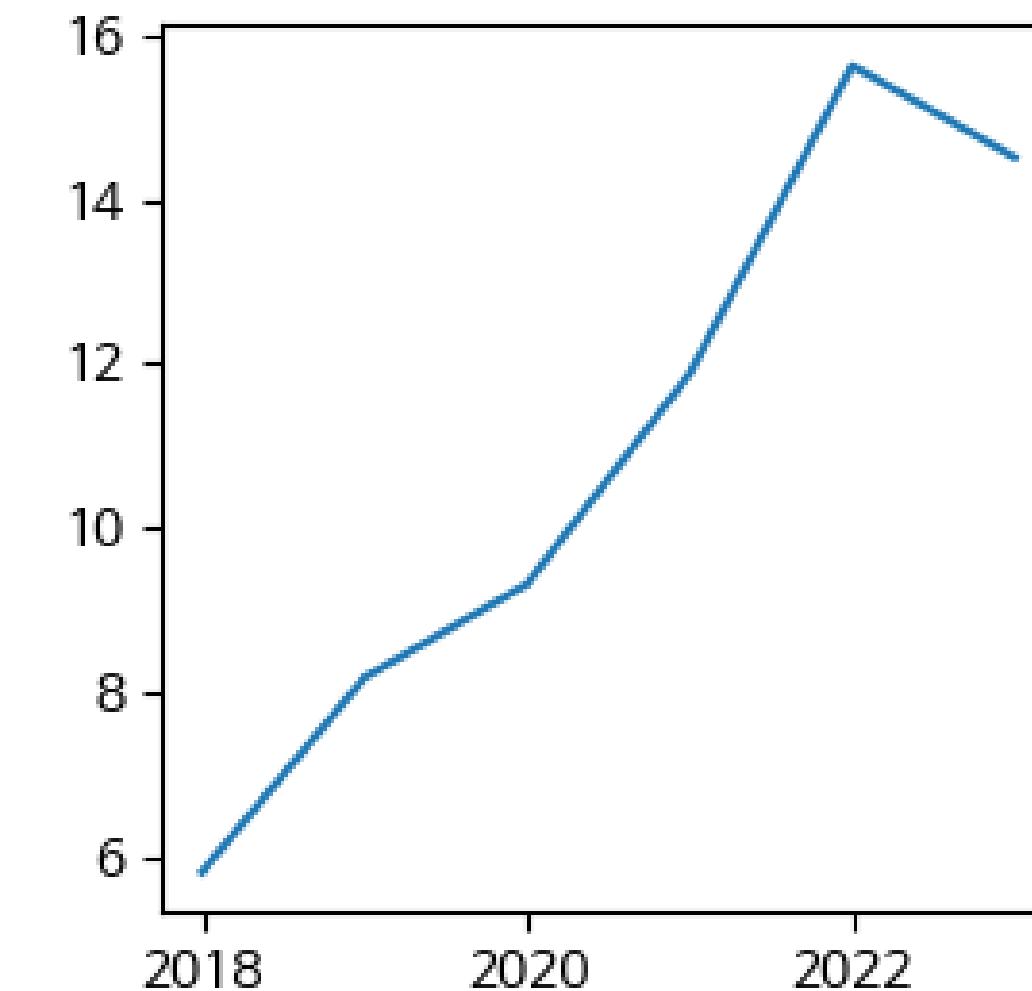
```
import matplotlib.pyplot as plt
```

- 다양한 시각화 그래프를 그리는데 도움을 주는 패키지
- 관용적으로 plt라는 이름으로 import

⑤ 선 그래프 그리기

```
plt.plot(xdata, ydata) # 간단한 선 그래프를 그림  
plt.show() # 그린 그래프를 화면에 표시
```

- X축의 데이터, Y축의 데이터를 순서대로 전달
- plt.show() 함수를 통해 그린 그래프를 화면에 출력



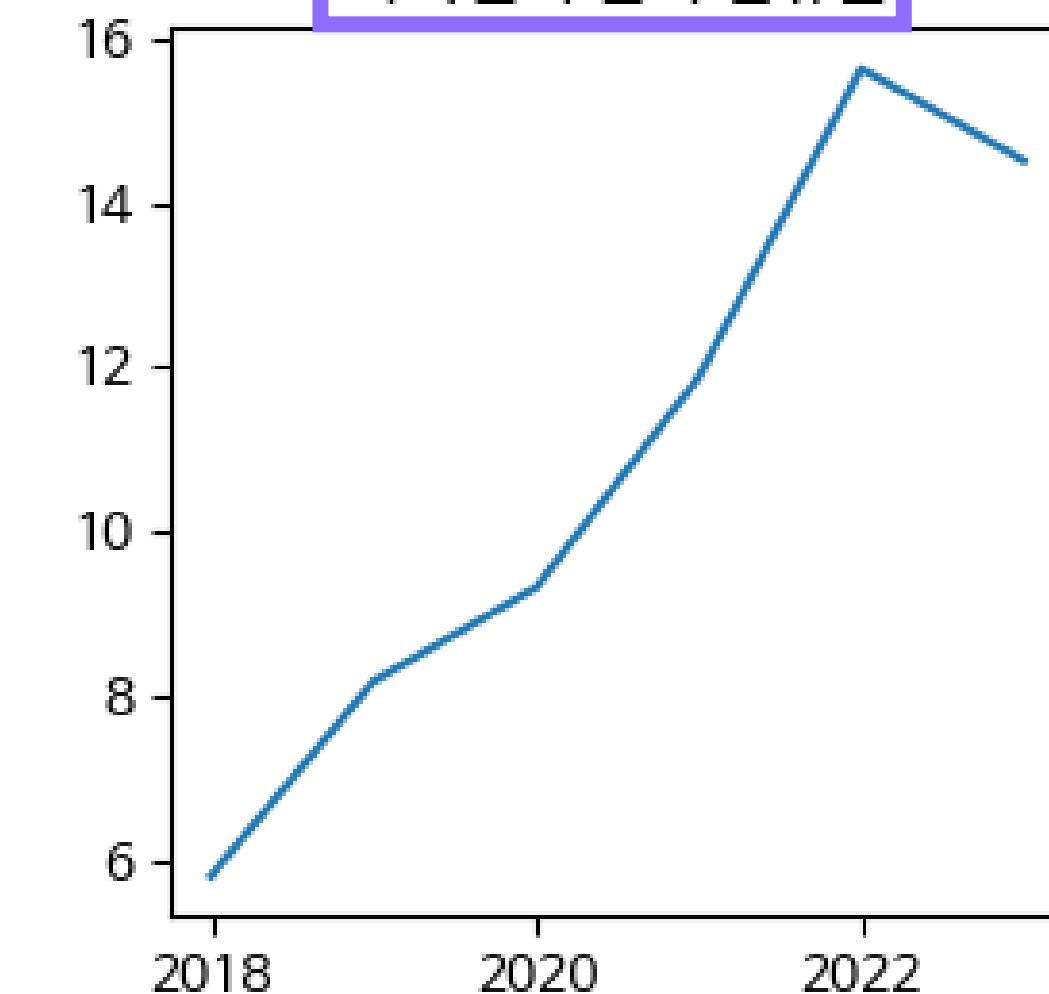
⑤ 그래프의 제목 설정

```
plt.title("파이썬의 언어 점유율") # 그래프의 제목을 설정
```

- plt.title()함수에 제목을 문자열로 전달하여 설정

그래프의 제목

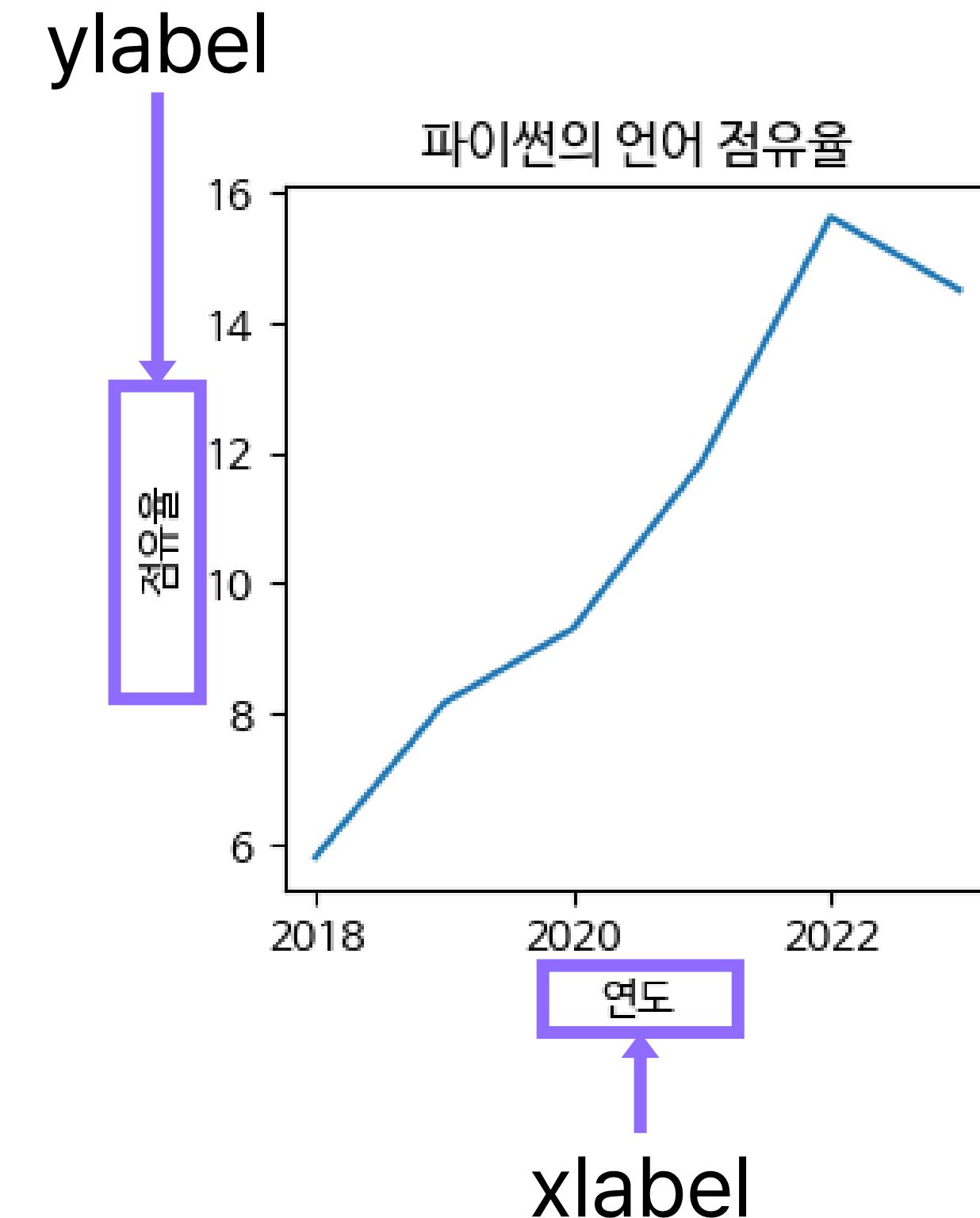
파이썬의 언어 점유율



✓ 축의 레이블 표시하기

```
plt.xlabel("연도") # x축의 레이블을 "연도"로 설정  
plt.ylabel("점유율") # y축의 레이블을 "점유율"로 설정
```

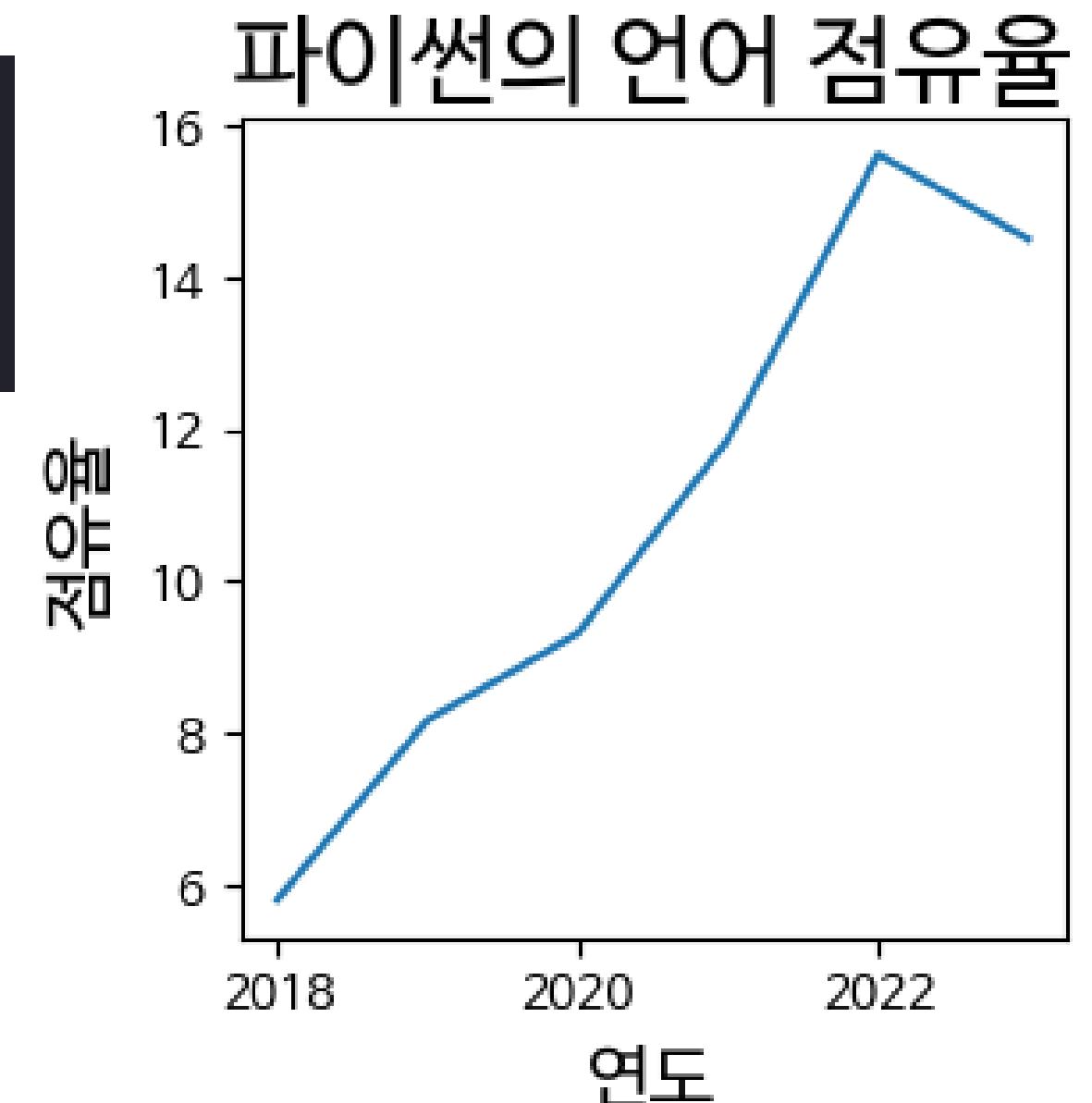
- plt.xlabel(): 문자열을 전달하여 x축 레이블 설정
- plt.ylabel(): 문자열을 전달하여 y축 레이블 설정



✓ 텍스트의 크기 변경하기

```
plt.title("파이썬의 언어 점유율", fontsize=20) # 제목의 크기를 20으로 설정  
plt.xlabel("연도", fontsize=15) # 레이블의 크기를 15로 설정  
plt.ylabel("점유율", fontsize=15) # 레이블의 크기를 15로 설정
```

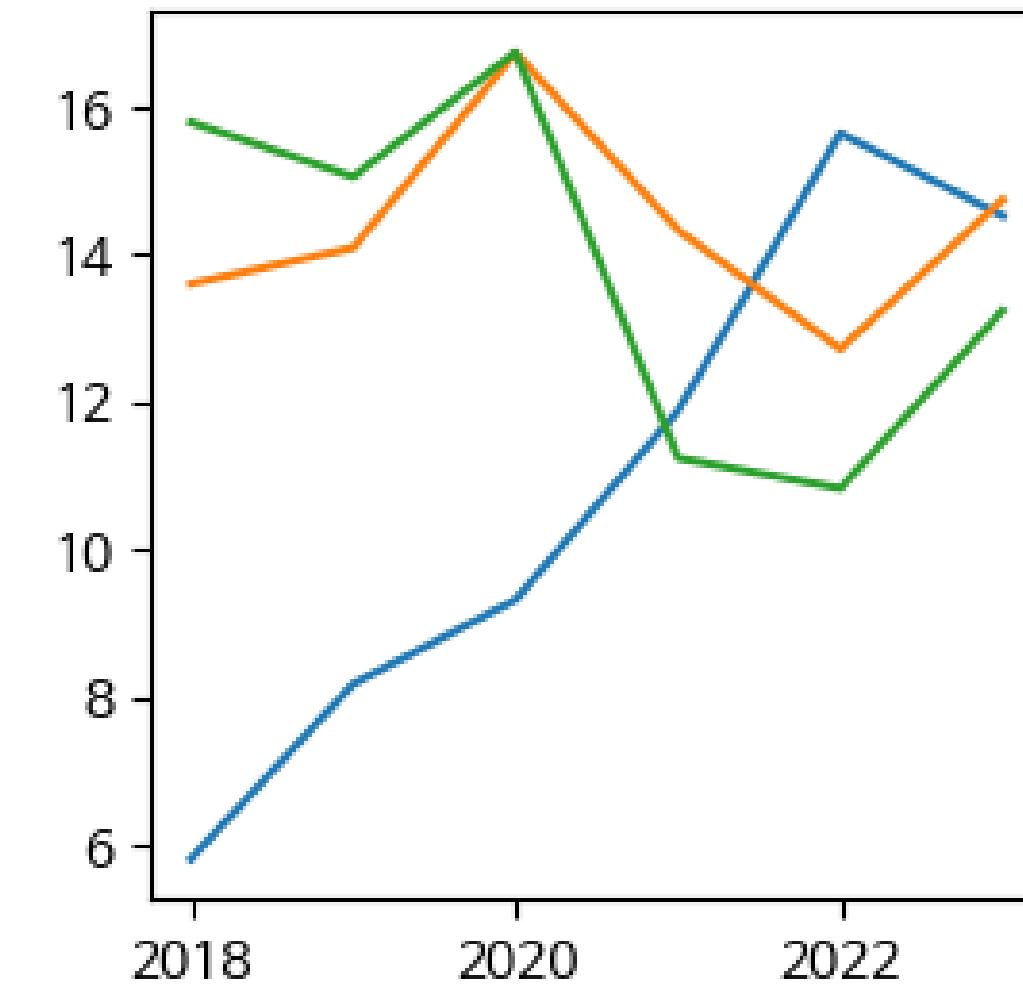
- `fontsize`: 글자의 크기를 변경하는 매개변수
- `title`, `xlabel`, `ylabel` 등 텍스트를 설정하는 함수에 `fontsize` 매개변수에 크기를 전달



✓ 여러 그래프를 함께 표시하기

```
plt.plot(year, python)  
plt.plot(year, C)  
plt.plot(year, java)
```

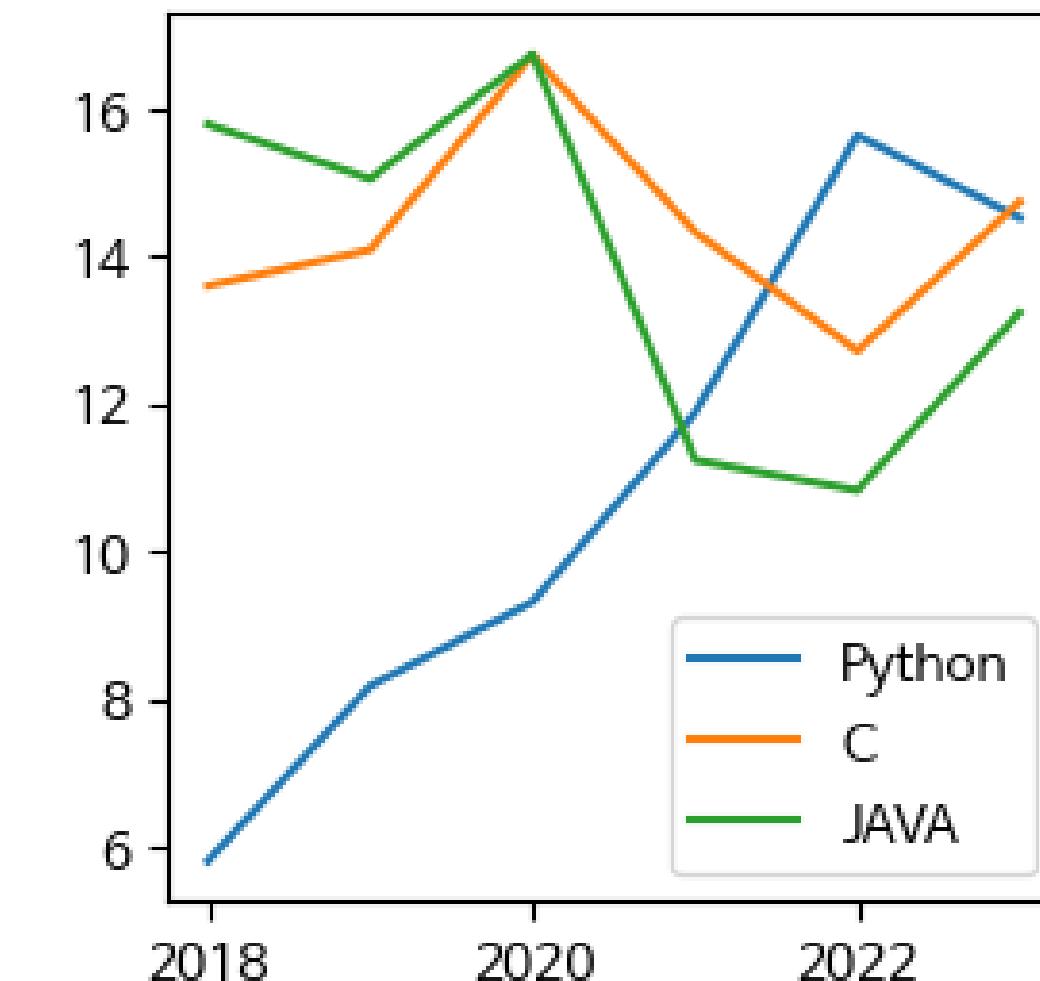
- 한번 호출할 때마다 그래프를 하나씩 추가



⑤ 그래프의 범례 표시하기

```
plt.plot(year, python)
plt.plot(year, C)
plt.plot(year, java)
plt.legend(['Python', 'C', 'JAVA']) # 그래프를 그린 순서대로 이름을 전달
```

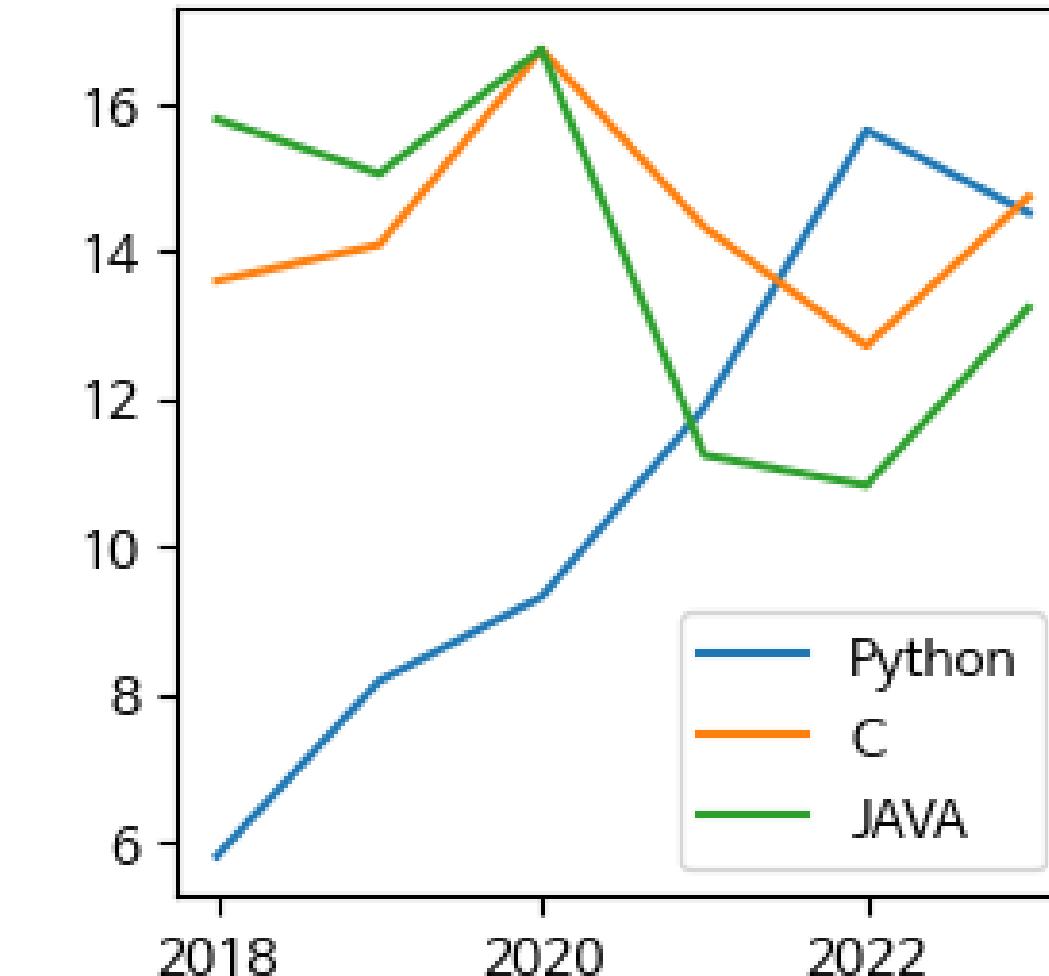
- plt.legend()함수에 각 그래프의 이름을 순서대로 전달
- 그래프의 순서가 달라지면 이름의 순서도 변경



✓ 그래프의 범례 표시하기

```
plt.plot(year, python, label="Python")
plt.plot(year, C, label="C")
plt.plot(year, java, label="JAVA")
plt.legend() # 각 그래프의 label을 이미 전달했으니 바로 호출
```

- 각 그래프를 그릴 때 label에 그래프 이름을 전달
- 각 그래프의 레이블이 있으니 legend에 전달은 하지 않음



✓ 지금까지의 코드

- plt에서 직접 함수들을 호출
- 서로 다른 그래프를 한번에 그리는 것은 불가능

```
plt.title("파이썬의 언어 점유율")
plt.xlabel("연도")
plt.ylabel("점유율")
plt.plot(year, python, label="Python")
plt.plot(year, c, label="C")
```

✓ plt의 함수를 이용해 그래프 그리기

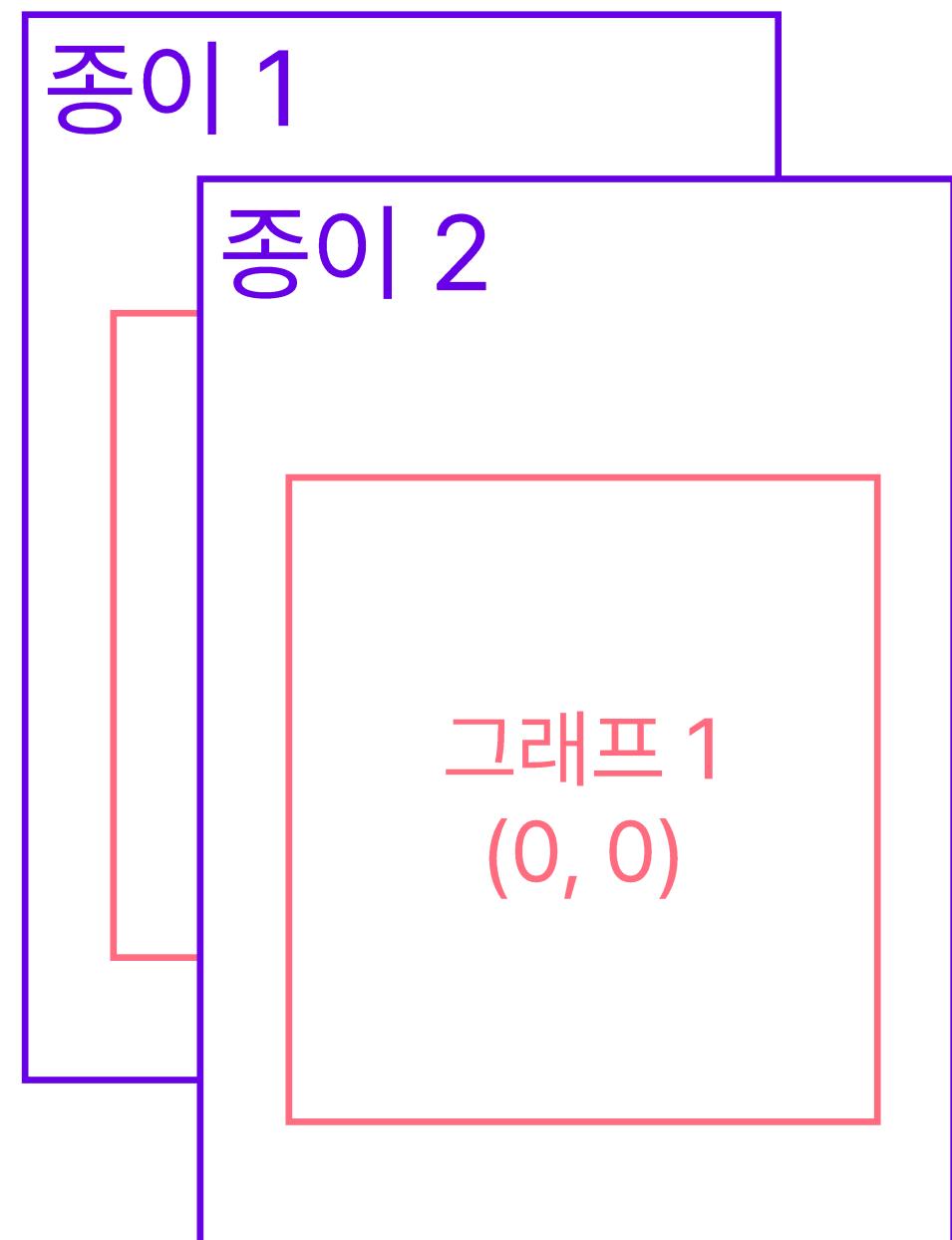
- 종이1의 그래프1에 계속해서 그래프를 그리는 방법
- 표현할 그래프가 하나라면 좋은 방법
- 그래프를 여러 개 그려야 하는 경우엔 다른 방법이 필요

종이 1

그래프 1
(0, 0)

✓ 종이를 여러 개 쓰기

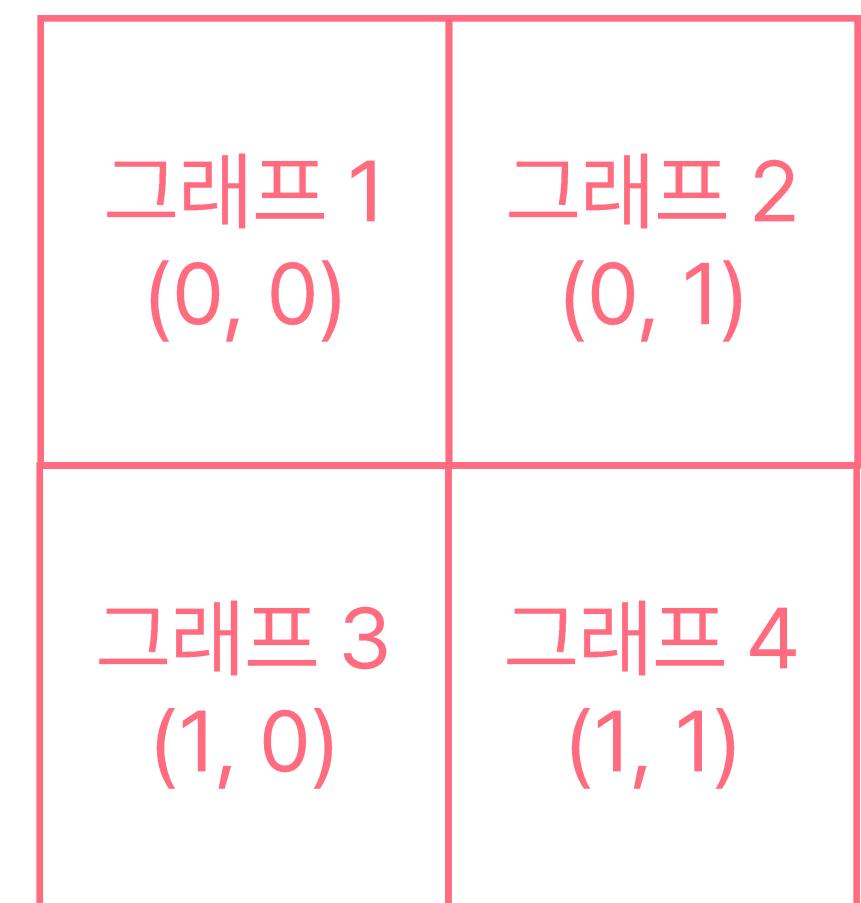
- 각 종이에 그래프를 각각 그리는 방법
- 각 그래프가 서로 연관이 전혀 없는 경우엔 이 방법을 사용
- 같은 주제의 그래프를 한번에 출력하기는 어려움



✓ 종이에 여러 그래프를 그리기

- 종이 하나를 나누어 여러 개의 그래프를 표현
- 같은 주제거나 연관이 있는 그래프일 경우 좋은 방법

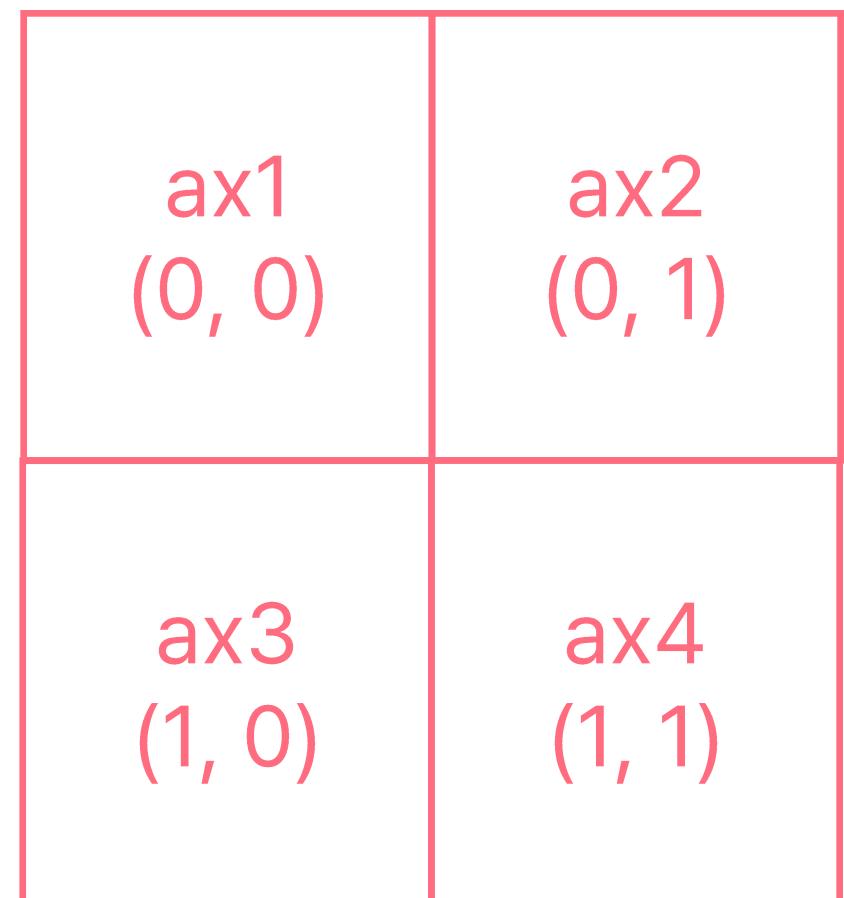
종이 1



✓ figure와 subplot

- figure:
 - 여러 그래프가 담길 수 있는 종이에 해당
- ax1~ax4
 - 실제 그래프가 그려지는 캔버스에 해당
 - 이들을 서브플롯(subplot)이라고 부름

figure 1



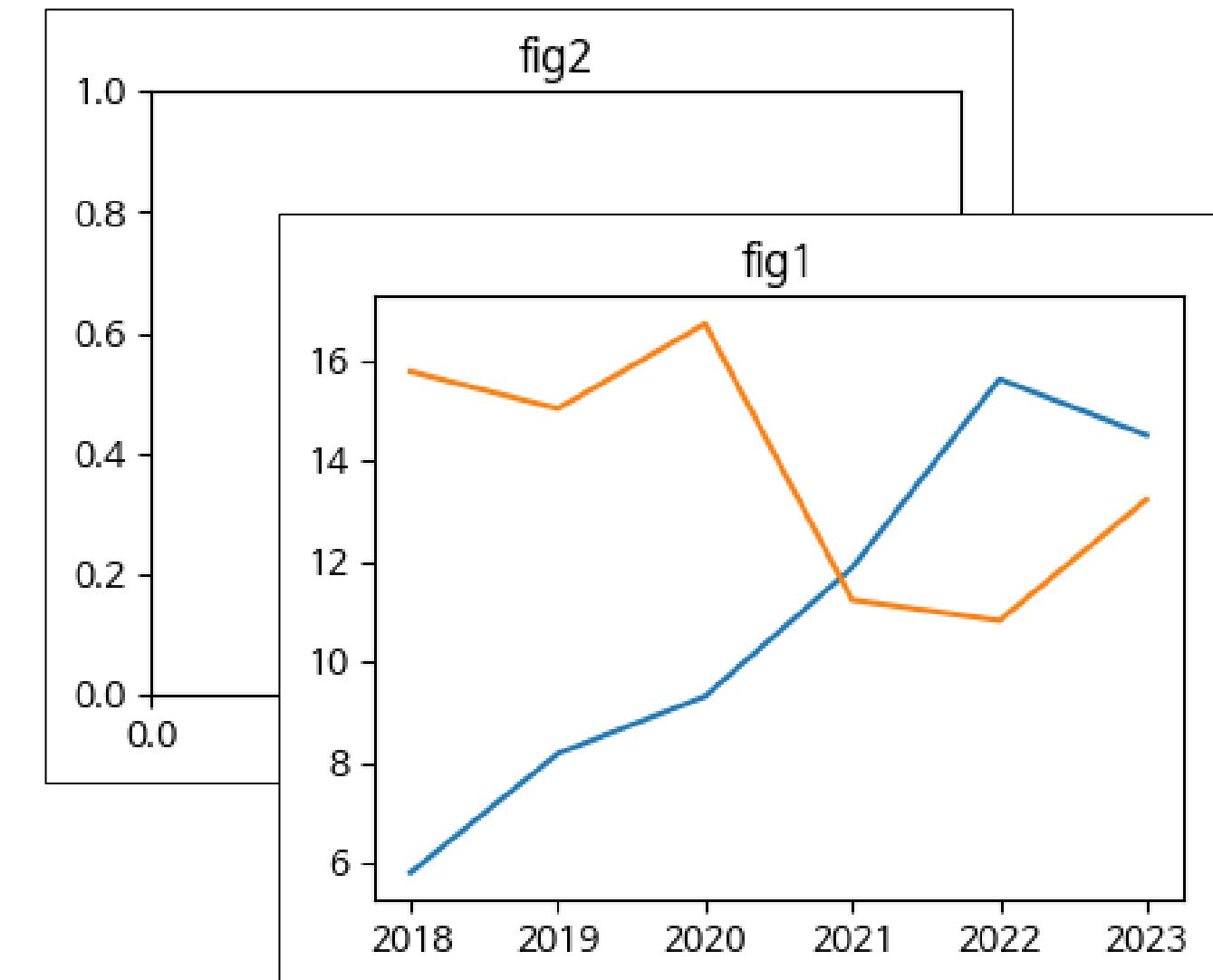
✓ 여러 개의 figure을 사용

```
fig1 = plt.figure(1, figsize=(4,3)) # ID가 1인 새로운 figure 객체를 생성하고 선택
plt.plot(year, python, label="Python") # 현재 선택된 figure에 선 그래프를 그림
plt.title("fig1") # 현재 선택된 figure의 제목을 설정

fig2 = plt.figure(2, figsize=(4,3)) # ID가 2인 새로운 figure 객체를 생성하고 선택
plt.plot(year, java, label="JAVA") # 현재 선택된 figure에 선 그래프를 그림
plt.title("fig2") # 현재 선택된 figure의 제목을 설정

plt.show() # 모든 figure를 화면에 그림
```

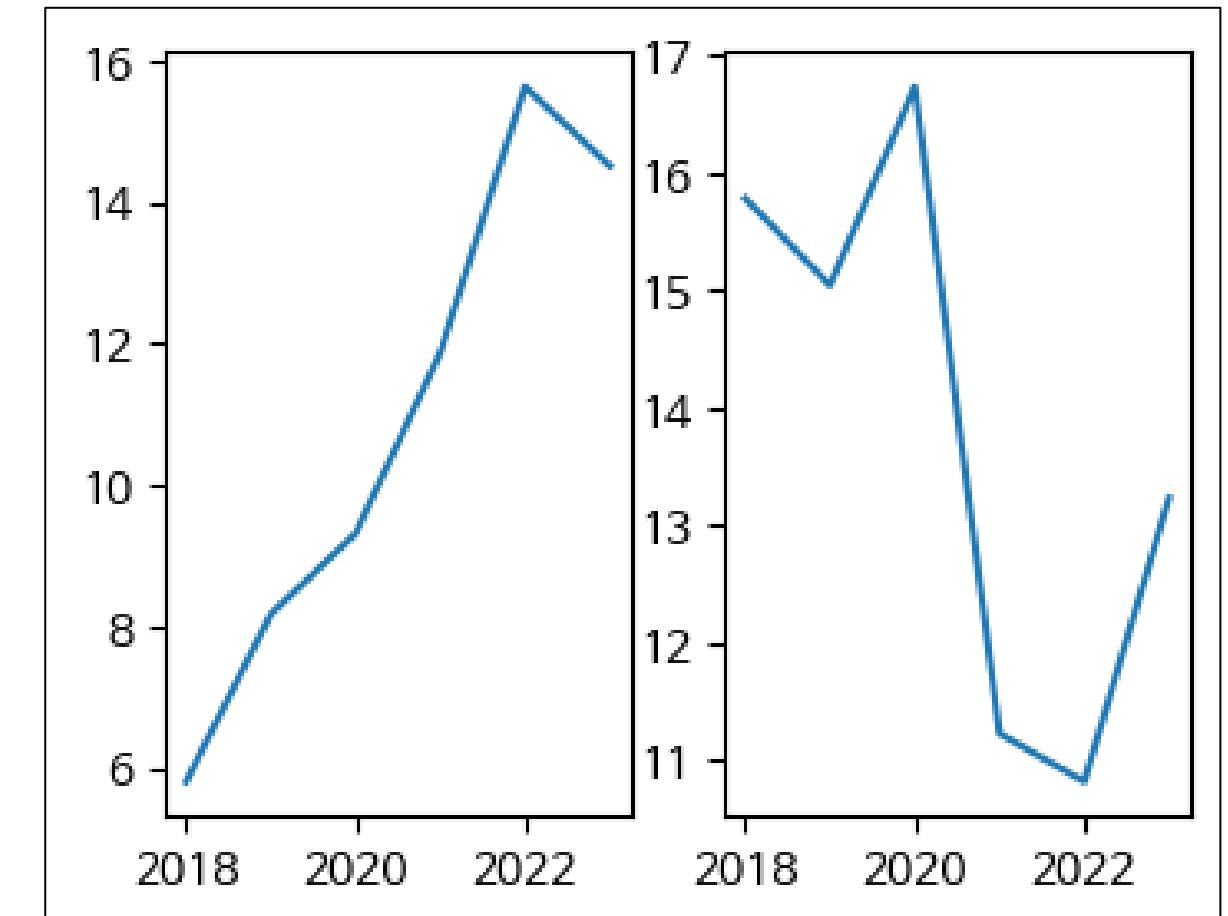
- 종이를 여러 개 사용하여 그래프를 표현



✓ 서브플롯을 사용

```
fig, (ax1, ax2) = plt.subplots(1, 2) # 가로 2칸, 세로 1칸 총 2개의 subplot을 생성  
# 생성된 subplot은 각각 ax1과 ax2에 저장  
  
ax1.plot(year, python) # ax1에 파이썬 선 그래프를 그림  
ax2.plot(year, java) # ax2에 자바 선 그래프를 그림  
  
plt.show() # 그래프를 표시
```

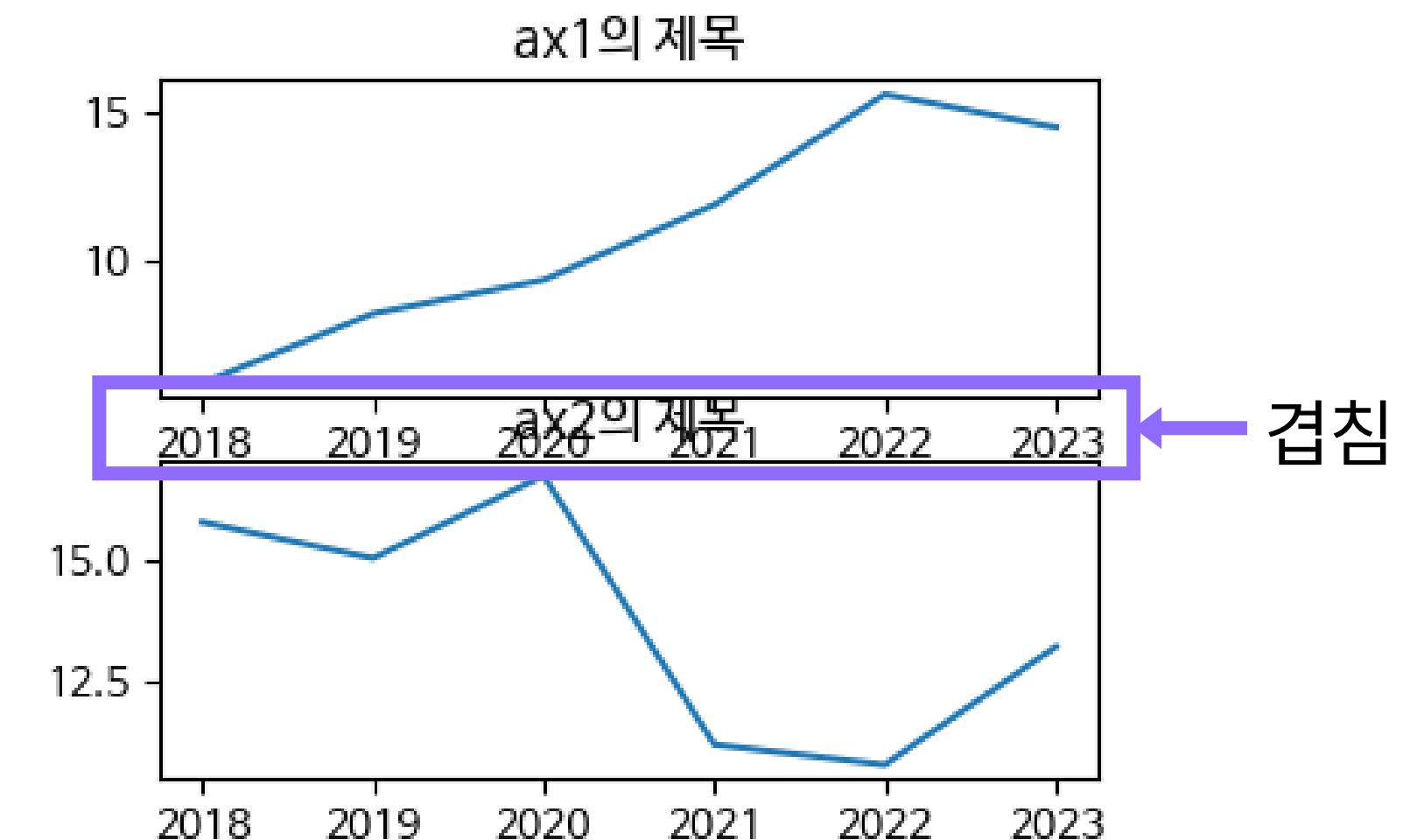
- 여러 개의 그래프를 하나의 figure에 그리는 방법
- ax1, ax2에도 비슷한 방법으로 그래프를 그림



✓ 서브플롯이 겹치는 현상

```
ax1.set_title("ax1의 제목") # ax1의 제목을 설정  
ax2.set_title("ax2의 제목") # ax2의 제목을 설정
```

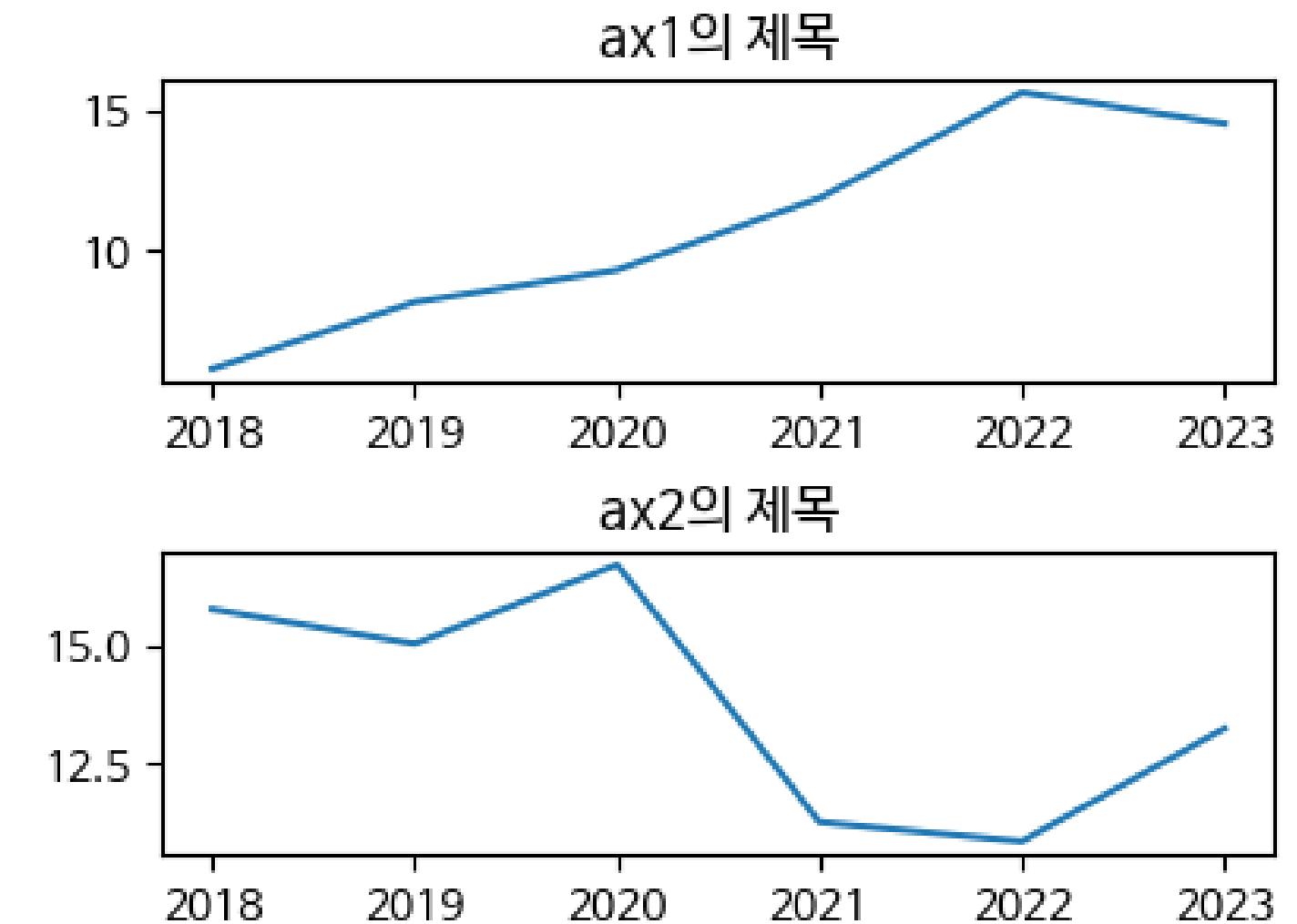
- 두 서브플롯의 간격이 충분하지 않아 일부가 겹치는 경우



⑤ constrained_layout 옵션

```
fig, (ax1, ax2) = plt.subplots(2, 1, constrained_layout=True)
# constrained_layout=True를 전달
ax1.plot(year, python) # ax1에 파이썬 선 그래프를 그립니다.
ax1.set_title("ax1의 제목") # ax1의 제목을 설정
ax2.plot(year, java) # ax2에 자바 선 그래프를 그립니다.
ax2.set_title("ax2의 제목") # ax2의 제목을 설정
```

- True를 전달하면 그래프 간의 위치를 고려하여 배치



02

Seaborn 기본 문법

✓ Seaborn



seaborn

- Matplotlib을 기반으로 한 Python 데이터 시각화 라이브러리
- Matplotlib만 사용할 때보다 시각적으로 더 완성도 높고 간단하게 시각화하는 함수를 제공

⑤ 라이브러리 불러오기

```
import matplotlib.pyplot as plt  
import seaborn as sns
```

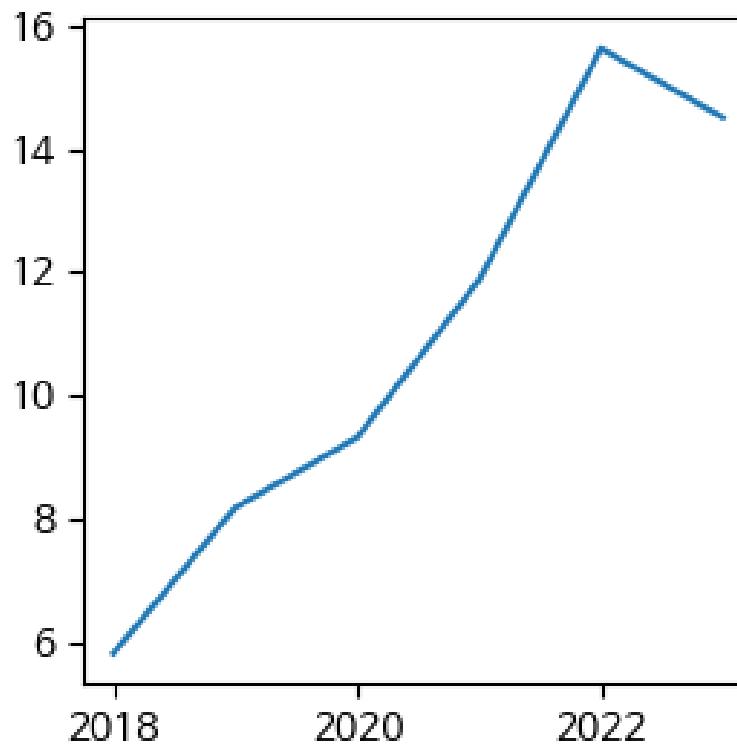
- 관용적으로 sns라는 이름으로 import
- 여전히 matplotlib의 함수들이 필요한 경우가 많으므로 함께 불러와야 함

 선 그래프 그리기

seaborn

```
sns.lineplot(x=year, y=python)
```

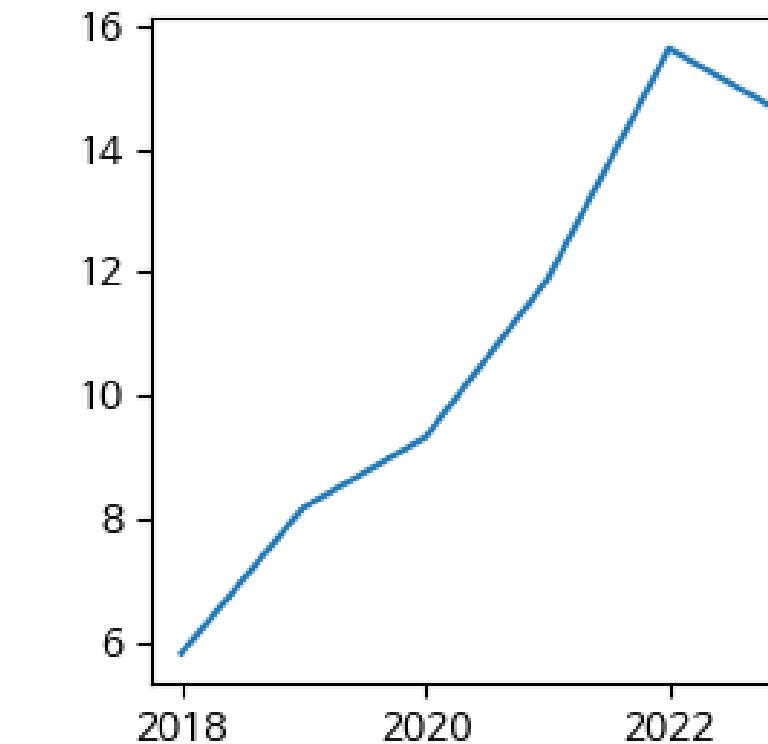
- lineplot() 함수로 그림
- x, y축에 해당하는 데이터를 전달



matplotlib

```
plt.plot(year, python)
```

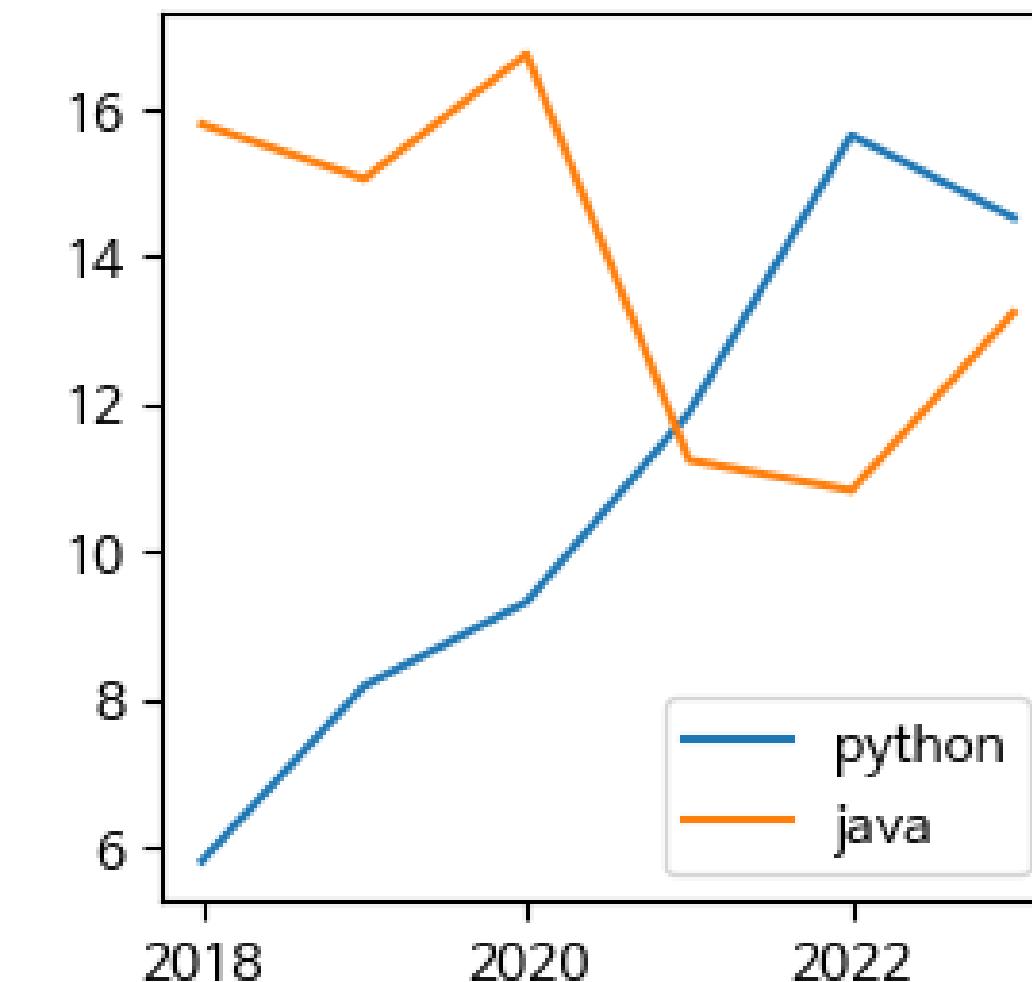
- plot() 함수로 그림
- x, y축에 해당하는 데이터를 전달



 그래프 범례 표시

```
sns.lineplot(x=year, y=python, label="python")
sns.lineplot(x=year, y=java, label="java")
```

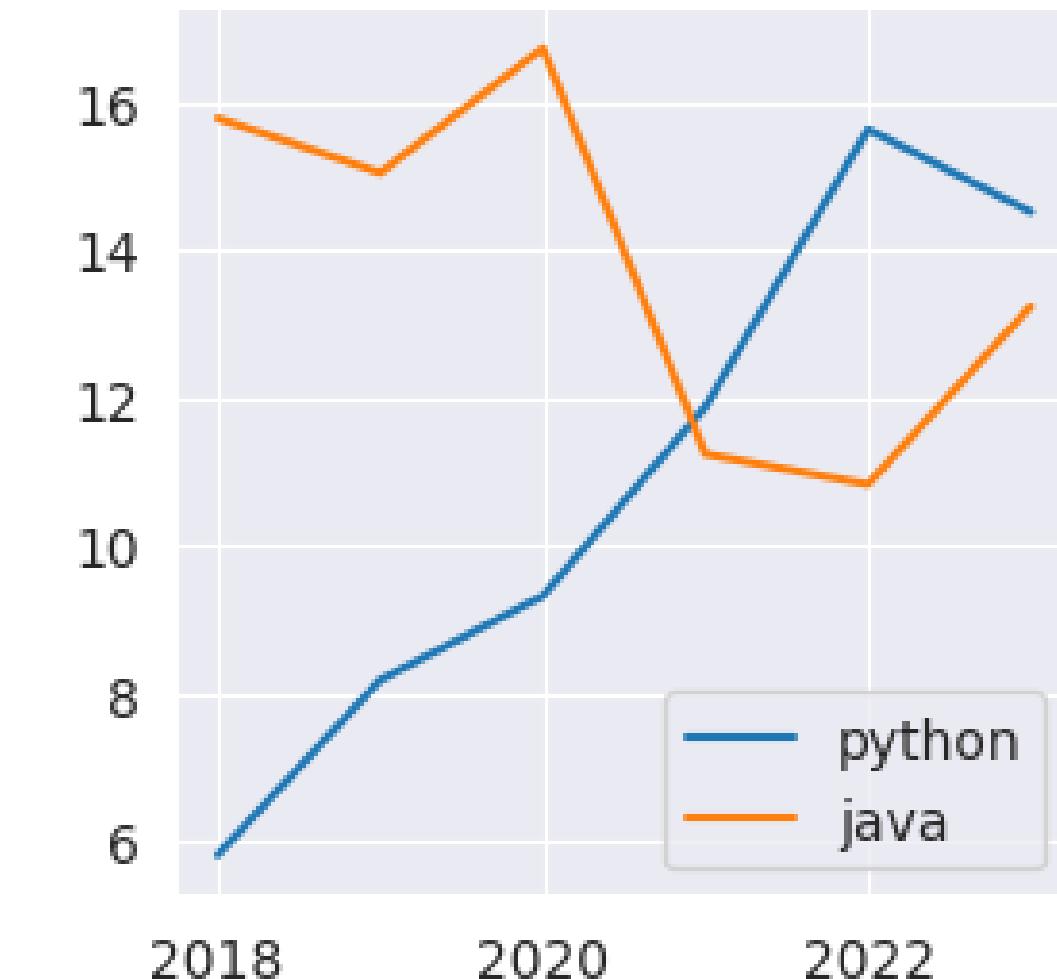
- legend()를 호출하지 않아도 범례를 표시함
- 이처럼 대부분의 매개변수가 matplotlib와 같음



⑤ 테마 사용하기

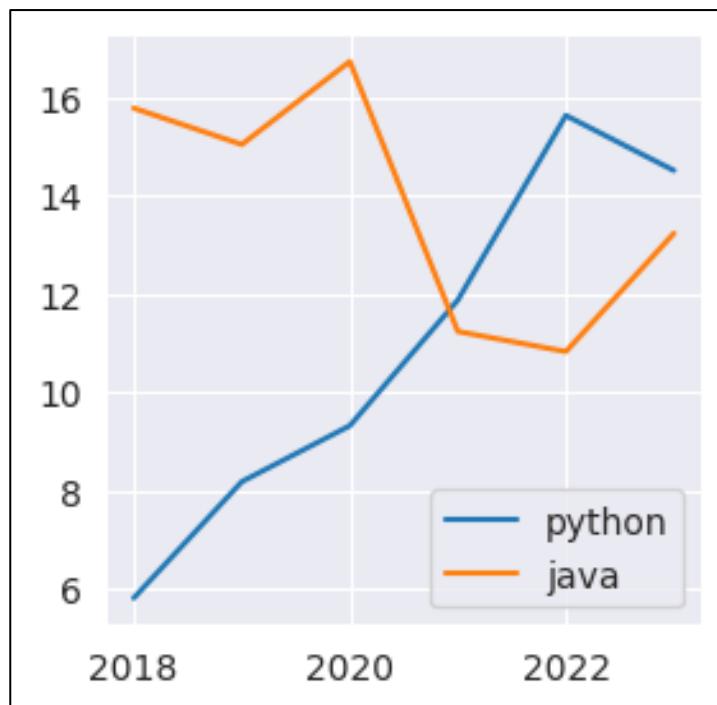
```
sns.set_style("darkgrid")
sns.lineplot(x=year, y=python, label="python")
sns.lineplot(x=year, y=java, label="java")
```

- 내장된 테마의 이름을 입력하여 스타일을 변경
- 더 완성도 높은 시각화 그림을 그릴 수 있음

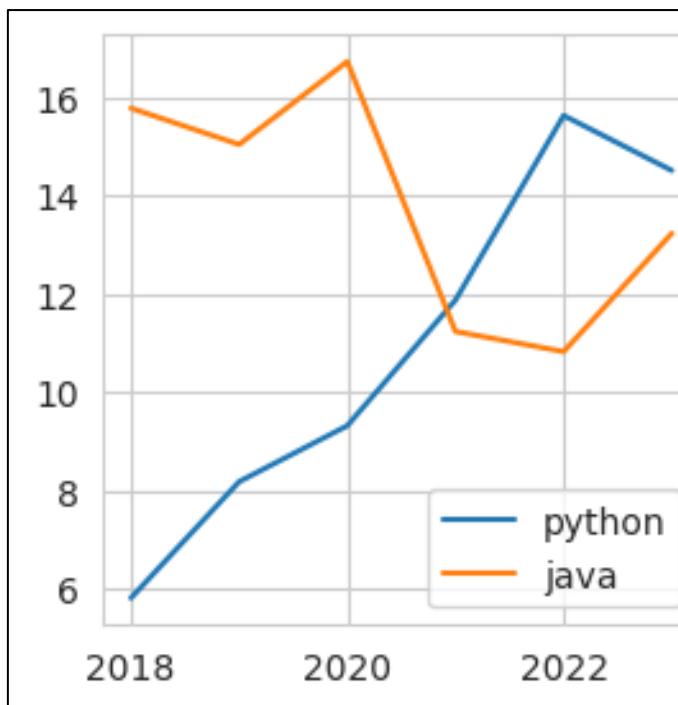


✓ 내장 테마 5가지

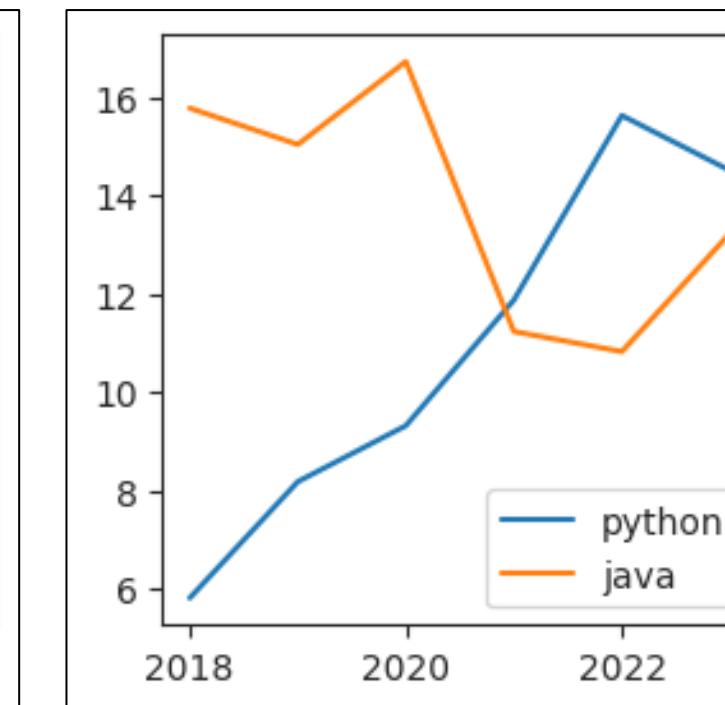
darkgrid



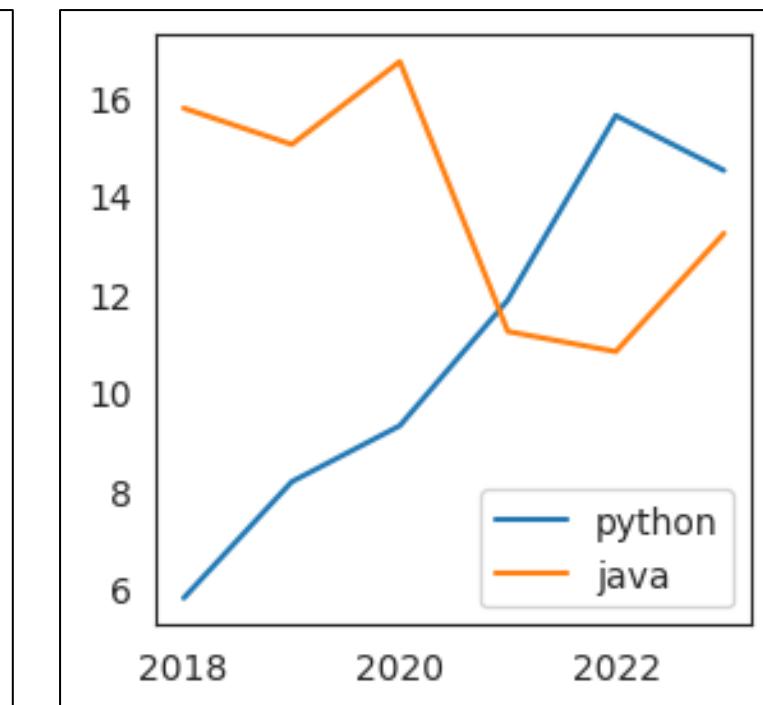
whitegrid



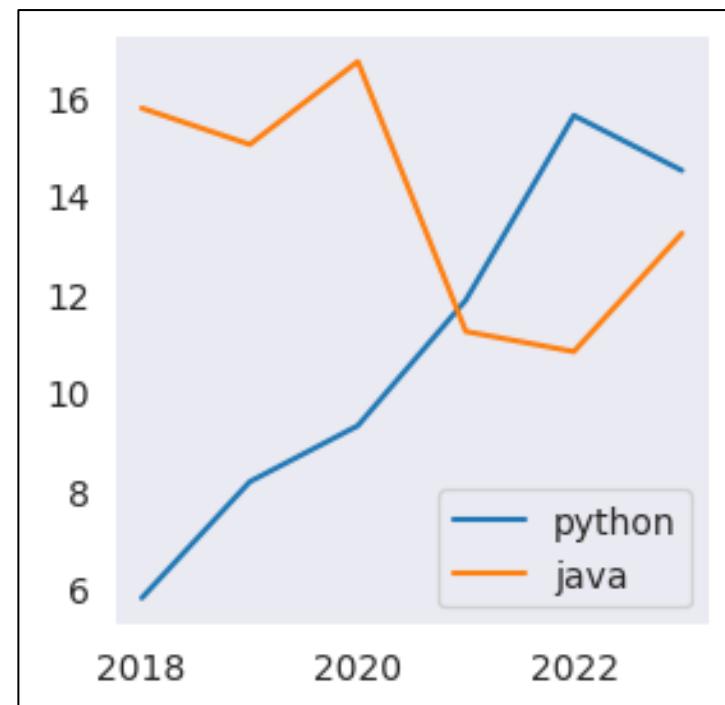
ticks



white



dark



03

선 그래프와 막대 그래프

⑤ 선 그래프

가로축과 세로축을 갖는 좌표 평면 상에 데이터를 선으로 표현한 그래프

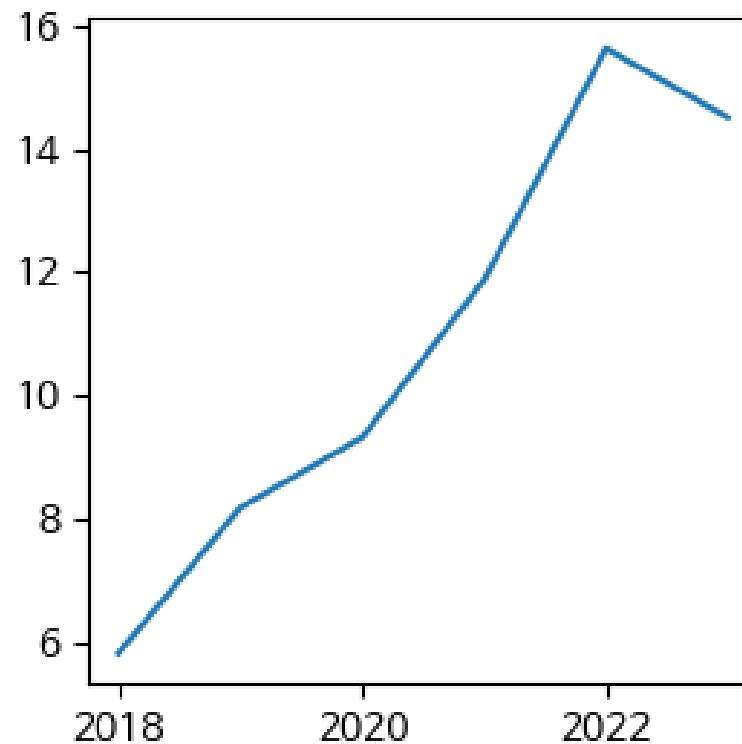
- 막대 그래프와 함께 복수의 데이터를 비교하는데 유용한 시각화 방법
- 시간, 순서, 변화 등에 따른 데이터의 추이와 패턴을 시각적으로 파악하기에 유용
- 간단하게 그려볼 수 있는 그래프

✓ 선 그래프 그리기

seaborn

```
sns.lineplot(x=year, y=python)
```

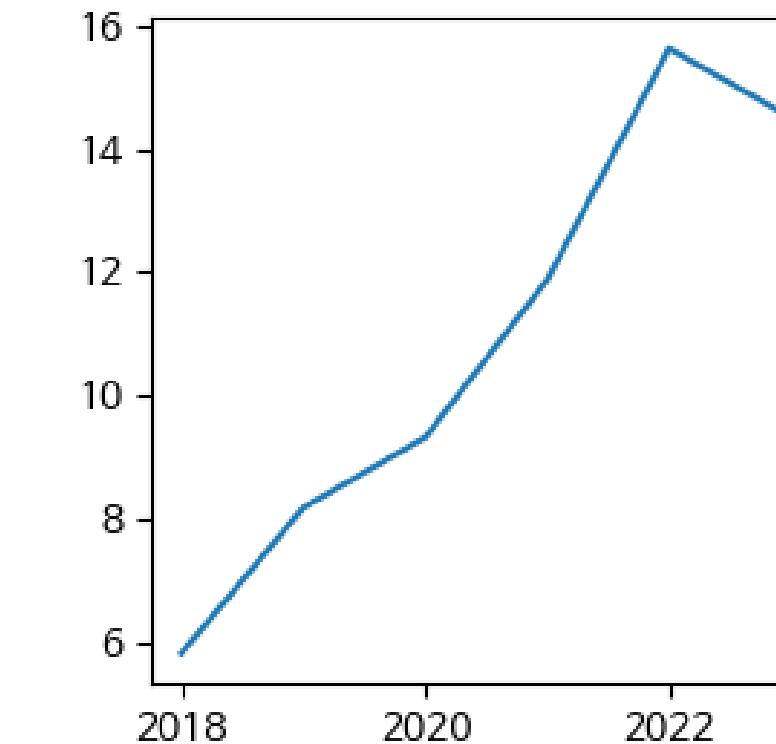
- lineplot() 함수로 그림
- x, y축에 해당하는 데이터를 전달



matplotlib

```
plt.plot(year, python)
```

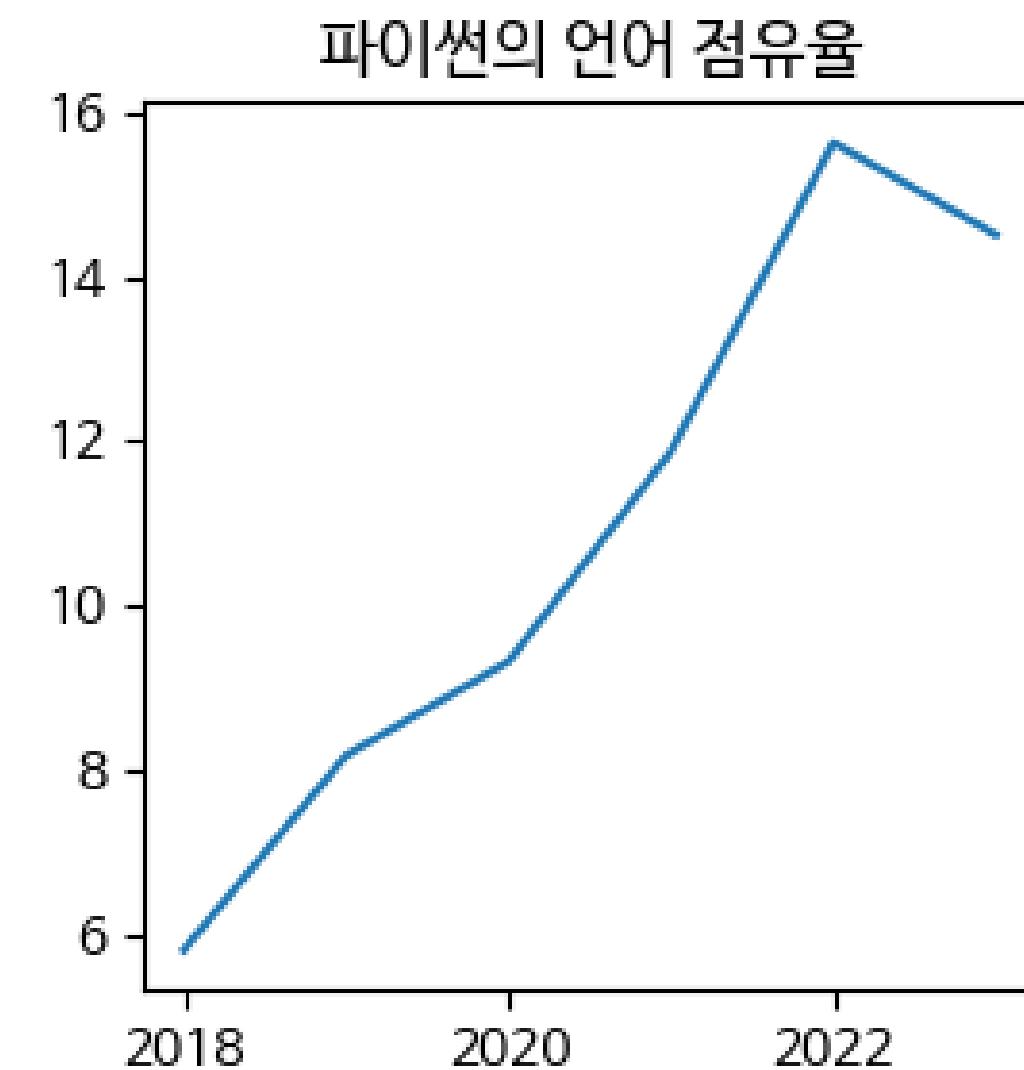
- plot() 함수로 그림
- x, y축에 해당하는 데이터를 전달



✓ 축의 눈금 조절하기

```
plt.title("파이썬의 언어 점유율")
plt.plot(year, python, label="Python")
```

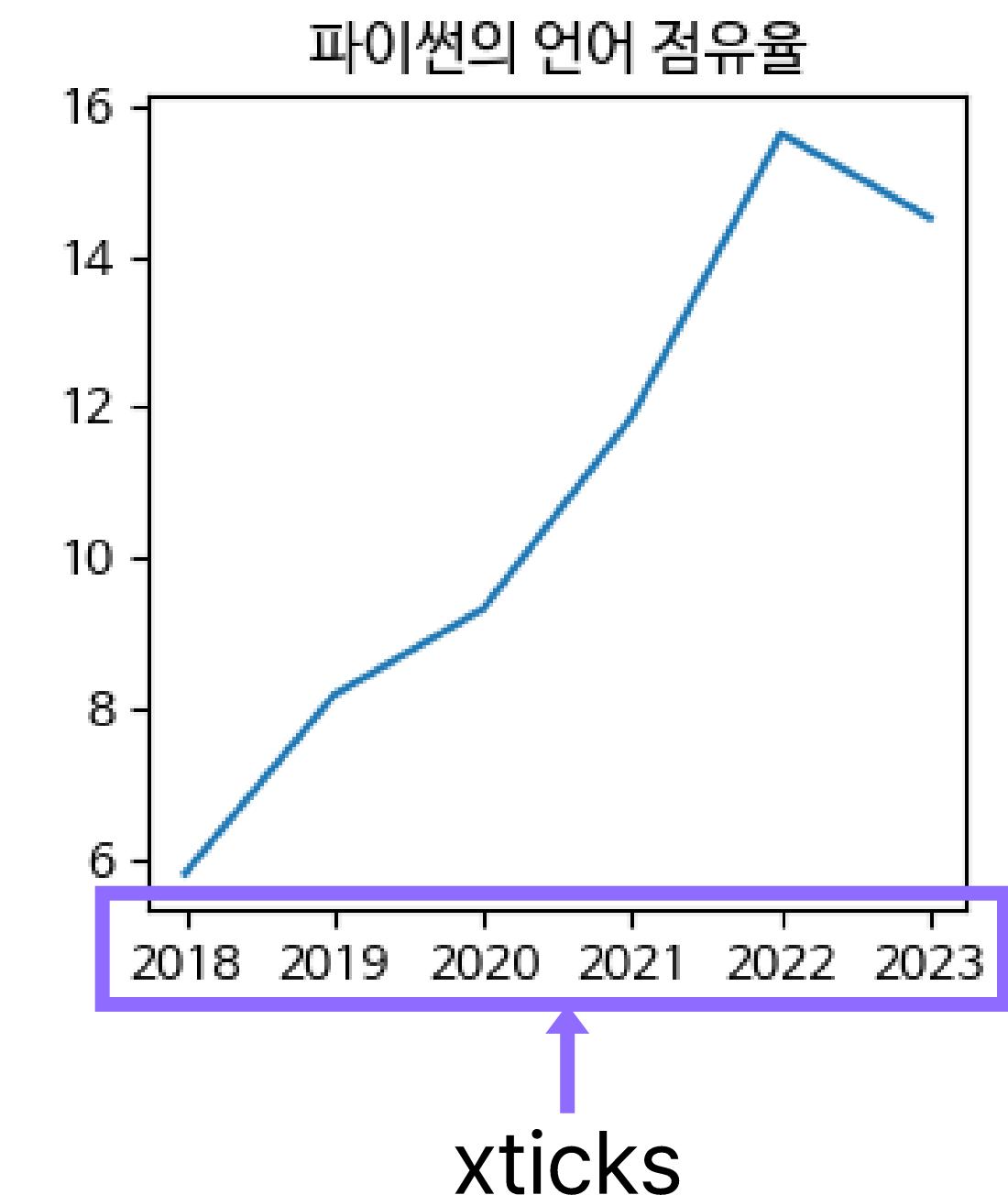
- 연도별로 보고 싶지만, x축 눈금이 자동으로 설정됨



✓ 축의 눈금 조절하기

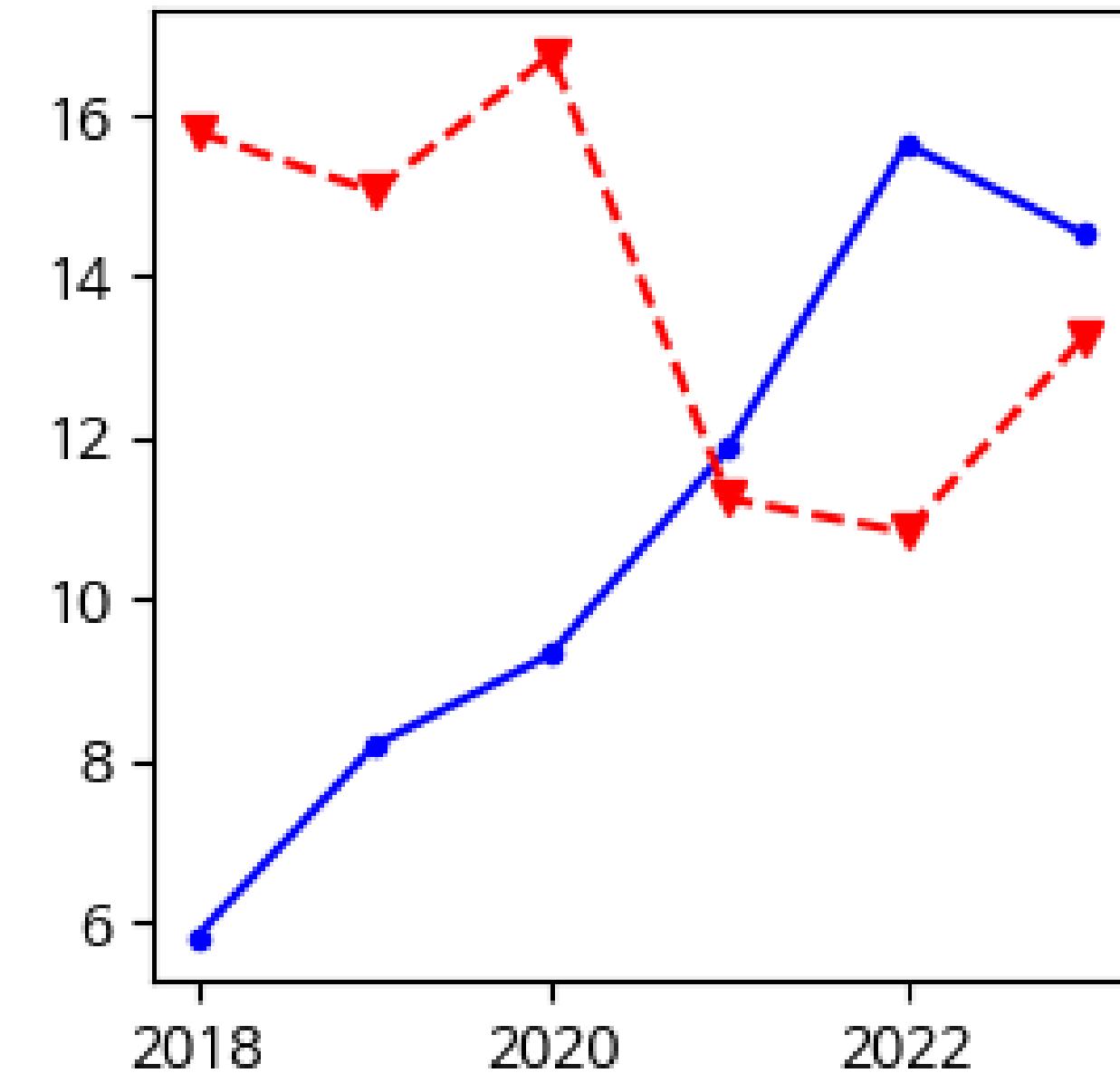
```
plt.title("파이썬의 언어 점유율")
plt.plot(year, python, label="Python")
plt.xticks([2018, 2019, 2020, 2021, 2022, 2023]) # x축 눈금
```

- xticks에 눈금의 숫자를 리스트로 전달
- y축의 눈금은 yticks로 설정



✓ 선 그래프의 모양

- 좀 더 직관적인 시각화를 위해 그래프의 모양을 변경
- 마커 (marker): 그래프의 데이터 포인트를 나타내는 작은 심볼
- 선 모양 (linestyle): 선의 모양의 설정
- 색 (color): 그래프의 색을 설정

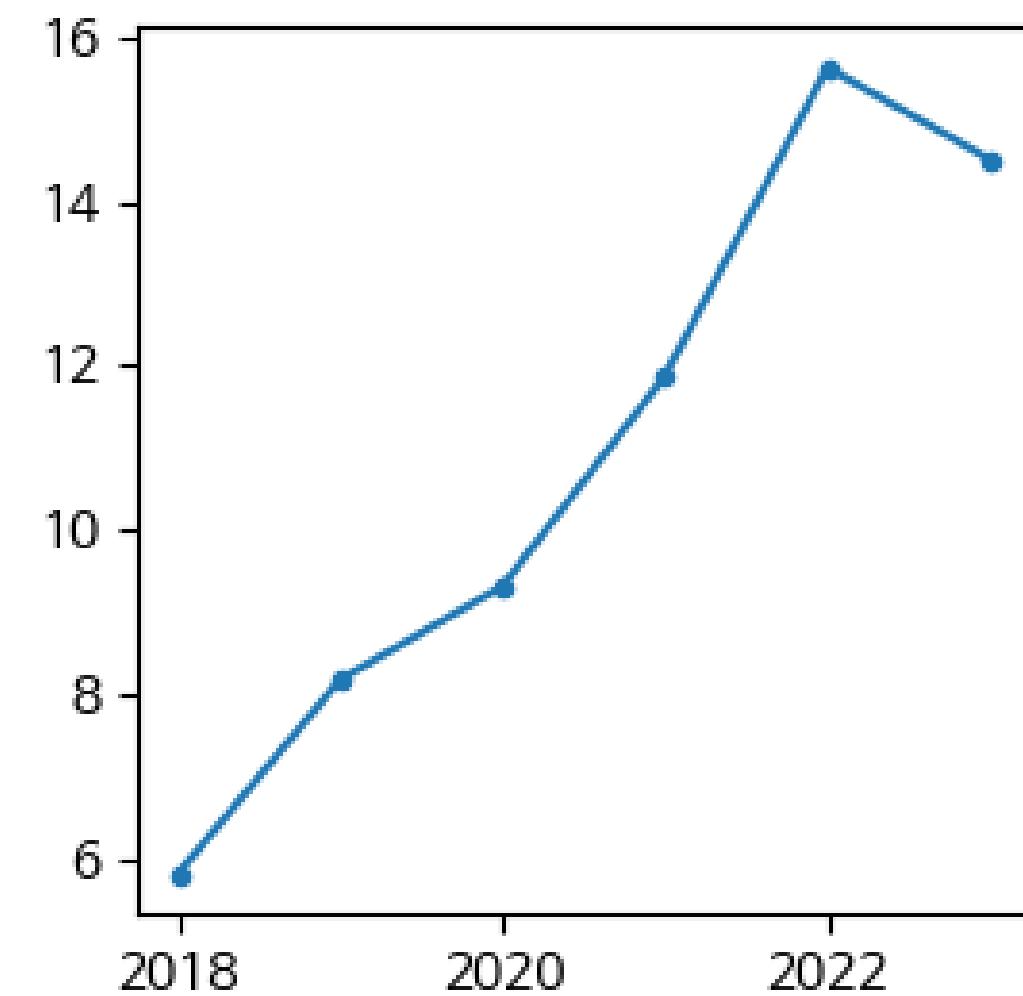


✓ 마커 변경하기

```
plt.plot(year, python, label="Python", marker=".")
```

- marker 매개변수에 모양을 전달하여 설정
- 모든 마커의 모양은 공식 문서를 참고

https://matplotlib.org/stable/api/markers_api.html

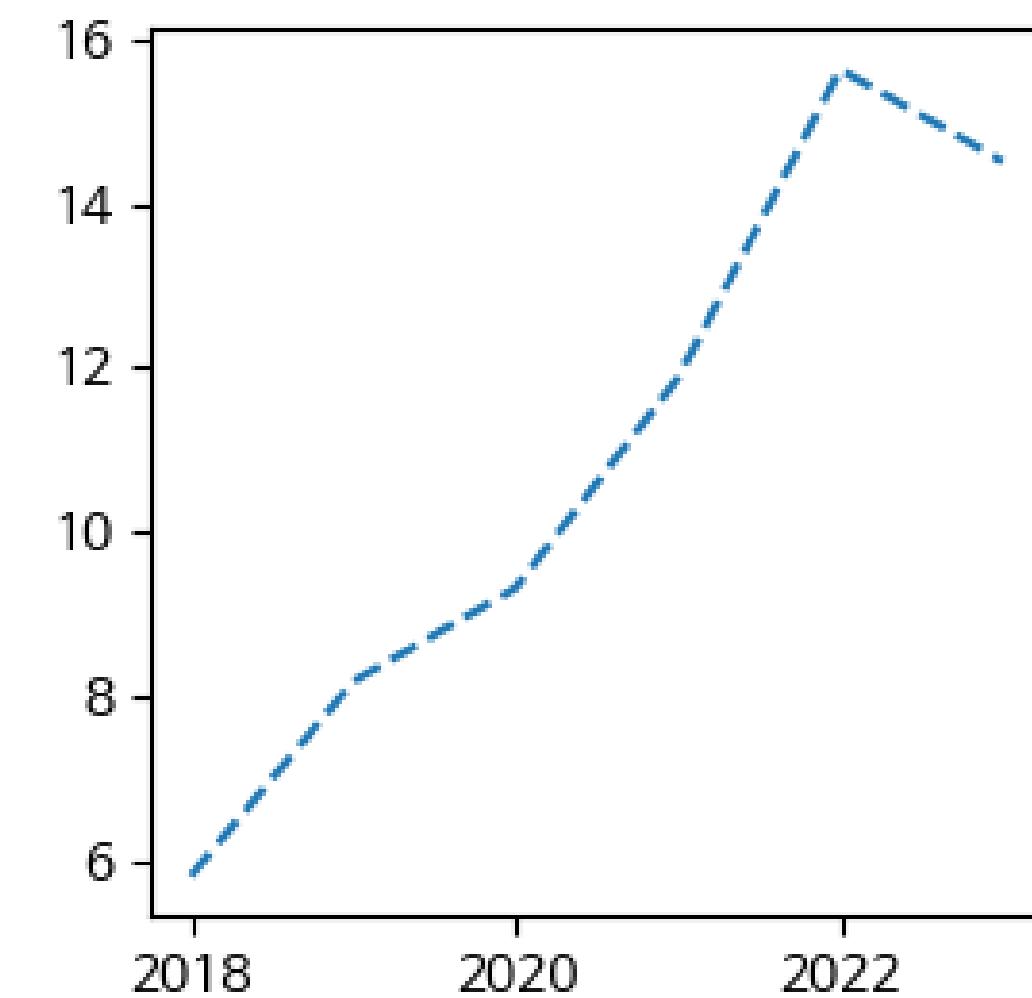


✓ 선 모양 변경하기

```
plt.plot(year, python, label="Python", linestyle="--")
```

- `linestyle` 매개변수에 모양을 전달하여 설정
- 모든 선의 모양은 공식 문서를 참고

https://matplotlib.org/stable/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D.set_linestyle

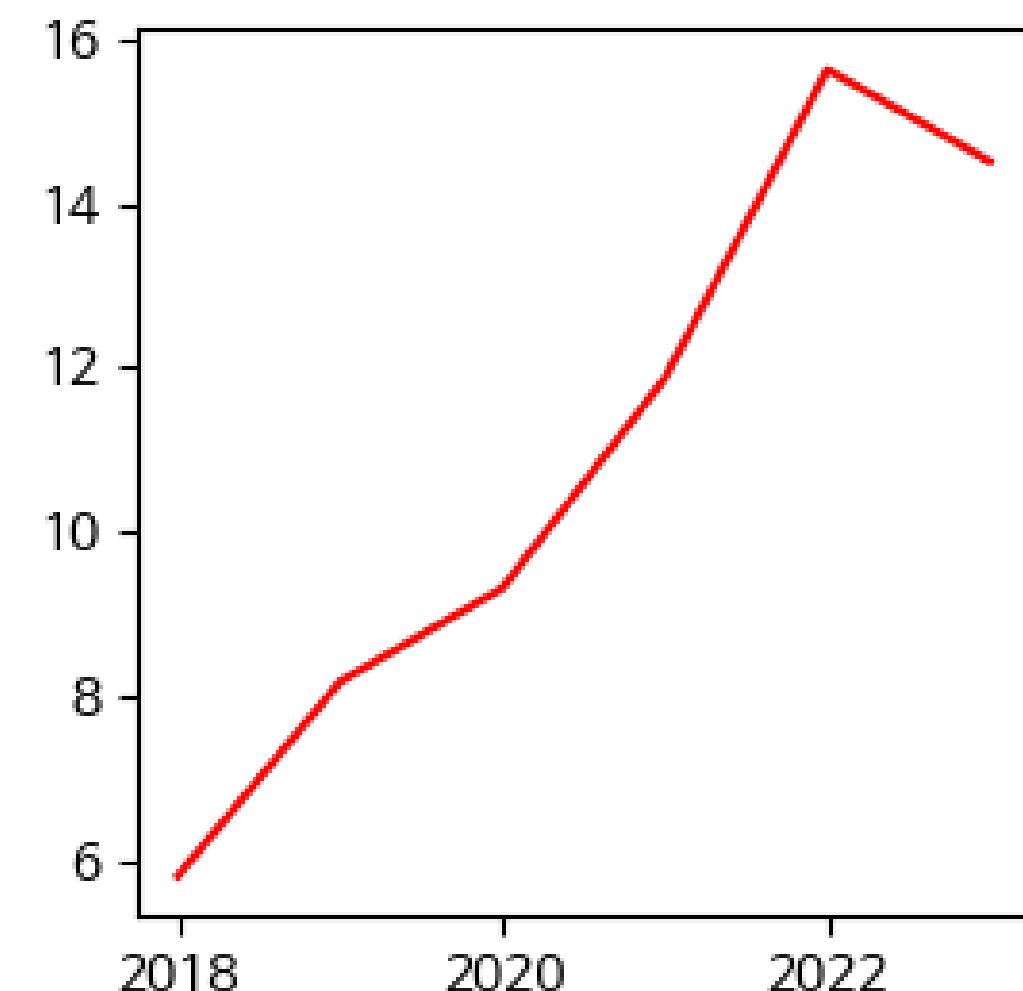


✓ 그래프 색 변경하기

```
plt.plot(year, python, label="Python", color="r")
```

- color 매개변수에 색의 이름을 전달하여 설정
- 선 그래프뿐 아니라 다양한 그래프에 적용 가능
- 모든 색은 공식 문서를 참고

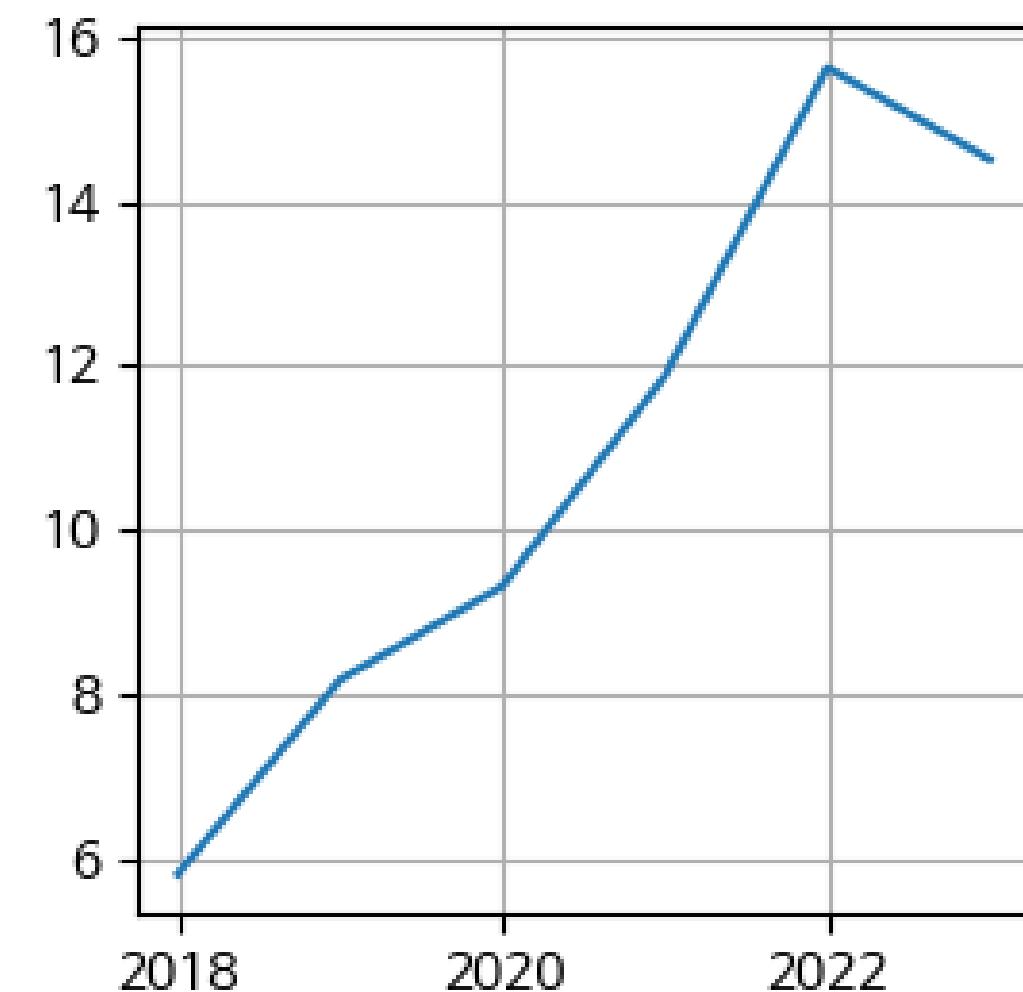
<https://matplotlib.org/stable/tutorials/colors/colors.html>



✓ 격자 추가하기

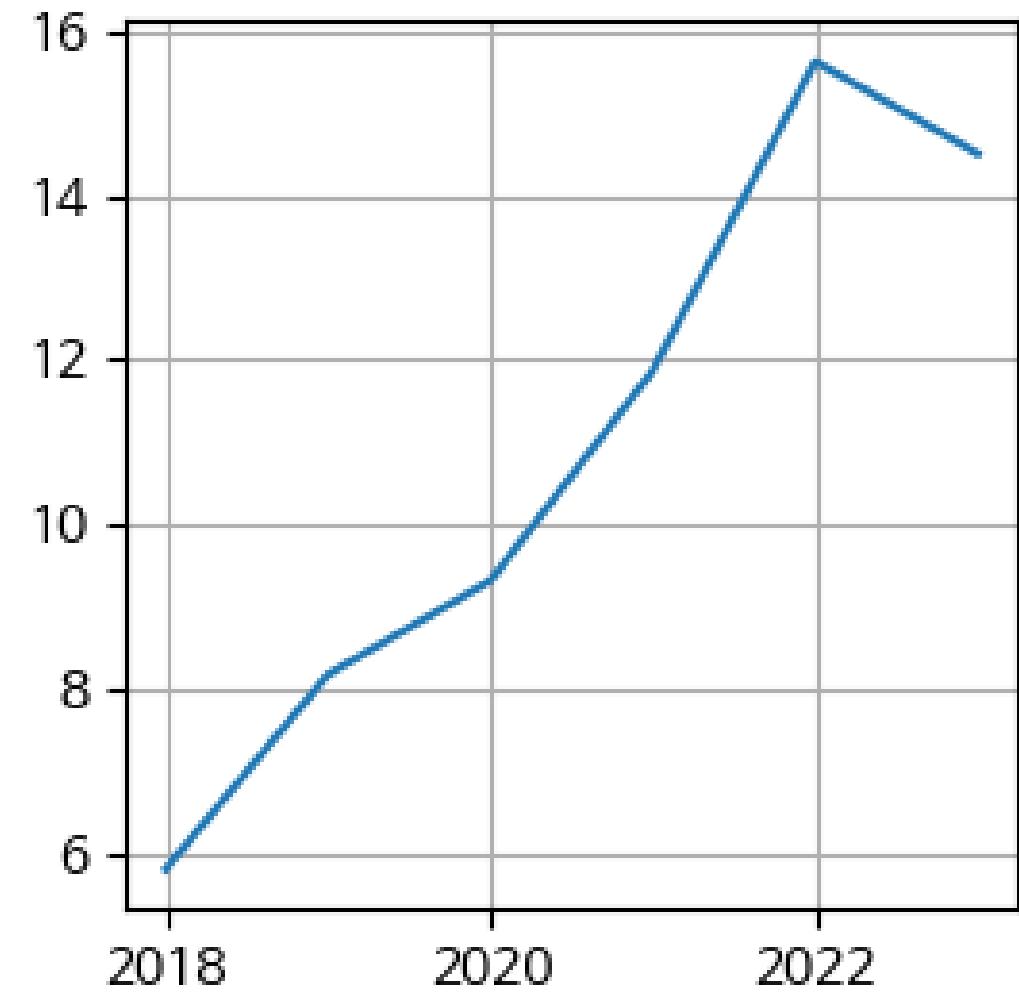
```
plt.plot(year, python, label="Python")
plt.grid(True)
```

- grid 매개변수에 True를 전달
- axis 매개변수를 통해 옵션을 설정할 수 있음
- axis의 기본값은 "both"

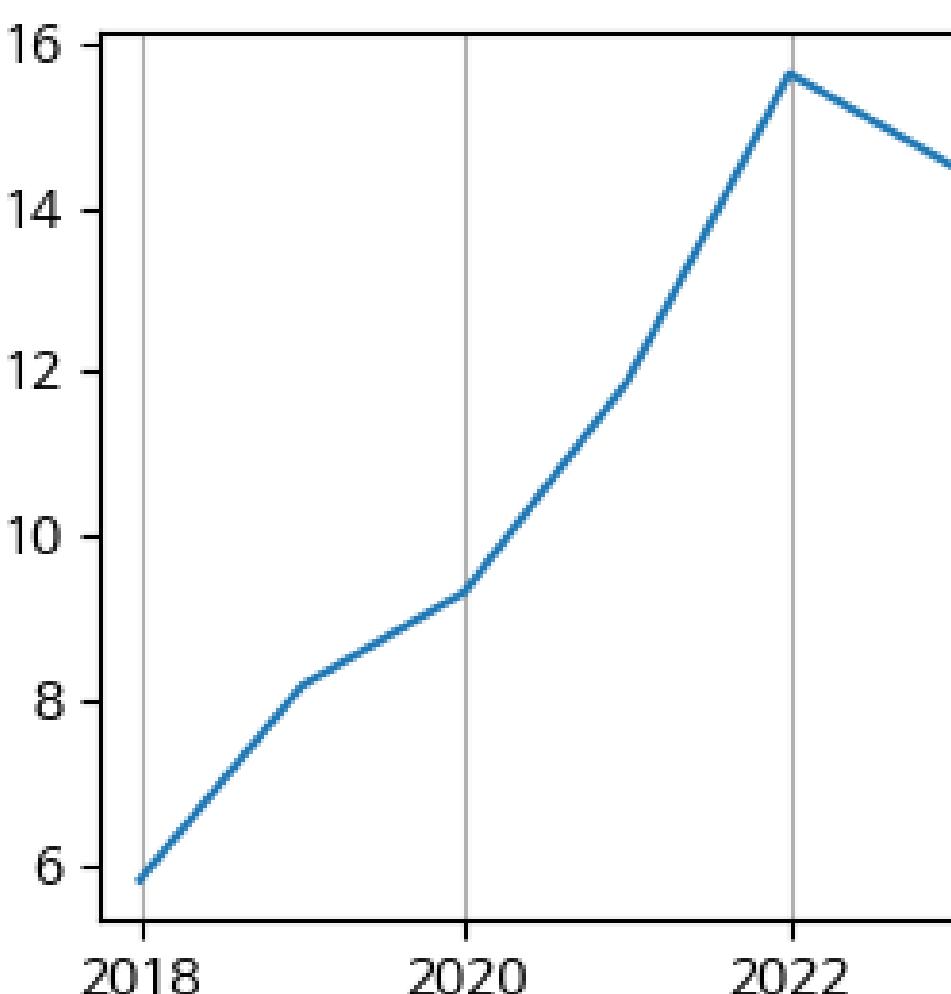


✓ 격자의 옵션

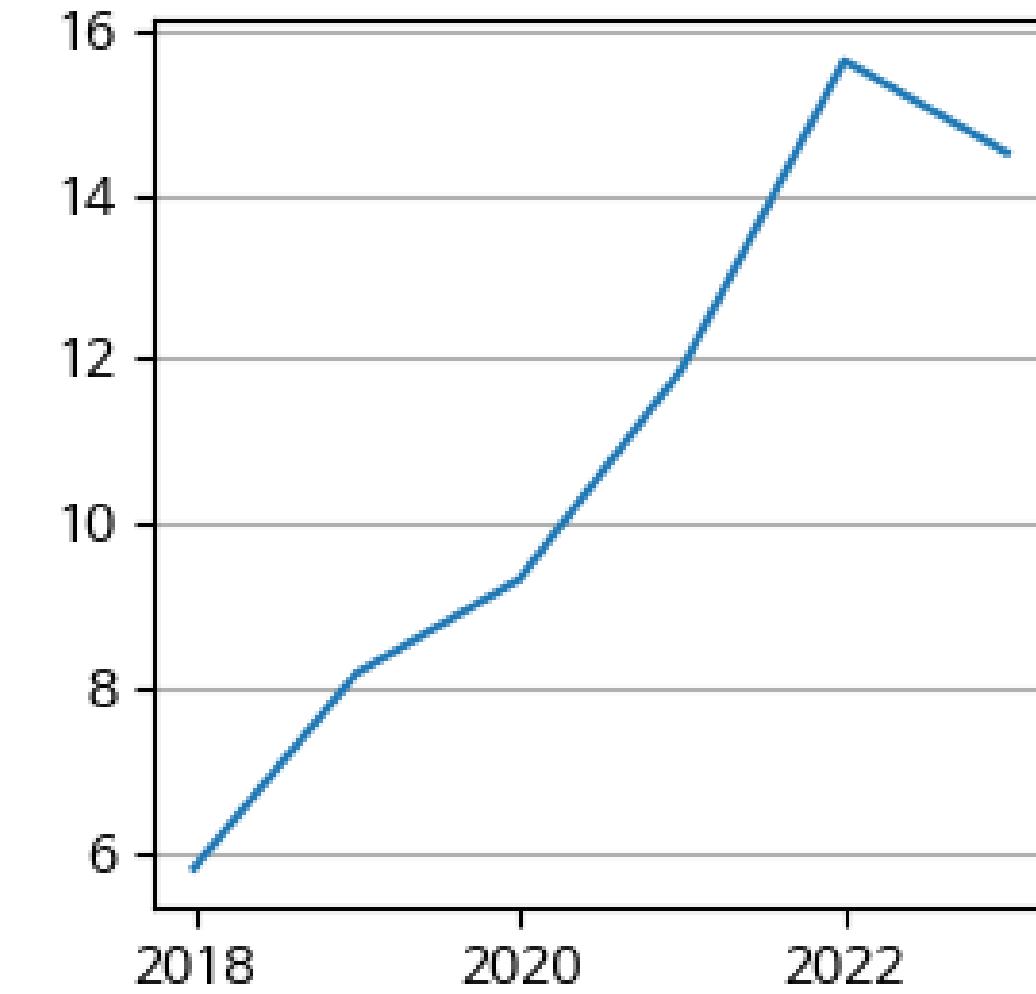
```
plt.grid(True, axis="both")
```



```
plt.grid(True, axis="x")
```



```
plt.grid(True, axis="y")
```



⑤ 데이터 묶기 (groupby)

```
gb_mon = df.groupby("월") # 월별로 묶기  
df_mon = gb_mon.mean(numeric_only=True) # 숫자형 열만 평균을 계산
```

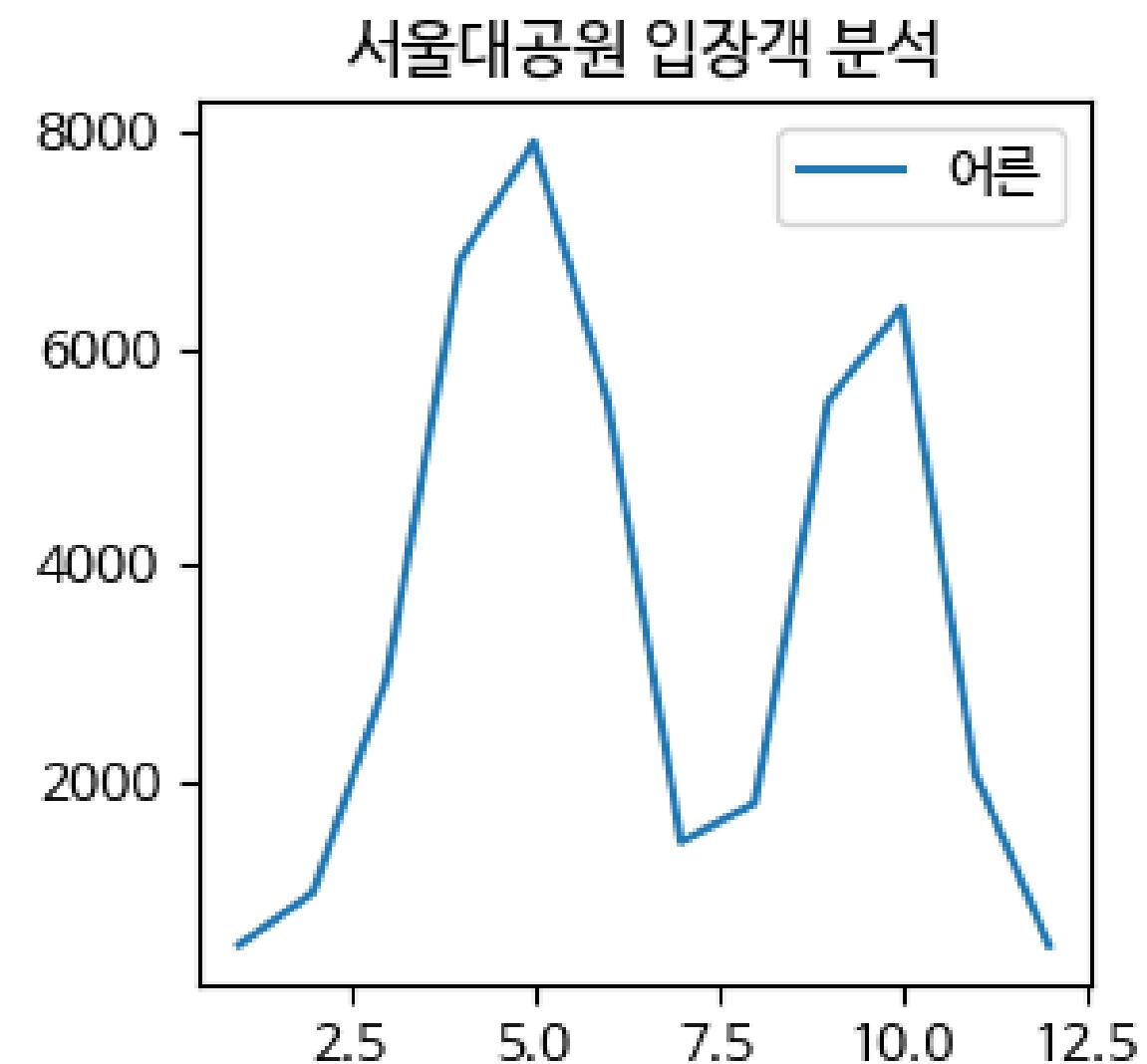
Unnamed: 0	어른	청소년	어린이	외국인	단체	총입장객수	연	일	매출액
월									
1	557.333333	497.892473	140.623656	79.784946	17.978495	144.516129	1375.258065	2017.666667	16.000000
2	580.623529	978.129412	128.247059	150.552941	41.447059	83.364706	2077.529412	2017.647059	14.670588
3	605.814433	2957.432990	209.000000	386.350515	41.123711	193.494845	5105.938144	2017.639175	16.556701

- 월별 데이터 시각화를 위해 데이터를 묶고 수치형 데이터만 평균을 구함

✓ Matplotlib로 선 그래프 그리기

```
plt.title("서울대공원 입장객 분석") # 그래프 제목  
plt.plot(df_mon["어른"], label="어른") # 월별 어른 이용객의 수  
plt.legend()  
plt.show()
```

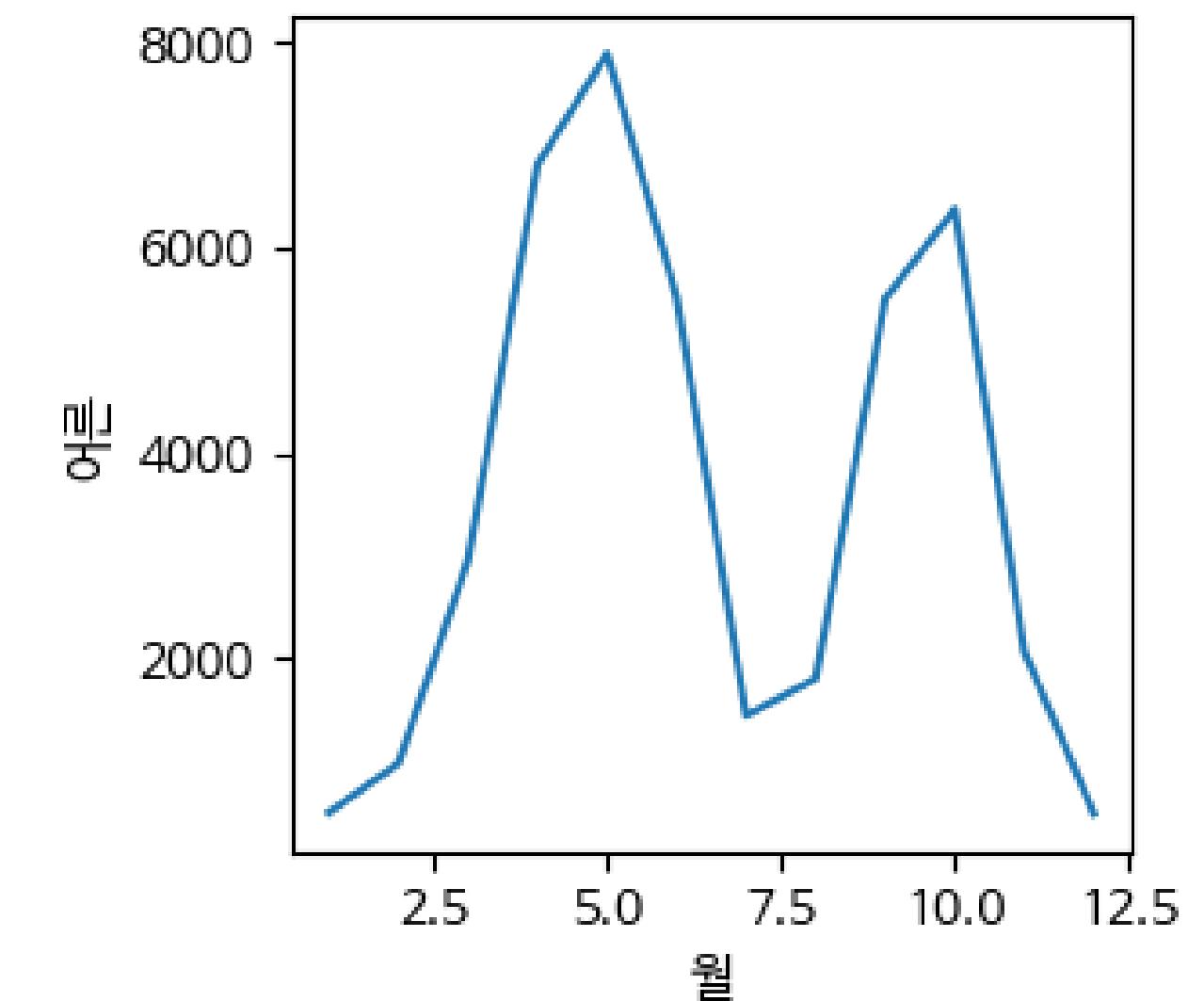
- 월별 그래프를 위해 데이터를 뭍어줌
- plot함수를 통해 그래프를 그림



✓ Seaborn으로 선 그래프 그리기

```
sns.lineplot(data=df, x="월", y="어른", errorbar=None)
```

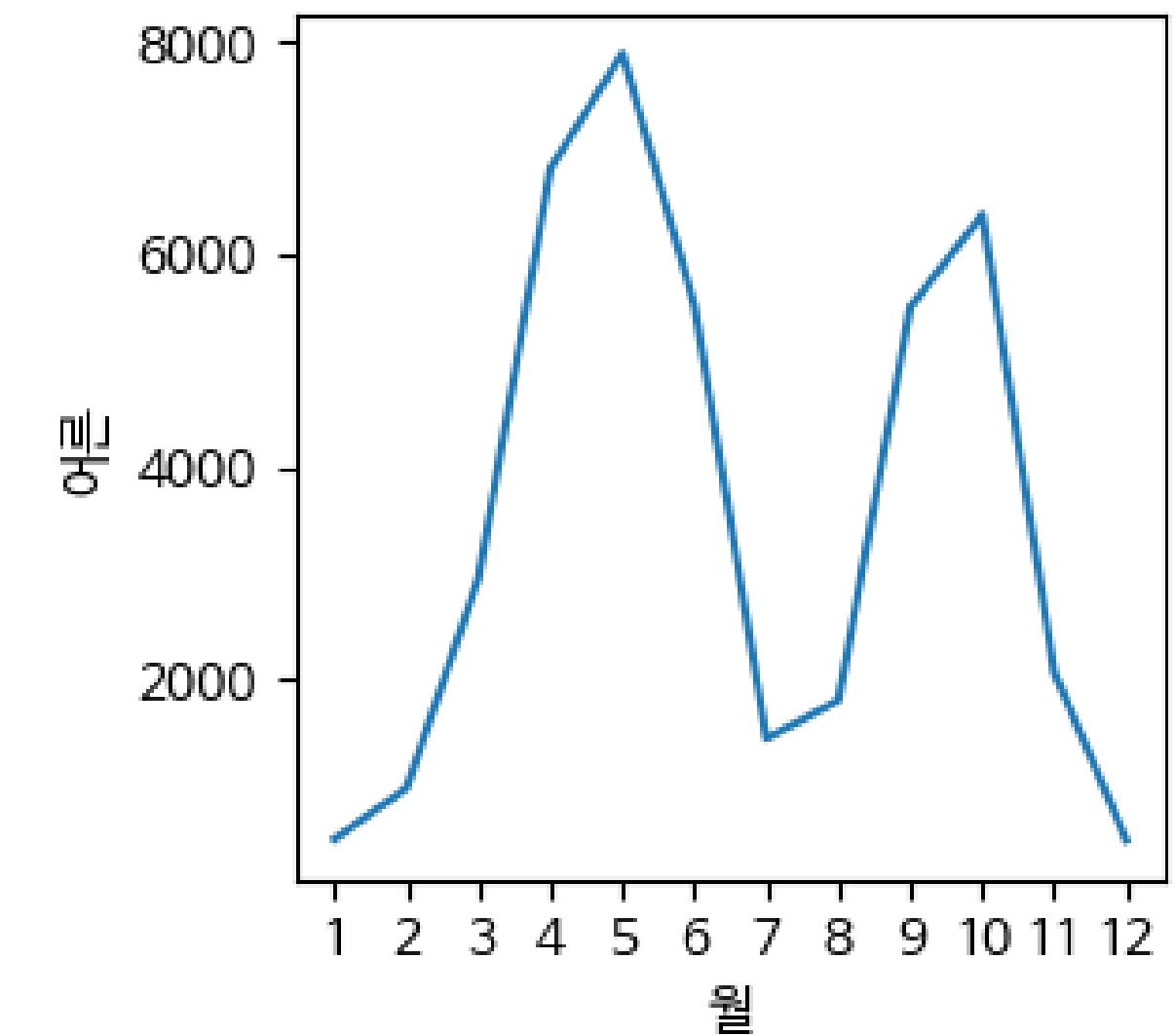
- 데이터를 뭍어줄 필요 없이 data 매개변수에 데이터프레임 전달
- x축은 “월”을 사용
- y축은 “어른”을 사용
- errorbar는 표시하지 않음



✓ Seaborn으로 선 그래프 그리기

```
sns.lineplot(data=df, x="월", y="어른", errorbar=None)  
plt.xticks([mon for mon in range(1, 13)]) # 1부터 12까지 전달
```

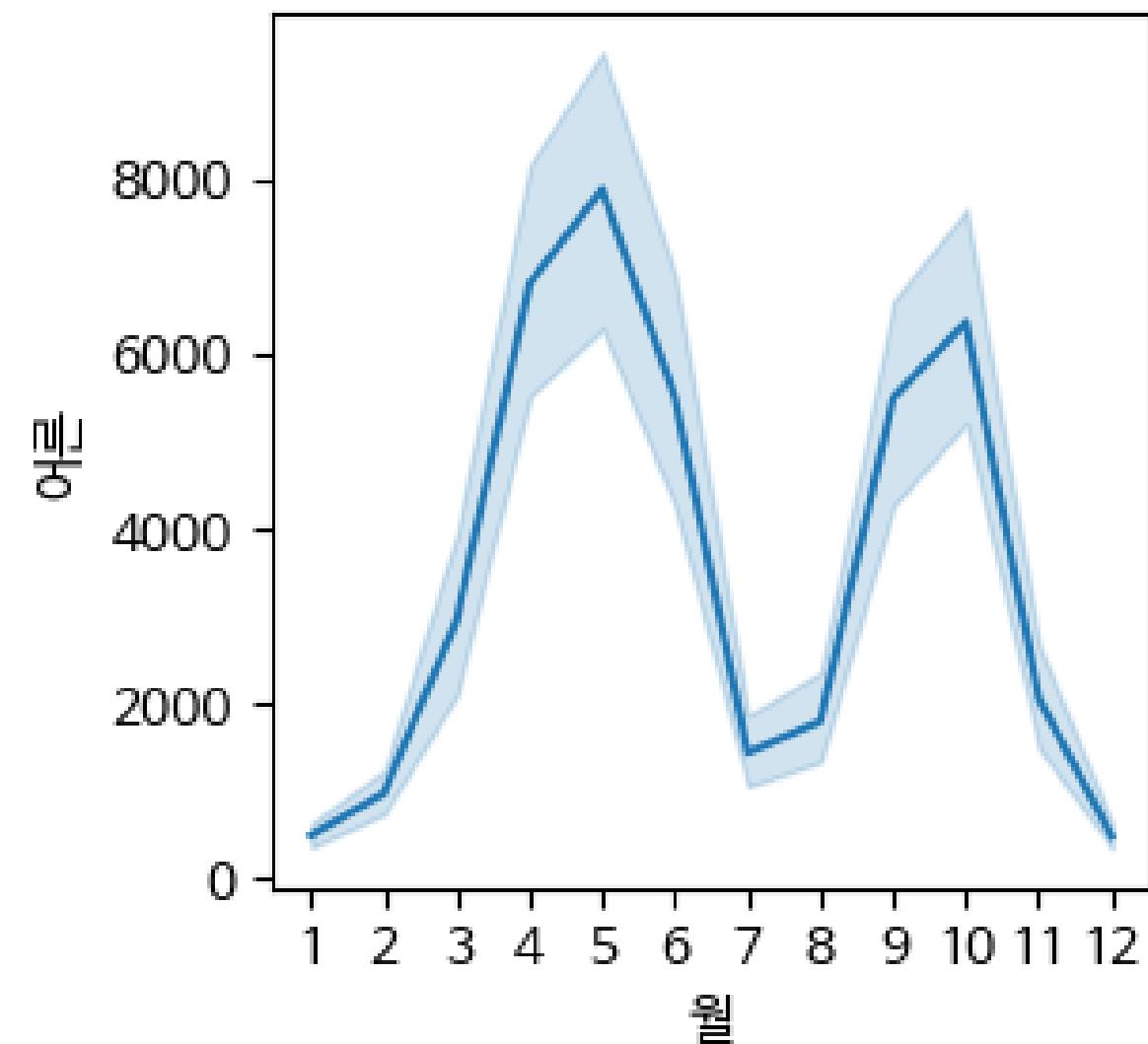
- seaborn을 사용하더라도 많은 옵션은 matplotlib로 설정
- 시각화 그래프는 seaborn으로, 옵션 조절은 plt로 설정



✓ 오차막대 (error bar)

```
sns.lineplot(data=df, x="월", y="어른")
```

- errorbar를 None으로 전달하지 않으면 오차 막대를 표시
- 오차 막대는 데이터의 편차를 표시하기 위한 그래프
- 기본값은 95% 신뢰구간의 오차 범위를 표시함
- 필요 없다면 None을 전달하여 삭제



⑤ 막대 그래프

가로축과 세로축을 갖는 좌표 평면 상에 데이터를 막대로 표현한 그래프

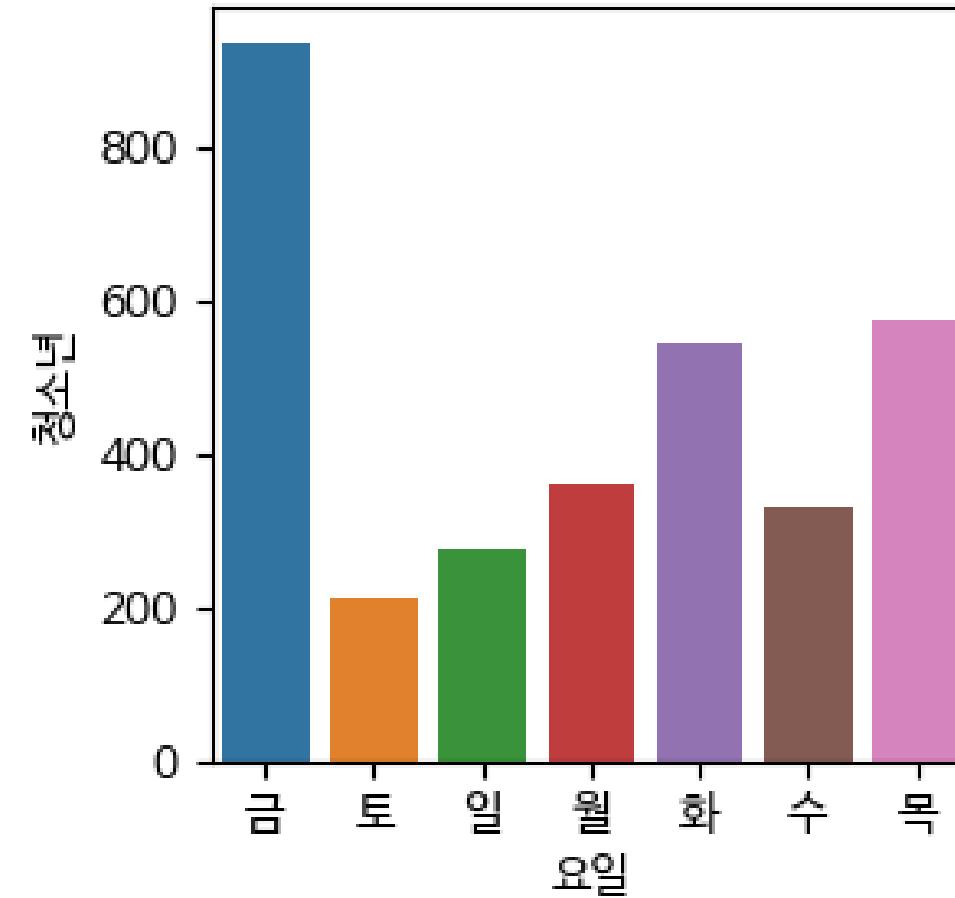
- 복수의 데이터를 비교하는데 유용한 시각화 방법
- 값의 크고 작음을 한눈에 파악하기 용이한 시각화 형태

✓ 막대 그래프 그리기

seaborn

```
sns.barplot(data=df, x="요일", y="청소년", errorbar=None)
```

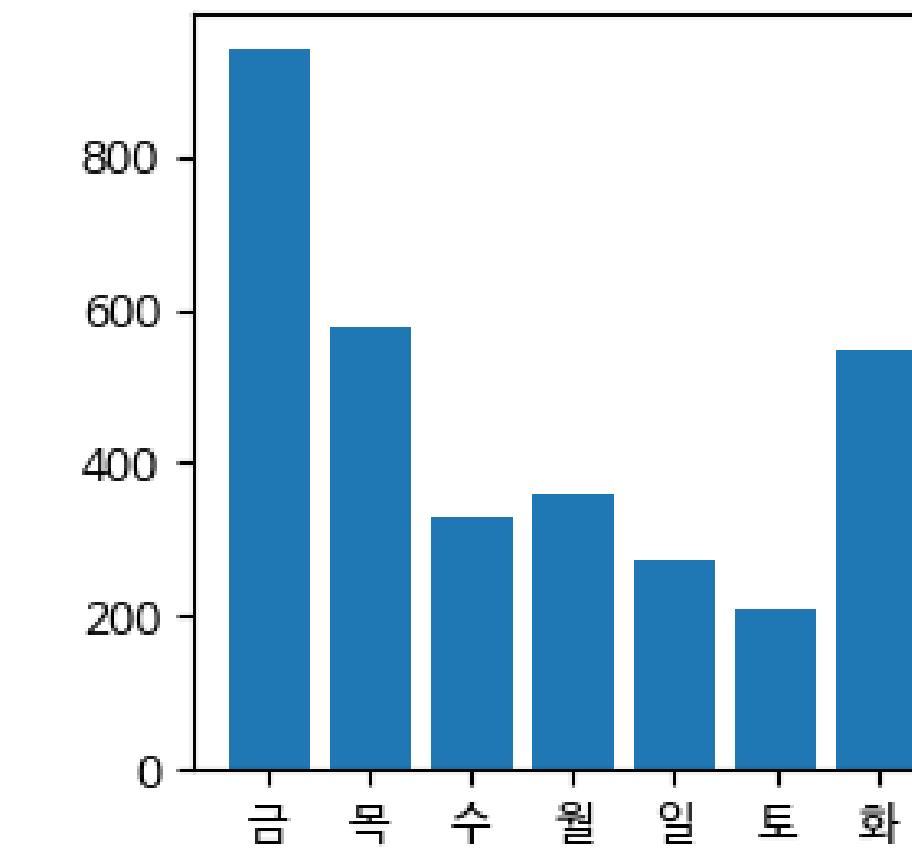
- x, y축에 해당하는 데이터를 전달
- 칼럼 이름으로 전달 가능



matplotlib

```
plt.bar(df_week["요일"], df_week["청소년"])
```

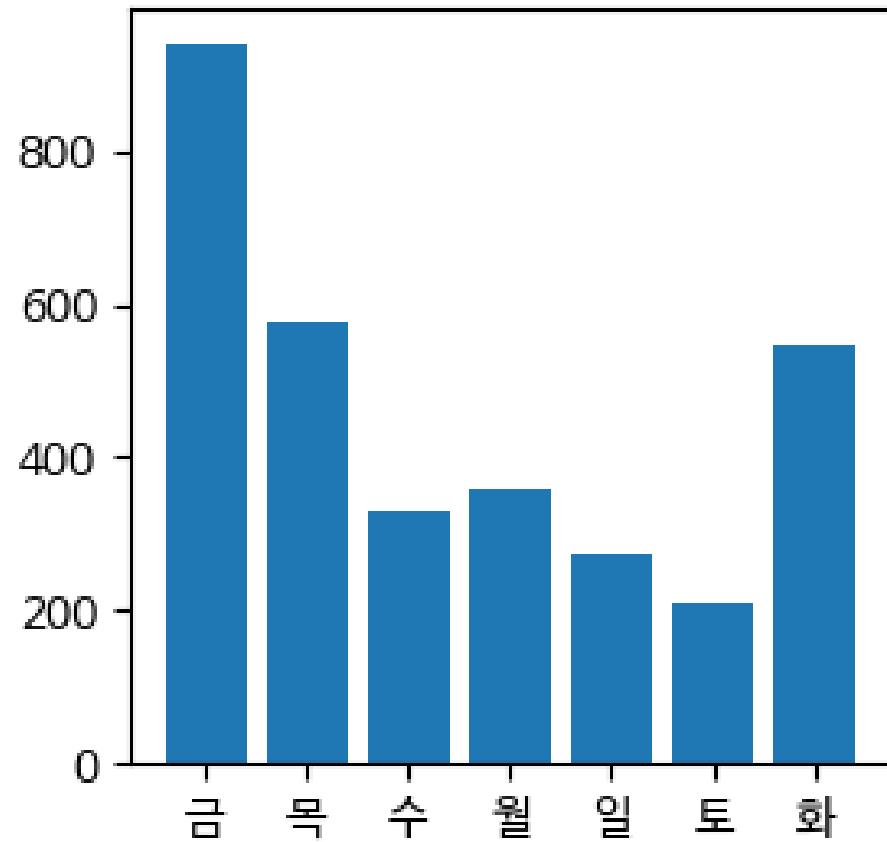
- x, y축에 해당하는 데이터를 전달



✓ matplotlib으로 막대 그래프 그리기

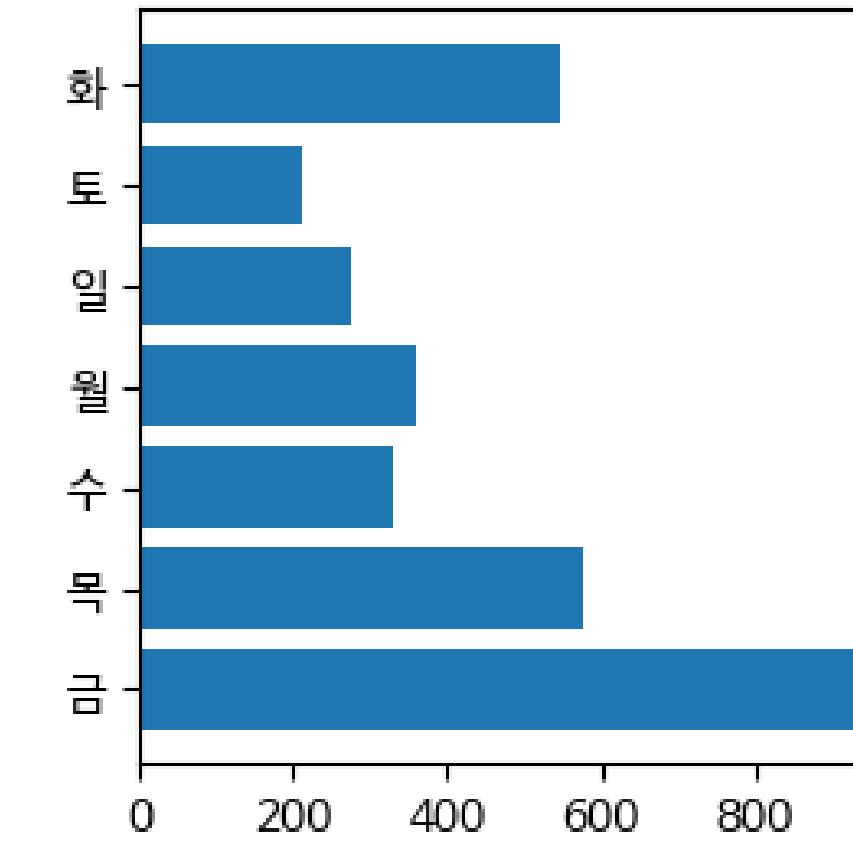
세로 막대그래프

```
plt.bar(df_week["요일"], df_week["청소년"])
```



가로 막대그래프

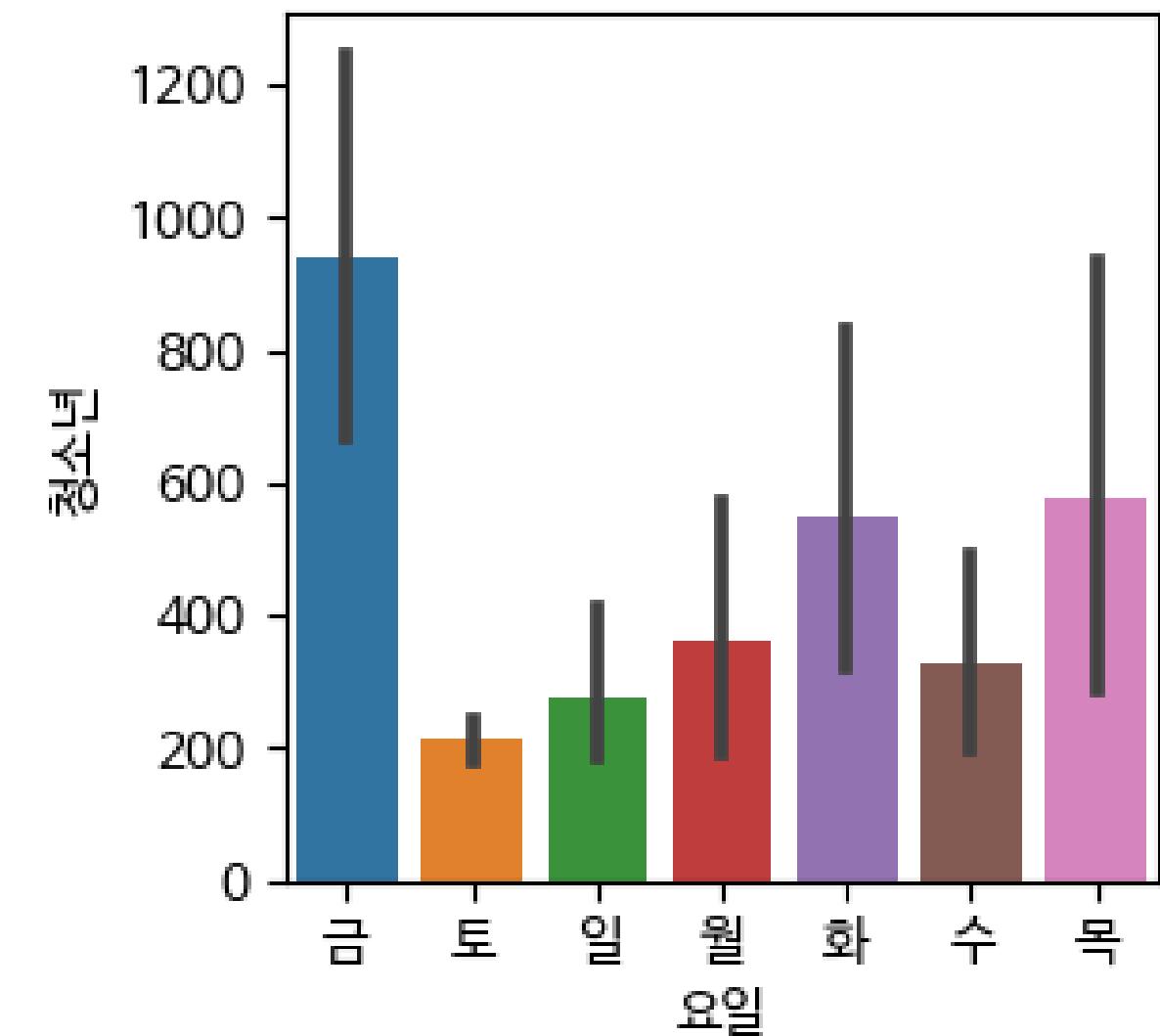
```
plt.barh(df_week["요일"], df_week["청소년"])
```



✓ Seaborn으로 막대 그래프 그리기

```
sns.barplot(data=df, x="요일", y="청소년")
# x, y축에 해당하는 칼럼의 이름을 전달
```

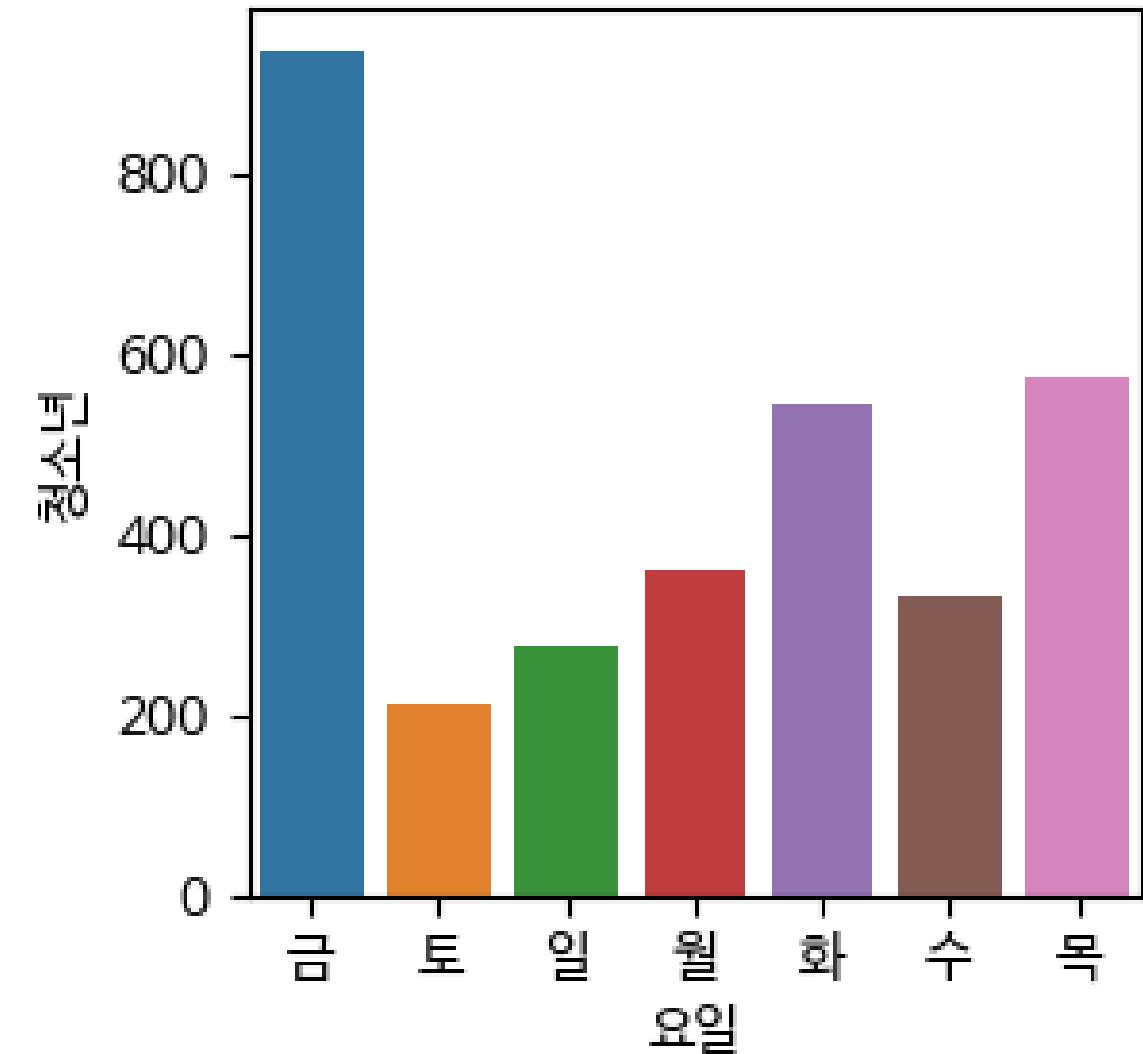
- barplot 함수를 사용
- data에 데이터 프레임을 전달
- x, y에 각각 칼럼이름을 전달
- 오차 막대의 기본 값은 95% 신뢰구간을 보여줌



✓ Seaborn으로 막대 그래프 그리기

```
sns.barplot(data=df, x="요일", y="청소년", errorbar=None)  
# errorbar=None을 전달하여 오차 막대를 제거함
```

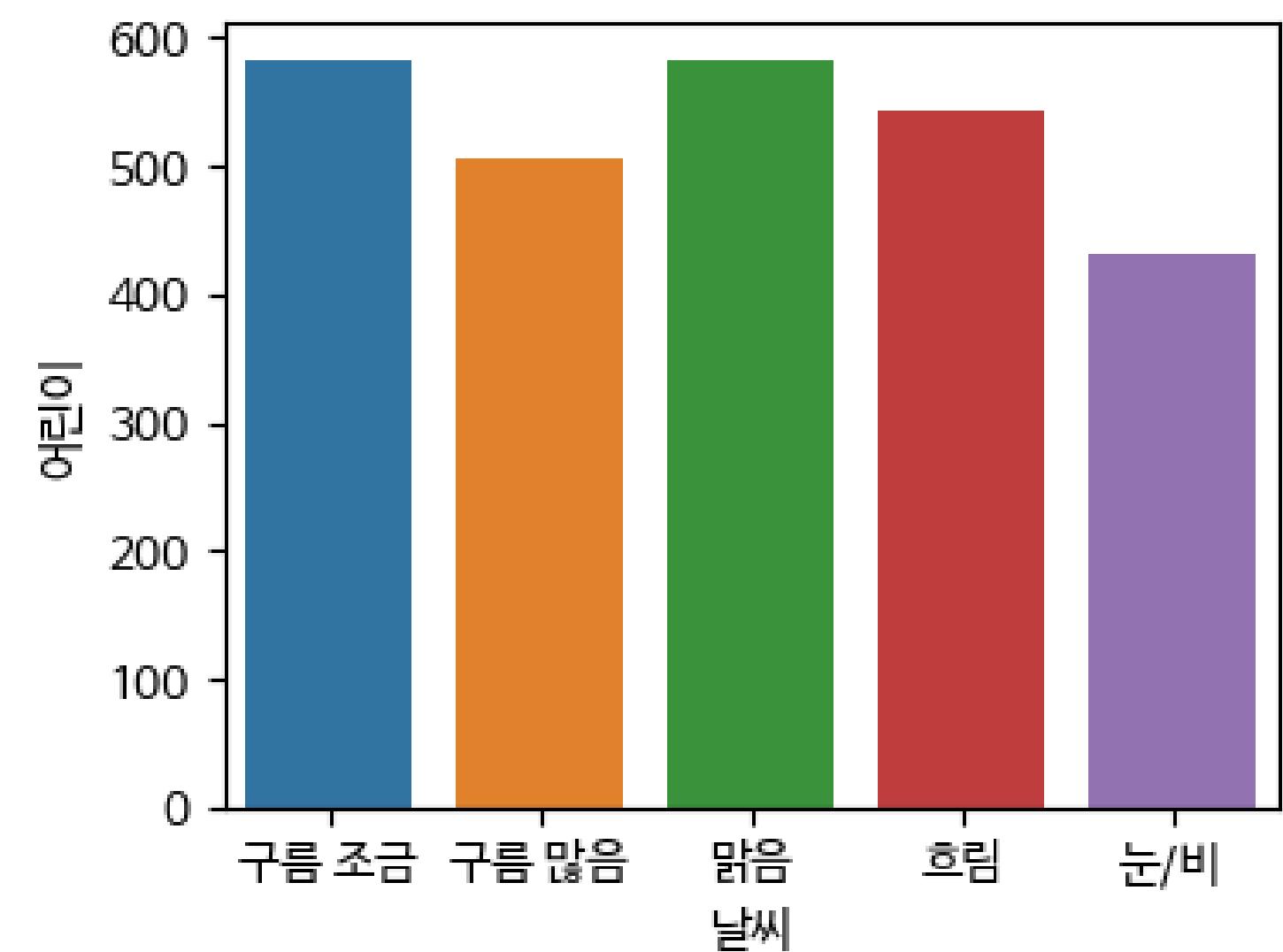
- errorbar에 None을 전달하여 오차 막대를 제거



✓ 그룹으로 묶어서 시각화하기

```
sns.barplot(data=df, x="날씨", y="어린이", errorbar=None)  
# 날씨별로 어린이 입장객의 수를 표현
```

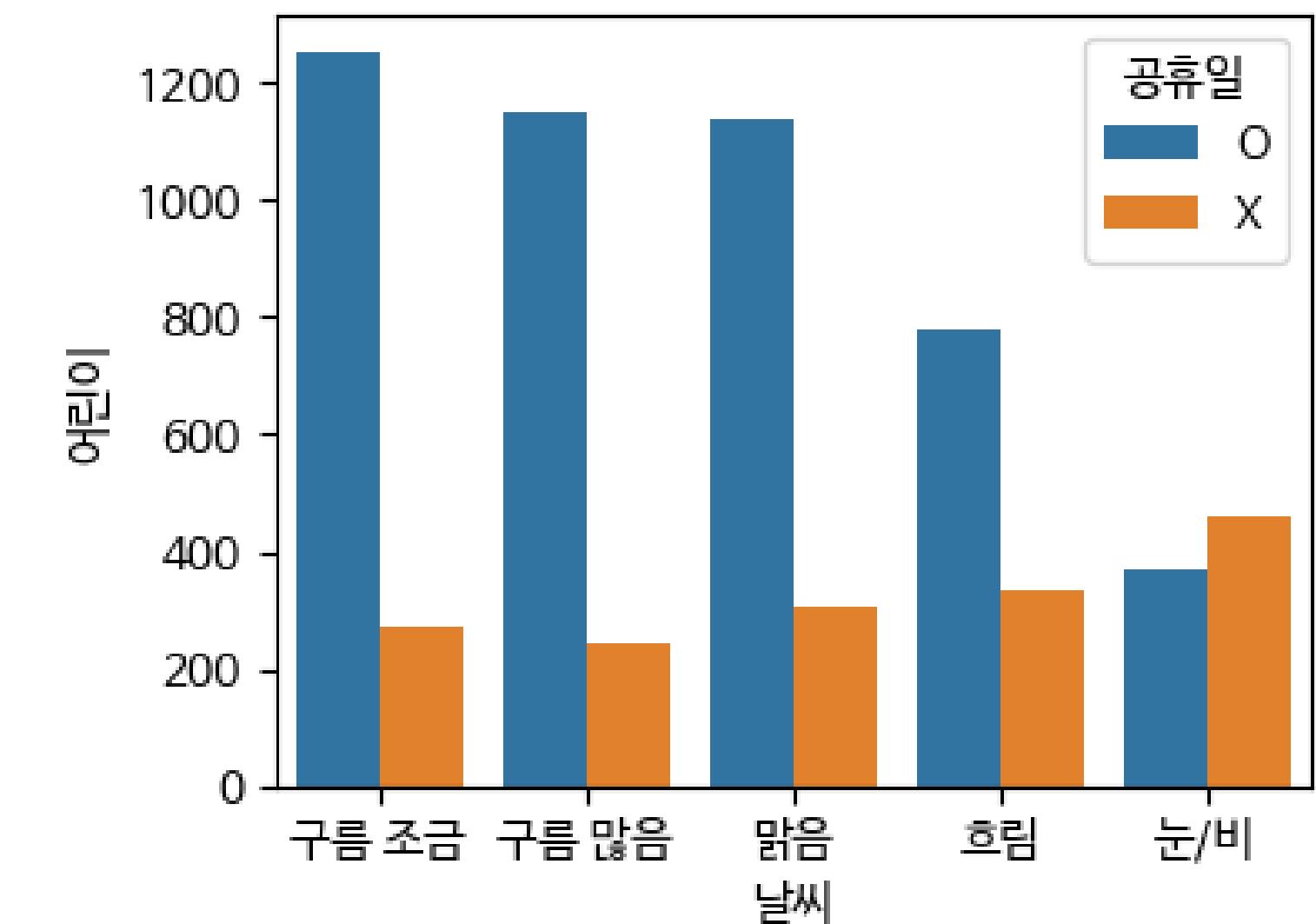
- 날씨별 어린이 입장객의 수를 시각화한 막대그래프
- 공휴일 여부에 따라 같은 날씨라도 결과가 다를 수 있음
- 그래프를 날씨를 기준으로 그룹을 만들고 출력해야 함



⑤ 그룹으로 묶어서 시각화하기

```
sns.barplot(data=df, x="날씨", y="어린이", errorbar=None, hue="공휴일")  
# "공휴일" 값을 기준으로 그룹을 나눔
```

- hue 매개변수에 그룹을 나눌 기준의 이름을 전달
- 공휴일 여부에 따라 달라지는 것을 시각화할 수 있음

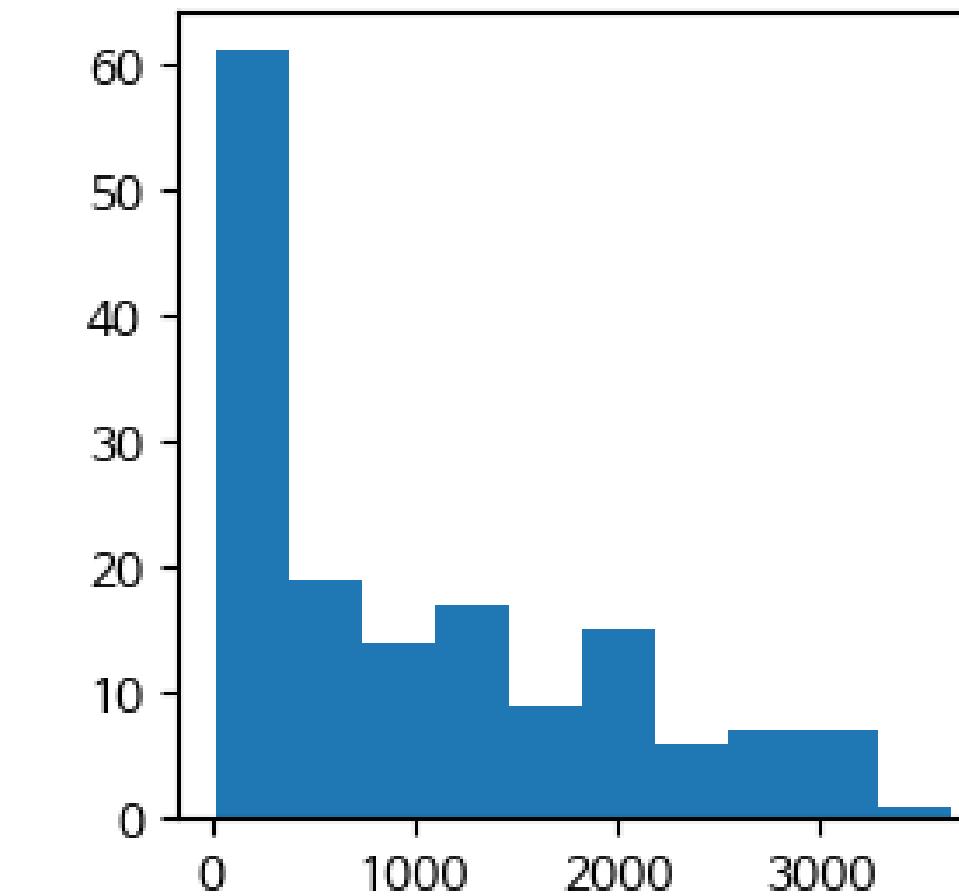
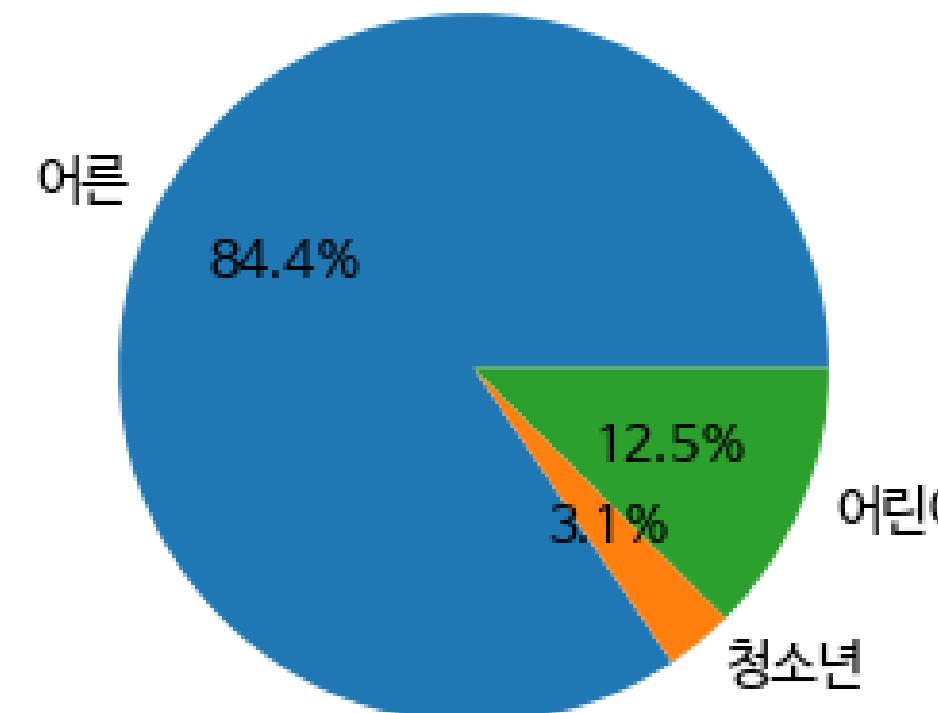


04

파이 차트와 히스토그램

⑤ 파이 차트와 히스토그램

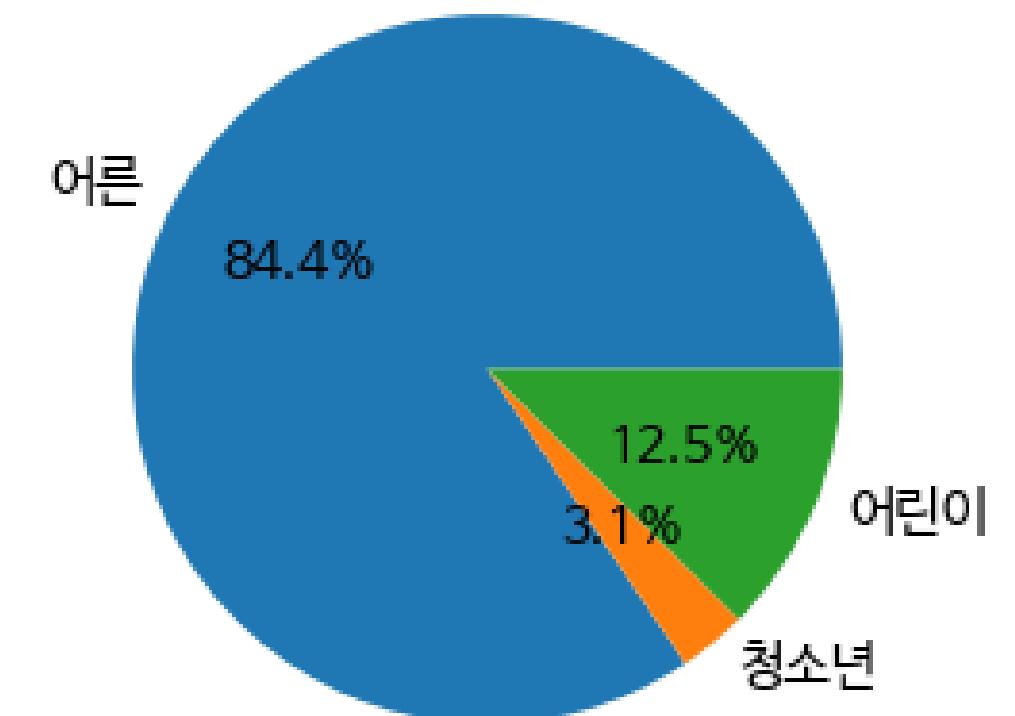
데이터의 분포나 비율을 시각화하는 그래프



⑤ 파이 차트

전체 데이터에서 그 범주에 해당하는 비율을 표현

- 원형으로 표현되며, 부채꼴의 중심각은 해당 범주의 상대적인 비율을 표현
- matplotlib의 pie 함수를 사용(seaborn에는 별도로 없음)



⑤ 데이터 불러오기

```
data_2019 = df[df["연"] == 2019][["어른", "청소년", "어린이"]].sum()
```

1. df에서 “연”이 2019인 데이터들만 가져옴
2. 가져온 데이터에서 어른, 청소년, 어린이 칼럼을 가져옴
3. 가져온 데이터를 각각 sum하여 합계를 구함
4. 결과로 나온 시리즈를 data_2019에 저장

어른	103125.0
청소년	3817.0
어린이	15209.0
dtype:	float64

⑤ 레이블과 데이터 나누기

```
labels = data_2019.index.tolist()
# ['어른', '청소년', '어린이']
data = data_2019.tolist()
# [103125.0, 3817.0, 15209.0]
```

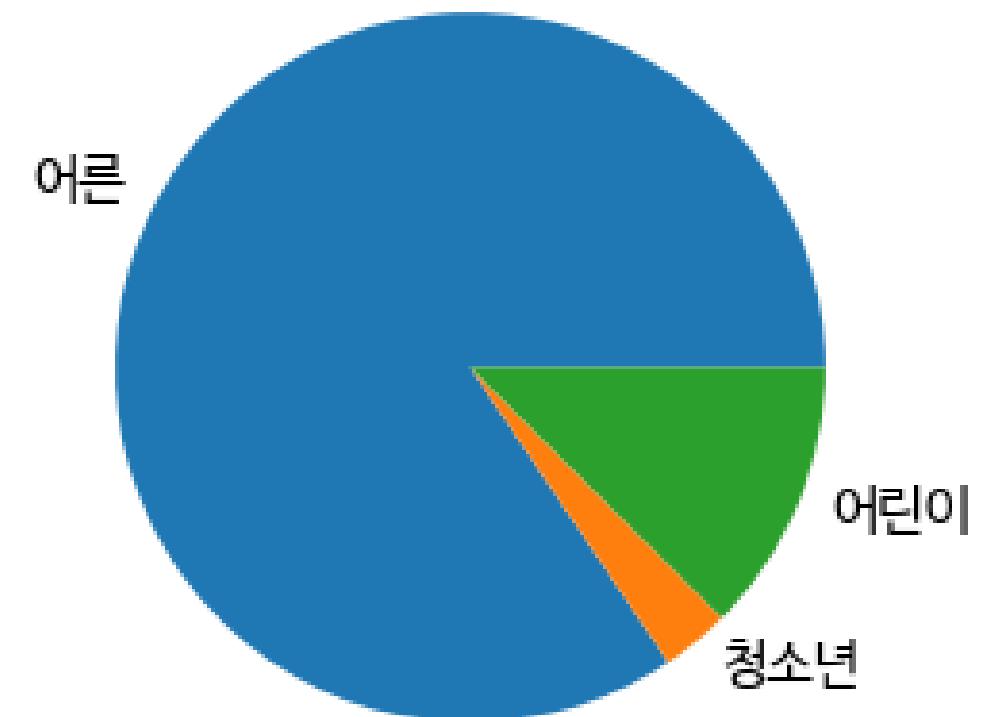
- 파이 차트를 그리기 위해 각각의 비율과 레이블을 가져옴

어른	103125.0
청소년	3817.0
어린이	15209.0
dtype:	float64

⑤ 파이 차트 그리기

```
plt.pie(x=data, labels=labels)
```

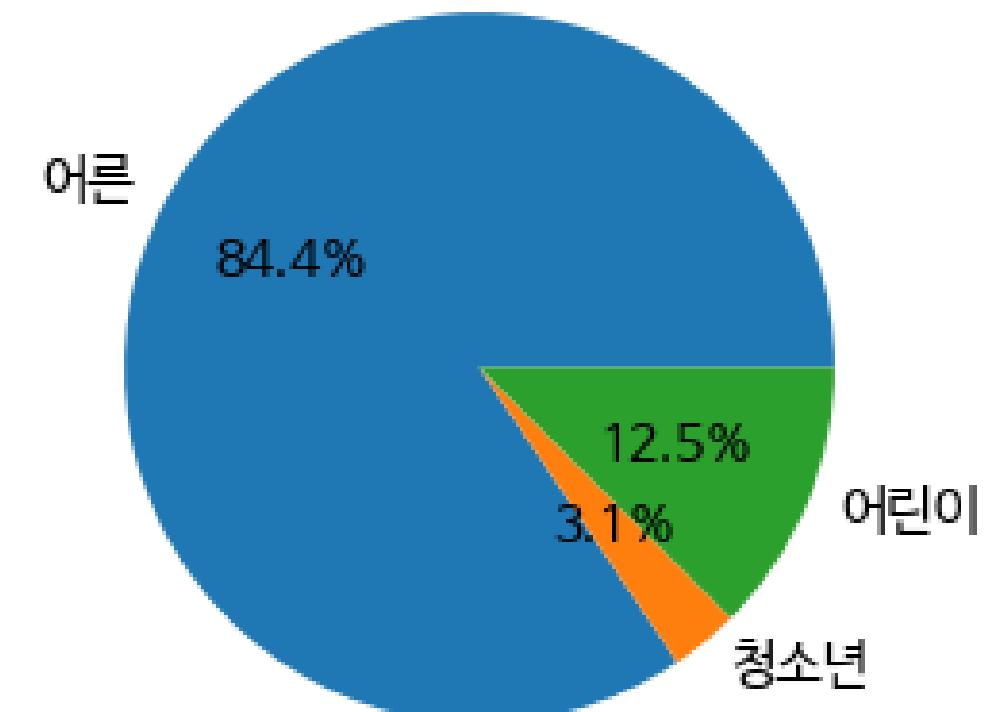
- pie 함수를 이용해 파이 차트를 그림
- x에 각 범주의 비율을 수치로 전달
- labels에 각 값이 이름을 전달



✓ 비율 표시하기

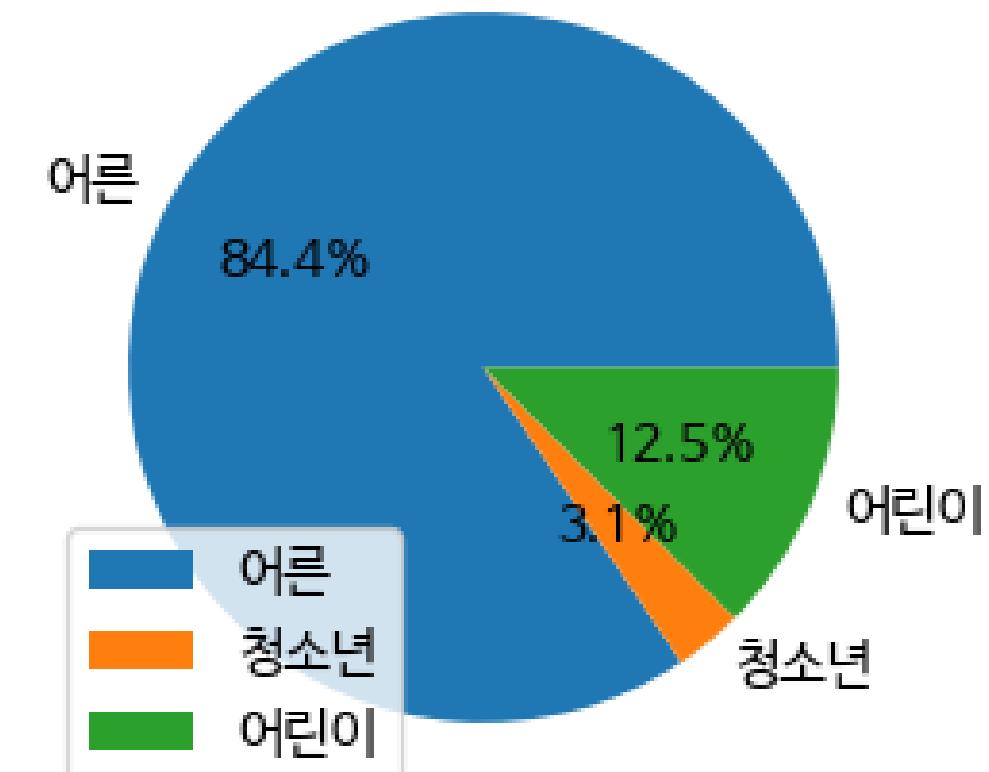
```
plt.pie(x=data, labels=labels, autopct="%1.1f%%")
```

- 구체적인 수치를 알 수 있도록 적어주는 옵션
- autopct에 포맷 문자열을 전달하여 출력
- 위 코드는 소수점 첫째 자리까지 보여주고 뒤에 %를 붙이는 방법



✓ 범례 추가하기

```
plt.pie(x=data, labels=labels, autopct="%1.1f%%")  
plt.legend(loc='lower left')
```

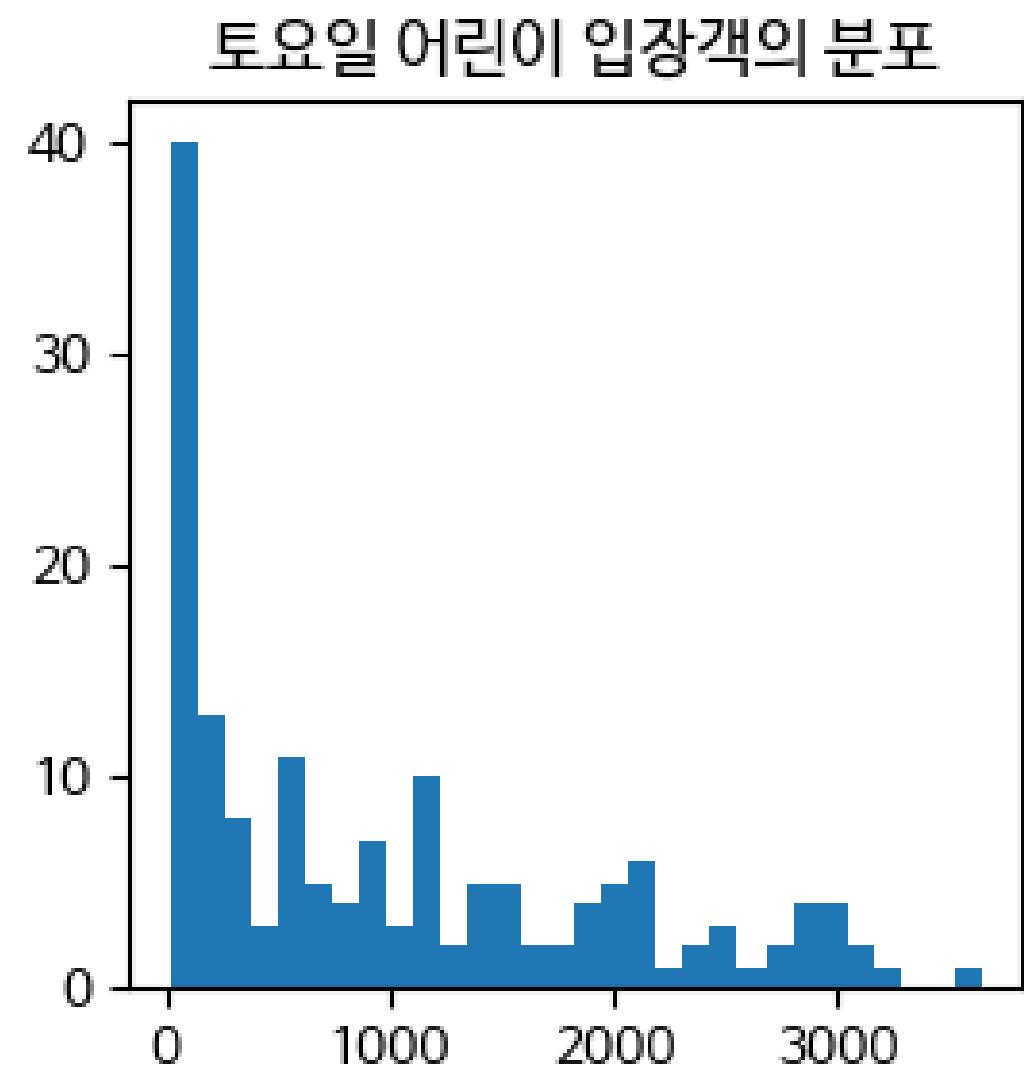


- legend함수를 통해 범례를 표시함
- loc 매개변수를 통해 범례의 위치를 조절할 수 있음
- lower left: 왼쪽 아래에 범례를 표시

✓ 히스토그램

특정 구간에 해당하는 빈도를 표현하는 그래프

- 이 구간에 속한 데이터는 몇 개가 있는지를 표현
- 특정 값의 빈도 분포를 분석할 때 사용함



토요일 데이터 가져오기

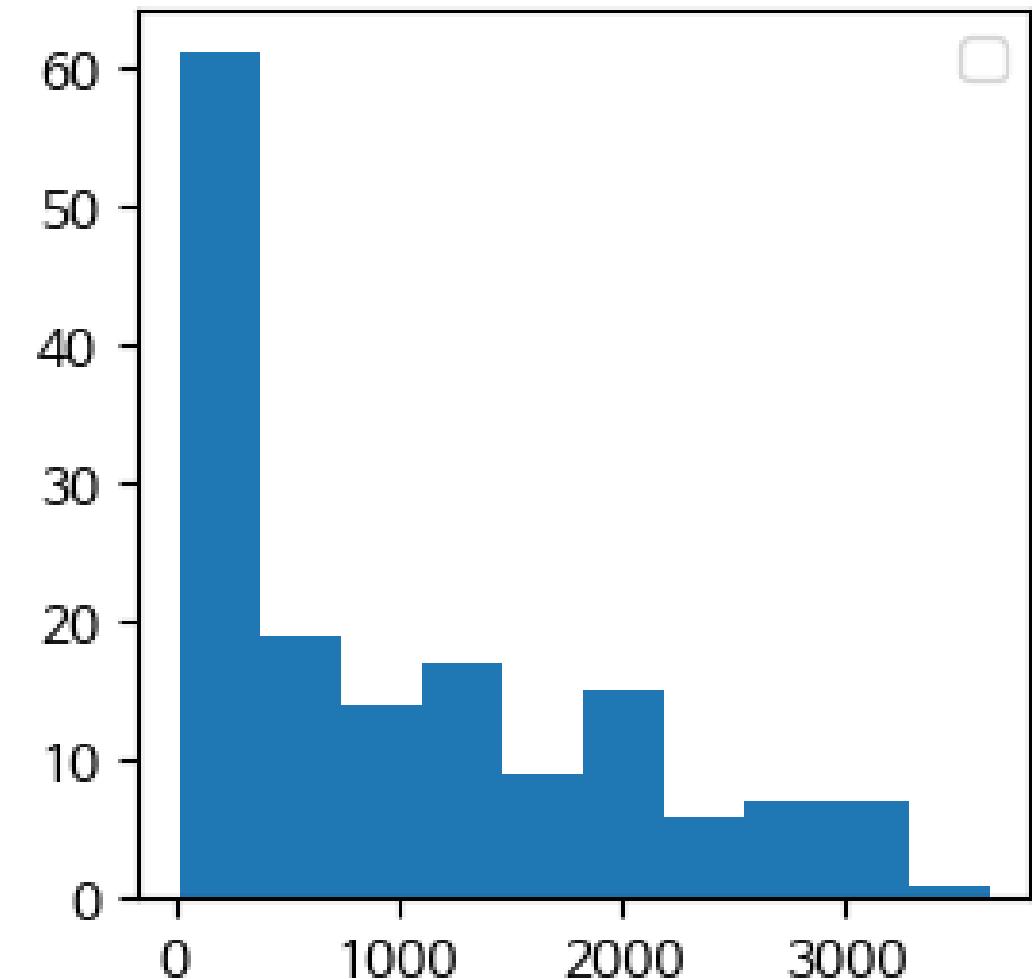
```
df_sat = df[df["요일"] == "토"]
# 토요일 데이터를 df_sat에 저장
```

- “요일”이 “토”인 데이터만 가져와 df_sat에 저장

✓ 히스토그램 그리기

```
plt.hist(df_sat["어린이"], bins=10)  
# bins: 몇개의 구간으로 나눌 것인가 -> 막대의 수
```

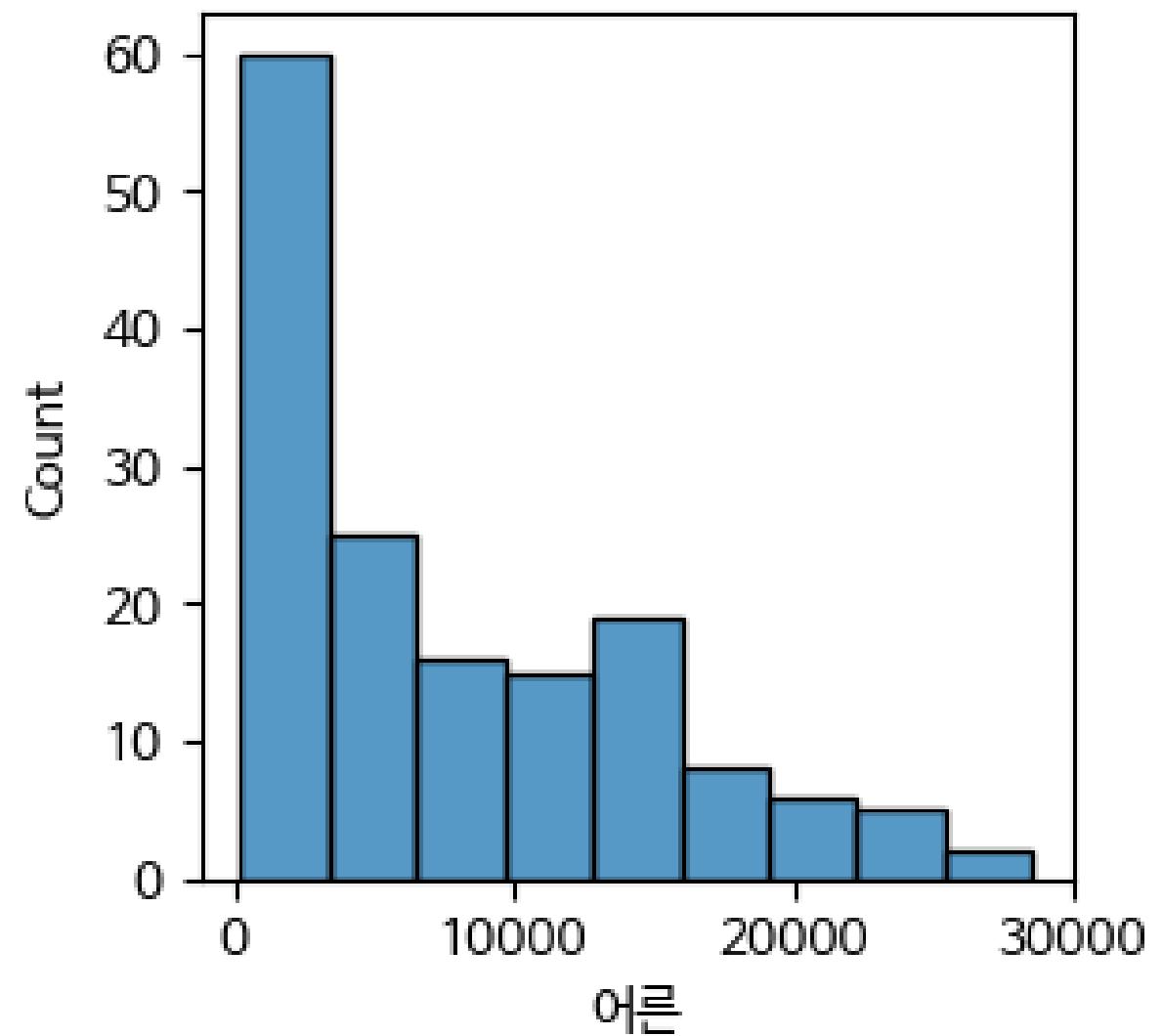
- hist 함수를 통해 히스토그램을 그림
- 값을 연속된 형태로 넣으면 그 값의 분포를 그림
- bins 매개변수에 구간의 수를 전달하여 몇 개로 나눌지 설정할 수 있음



✓ Seaborn으로 히스토그램 그리기

```
sns.histplot(data=df_sat, x="어른")
```

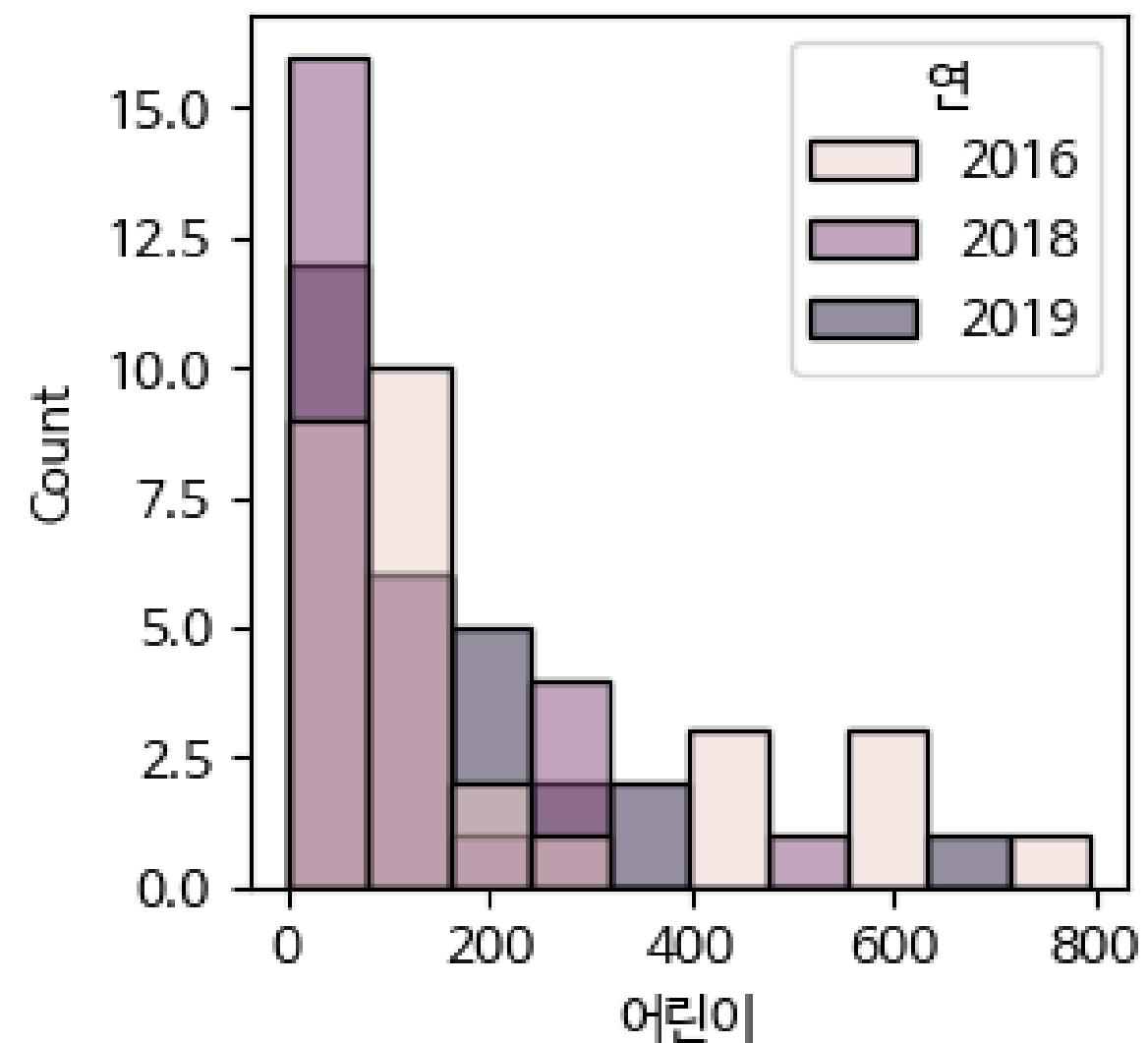
- data에 시각화할 데이터 프레임을 전달
- x에 시각화할 칼럼의 이름을 전달



✓ 여러 개의 히스토그램 그리기

```
df_feb = df[df["월"] == 2]
# 2월의 데이터만 가져옴
sns.histplot(data=df_feb, x="어린이", hue="연")
# 어린이 입장객의 분포를 그림
# 연도별로 그룹을 나눔
```

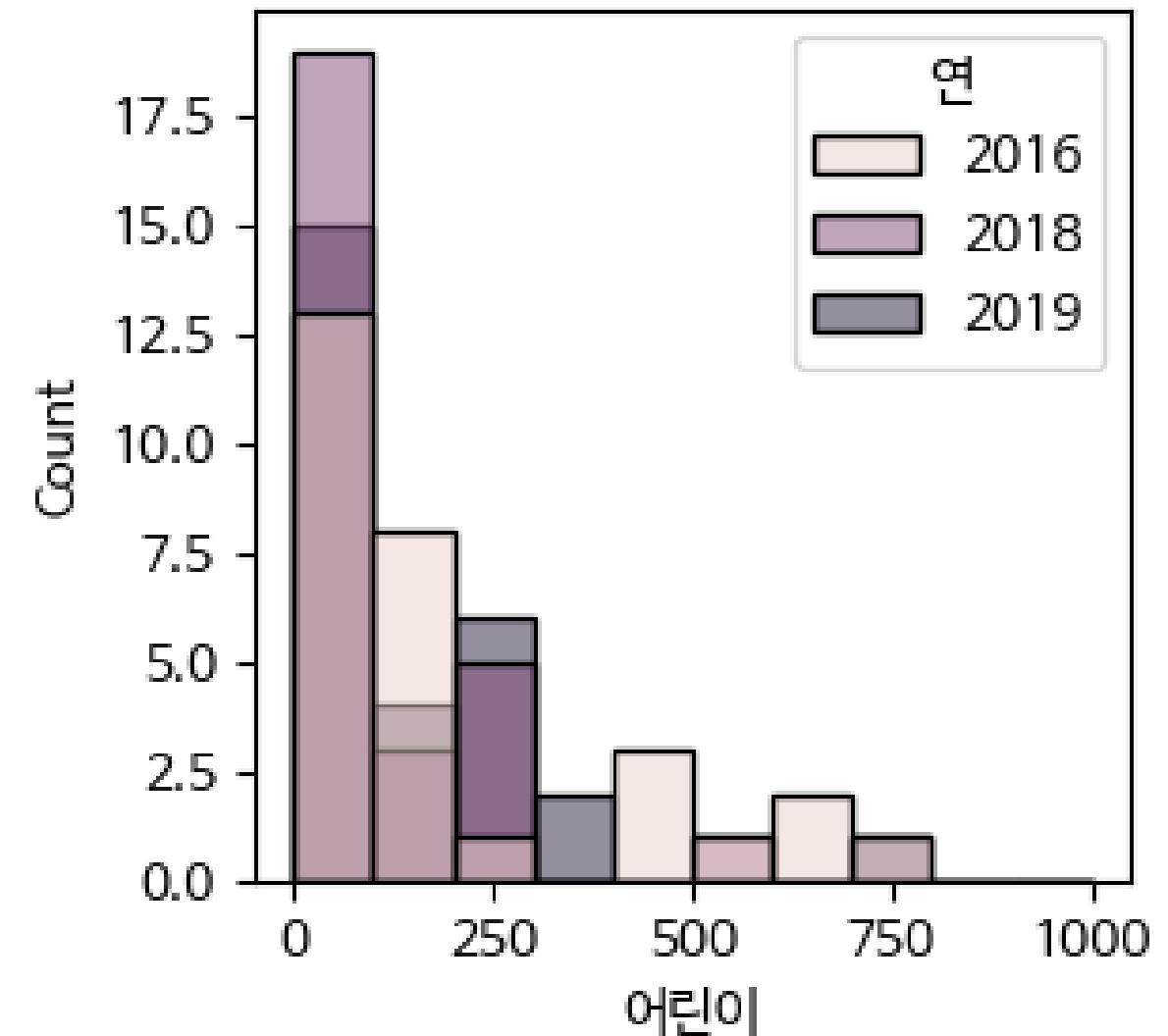
- seaborn의 그래프는 hue를 통해 그룹을 나눌 수 있음
- 연도별 분포를 보여주는 그래프를 그릴 수 있음



✓ 구간의 범위 설정하기

```
sns.histplot(data=df_feb, x="어린이", hue="연", binrange=(0, 1000), bins=10)
```

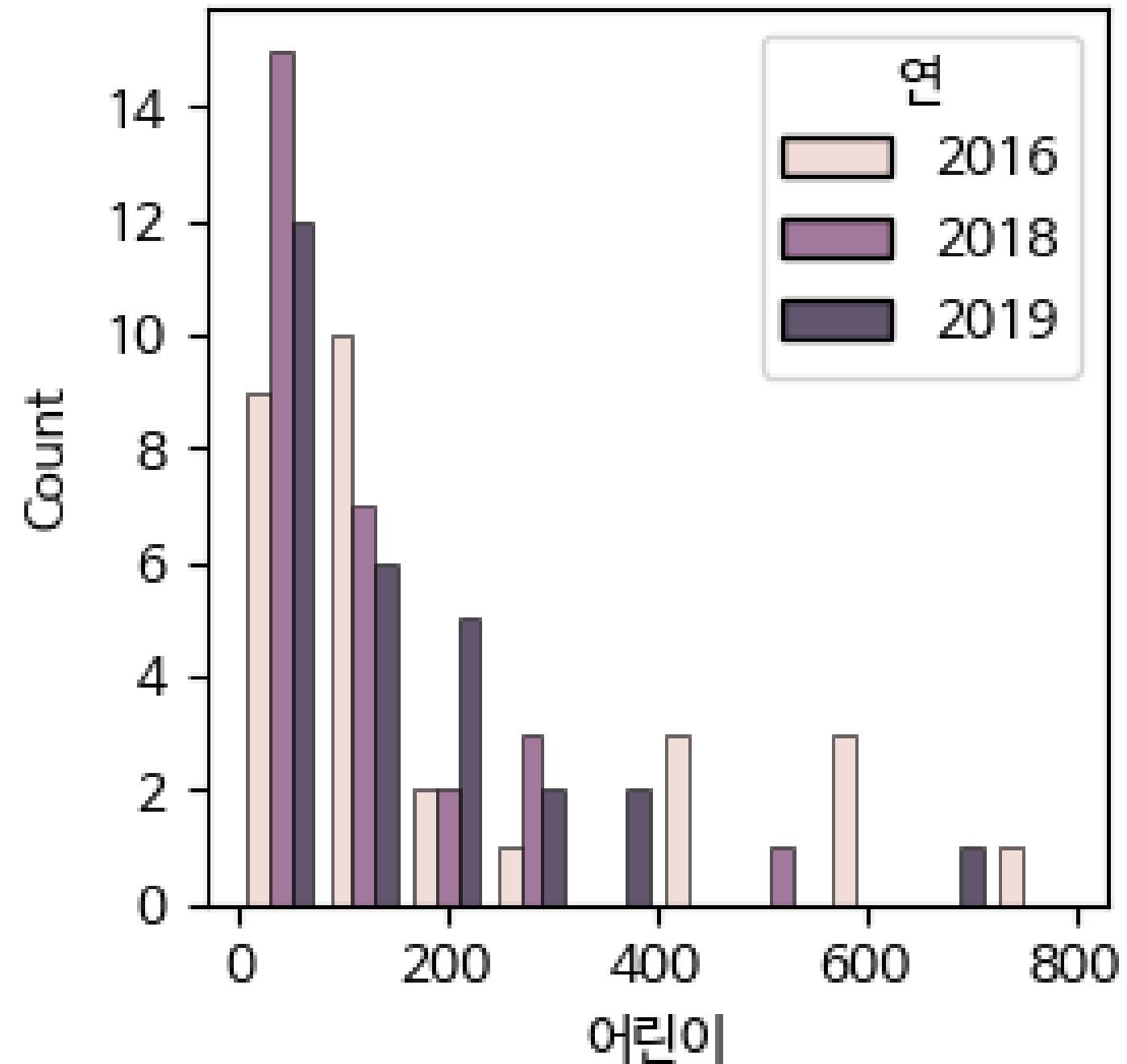
- binrange에 전체 구간의 범위를 전달할 수 있음
- binrange를 전달하지 않으면 데이터의 최소, 최대로 설정됨



✓ 겹치는 그래프 분리하기

```
sns.histplot(  
    data=df_feb,  
    x="어린이",  
    hue="연",  
    binrange=(0, 800),  
    bins=10,  
    multiple="dodge",  
    shrink=0.8,  
)
```

- multiple: 여러 그래프는 어떻게 그릴 것인지 (dodge: 겹치지 않게)
- shrink: 막대의 너비를 조절 (0.8: 원래 너비의 80%)



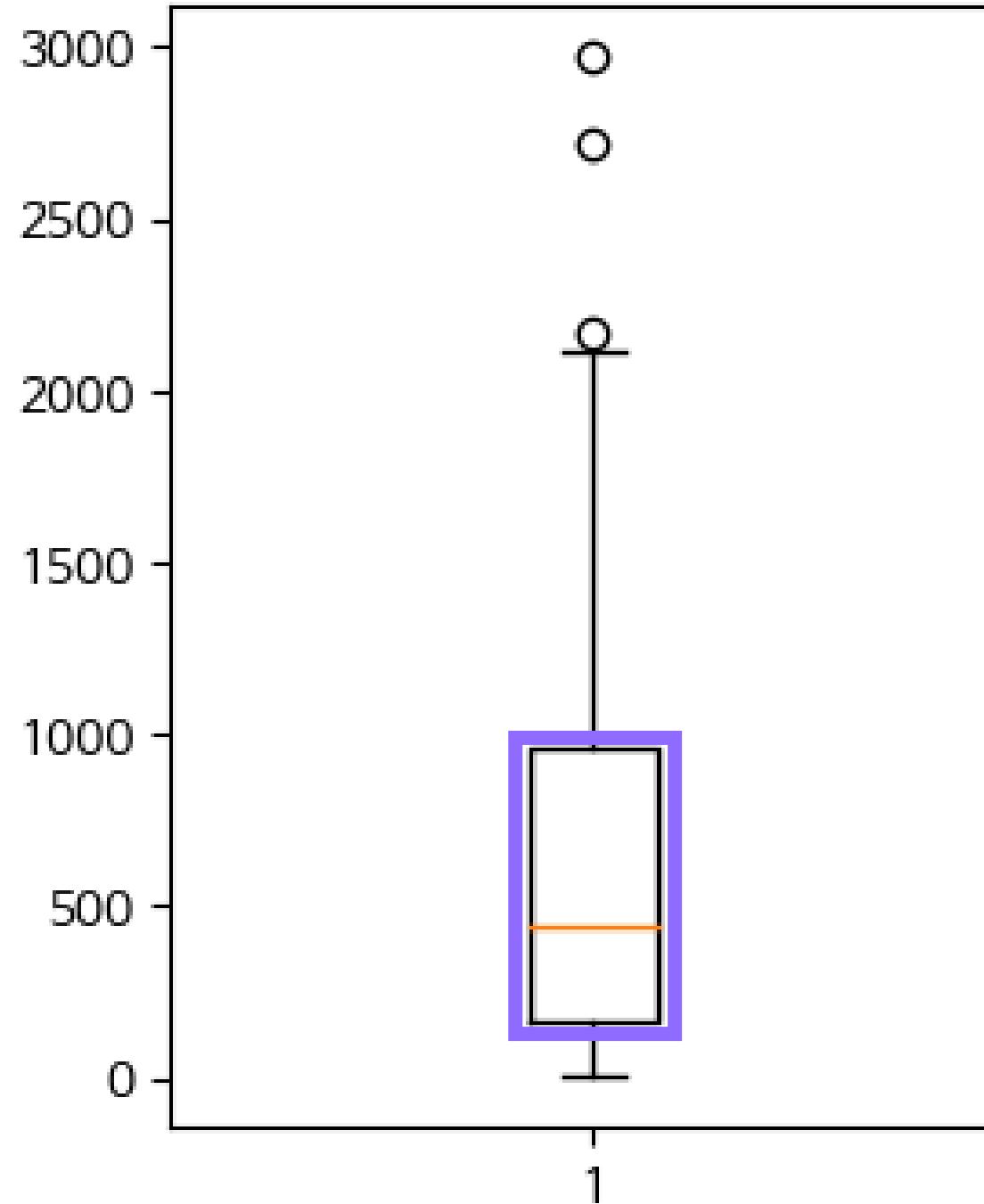
05

박스 그래프

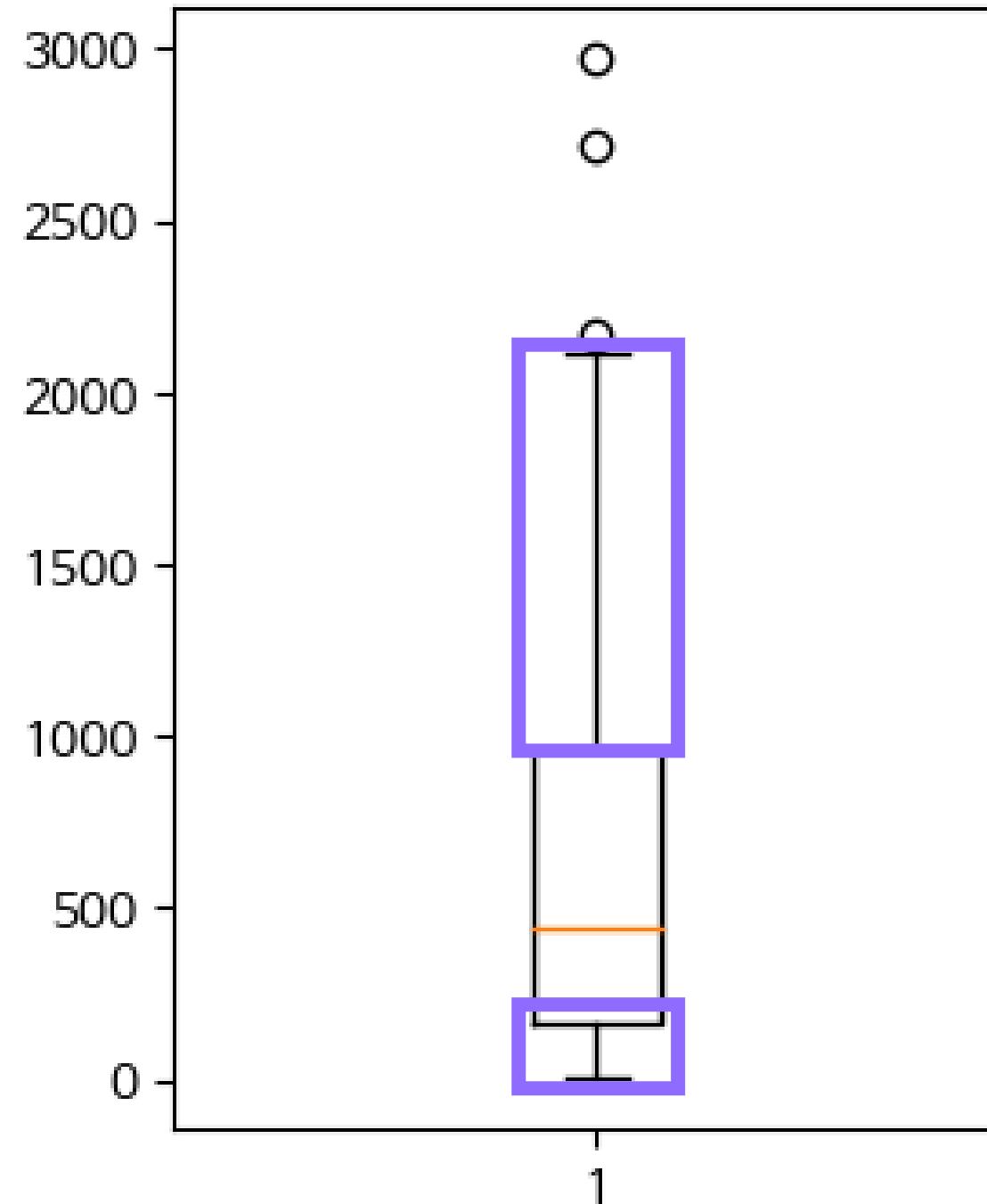
⑤ 박스 그래프

데이터의 분포를 보여주는 시각화 그래프

- 수치형 변수의 데이터를 요약하고 비교하는데 사용
- 단순하게 평균을 보면 알 수 없는 데이터의 중앙값과 이상치, 분포를 파악할 수 있음

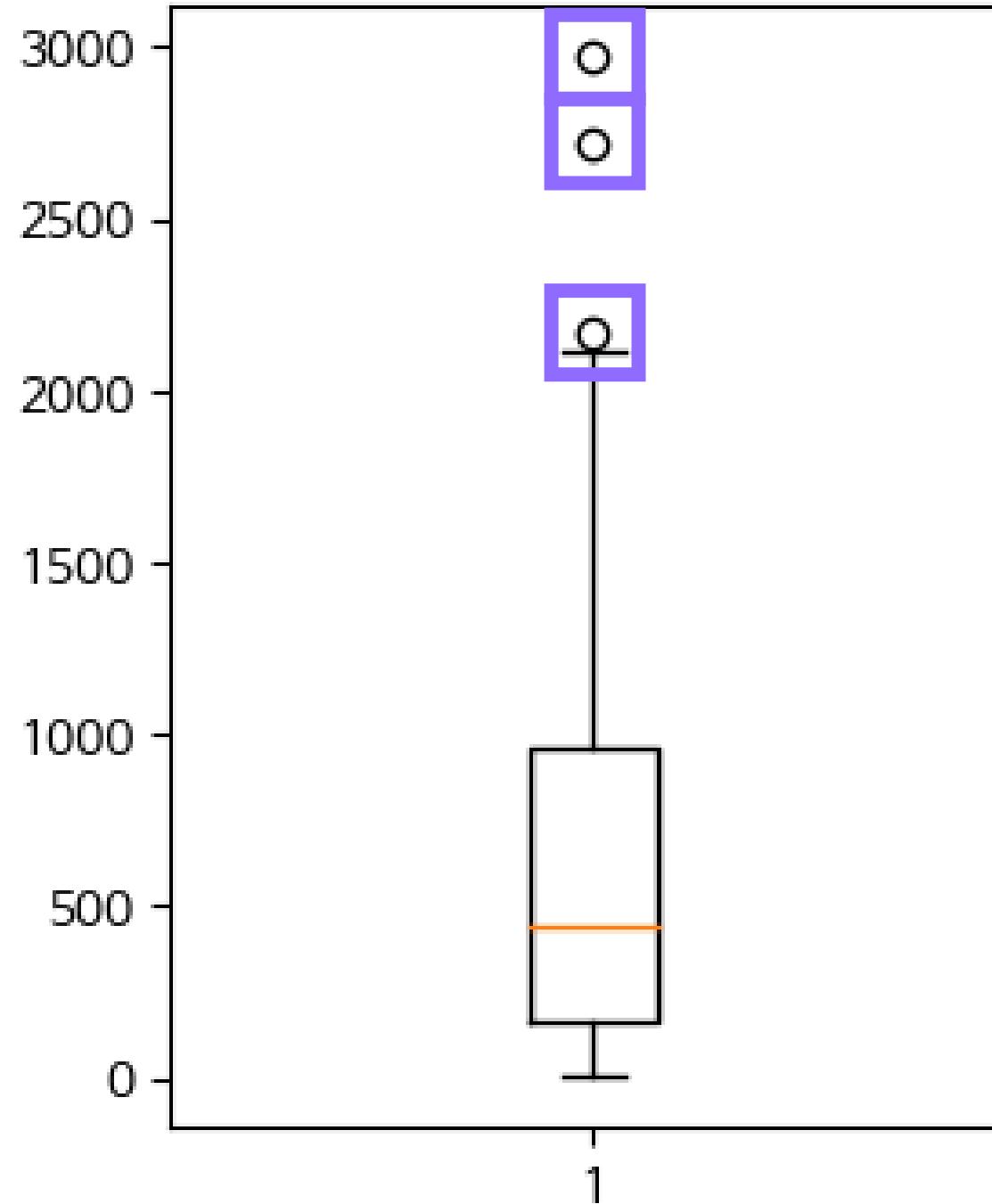
 박스(Box)

- 데이터의 중간값(Median)과 사분위수(Q1, Q3)를 나타냄.
- 가운데 그려진 선은 중앙값에 해당
- 박스의 상단과 하단은 각각 상위 25%와 하위 25%를 나타냄
- 박스의 높이는 IQR이라고 부름

 수염 (Whisker)

- 박스의 범위를 벗어난 데이터
- 일반적으로 Q1에서부터 $1.5 * \text{IQR}$ 만큼의 범위까지를 나타냄

✓ 이상치



- 수염의 범위를 벗어난 값
- 극단적인 값이거나 잘못된 측정으로 인해 발생한 값

⑤ 데이터 준비하기

```
df_2018_apr = df[(df["연"] == 2018) & (df["월"] == 4)]
```

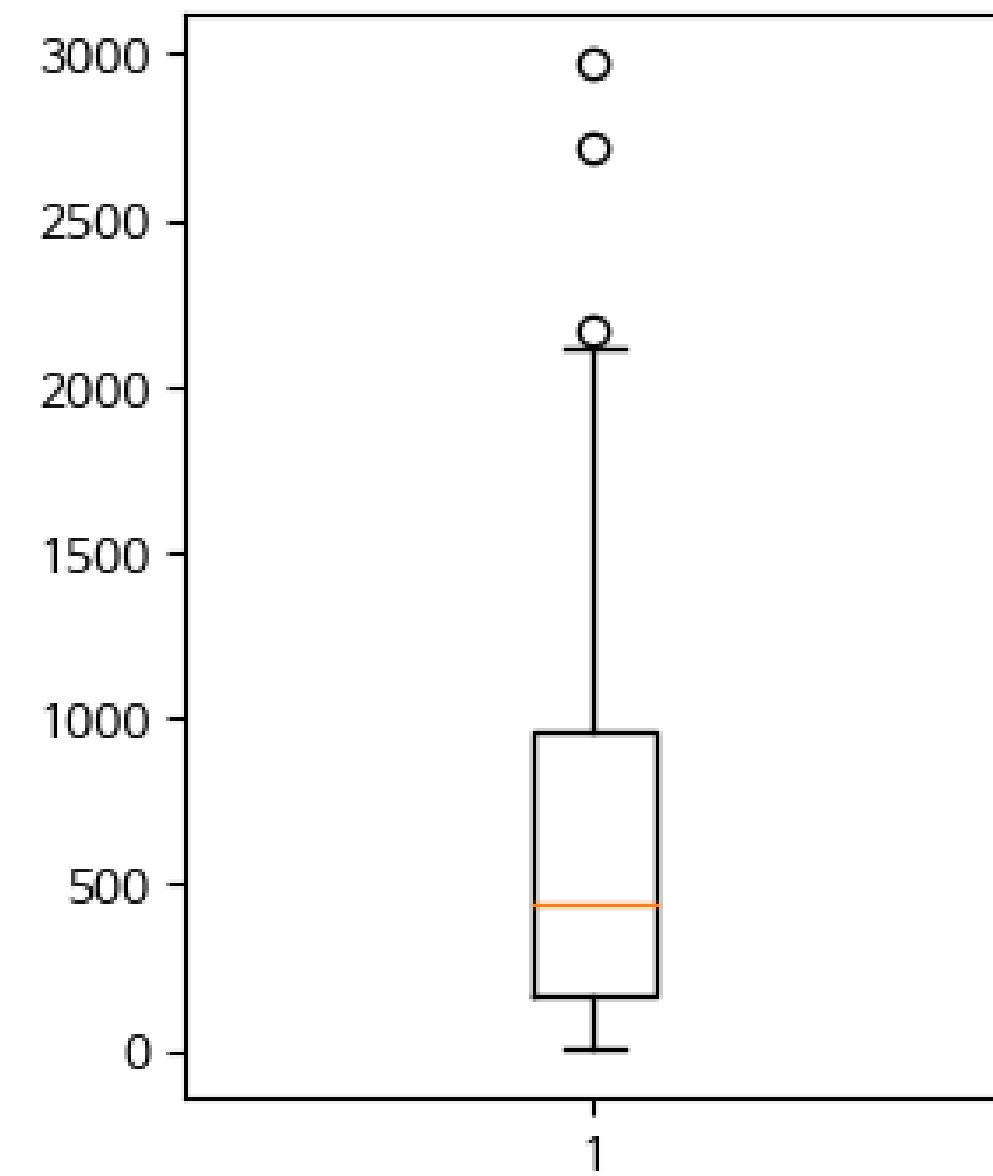
- df에서 "연" 열이 2018이고 "월" 열이 4인 조건을 만족하는 데이터를 선택
- df_2018_apr라는 새로운 데이터프레임을 저장

날짜	요일	공휴일	날씨	어른	청소년	어린이	외국인
2018-04-01	일	O	흐림	9247	197.0	1455	156
2018-04-02	월	X	Nan	1697	13.0	108	12
2018-04-03	화	X	Nan	1861	103.0	176	67
2018-04-04	수	X	Nan	1453	47.0	85	32
2018-04-05	목	X	Nan	145	408.0	6	7

✓ matplotlib로 그리기

```
plt.boxplot(df_2018_apr["어린이"])
```

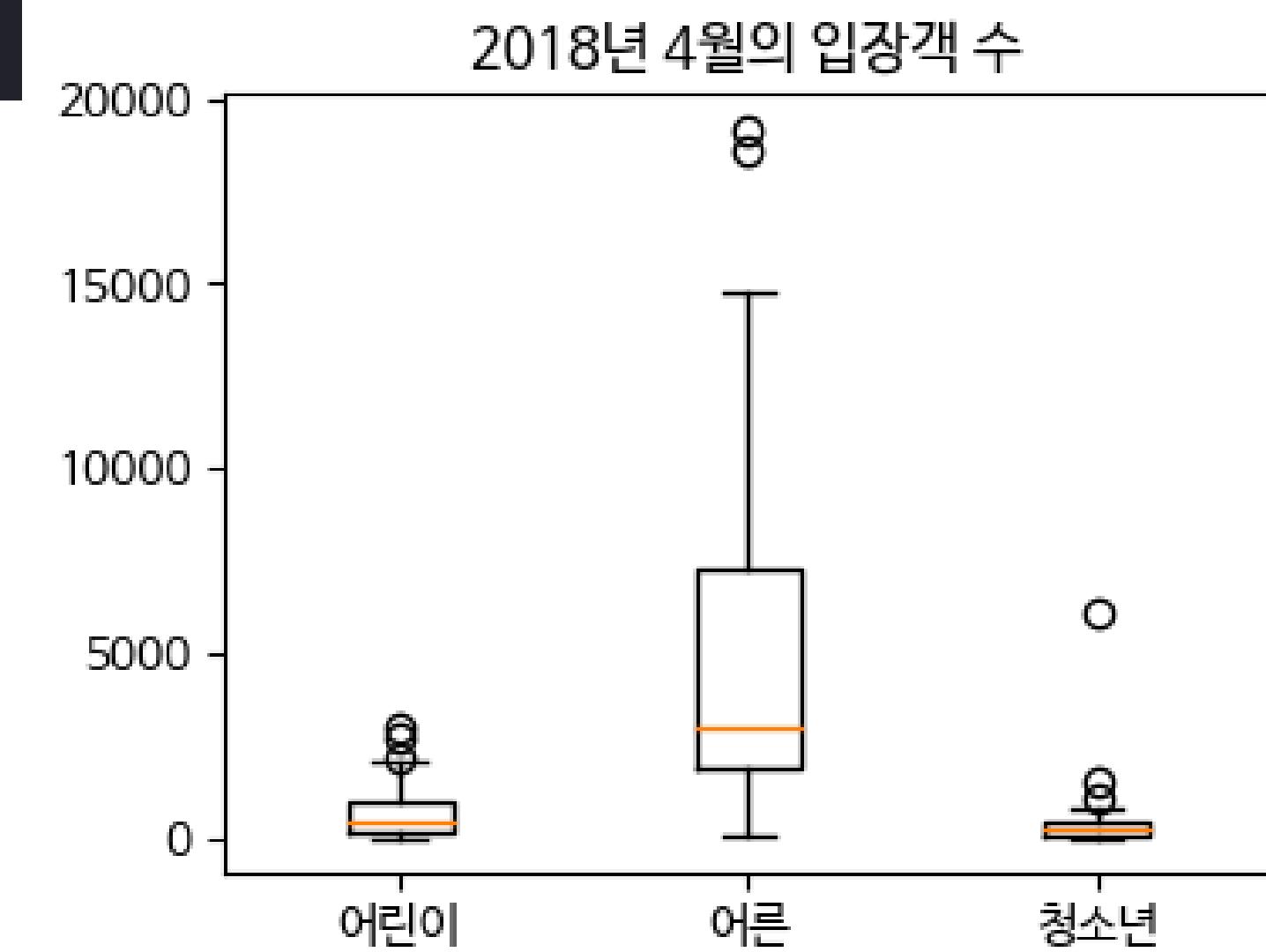
- 데이터프레임의 "어린이" 열을 선택하여 박스 그래프를 그림



✓ 여러 개 그리기

```
plt.boxplot([df_2018_apr["어린이"], df_2018_apr["어른"], df_2018_apr["청소년"]])  
plt.xticks([1, 2, 3], labels=["어린이", "어른", "청소년"])  
plt.title("2018년 4월의 입장객 수")
```

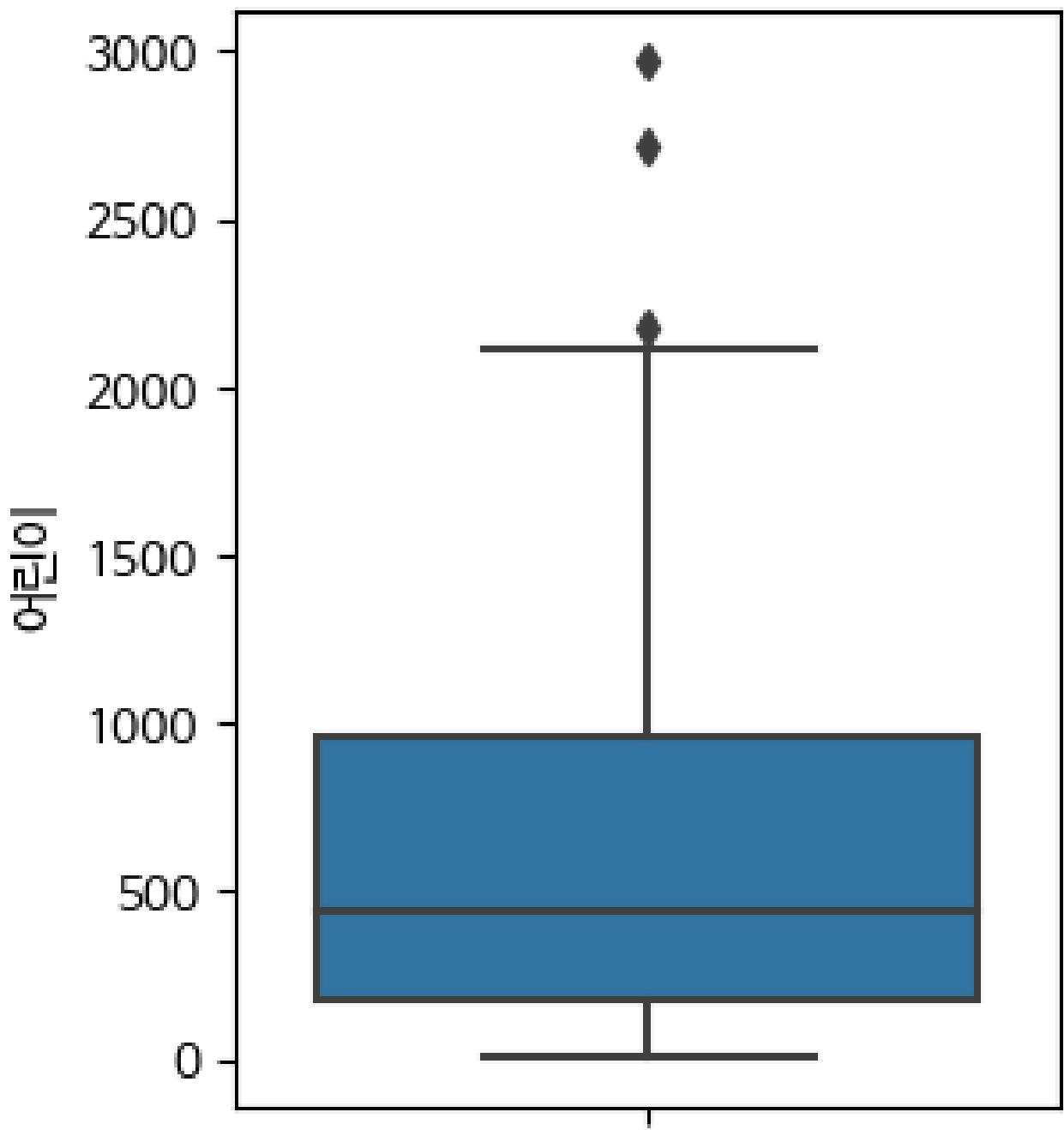
- "어린이", "어른", "청소년" 세 가지 열에 대한 데이터를 각각 박스 그래프로 그림



 seaborn으로 그리기

```
sns.boxplot(data=df_2018_apr, y="어린이")
```

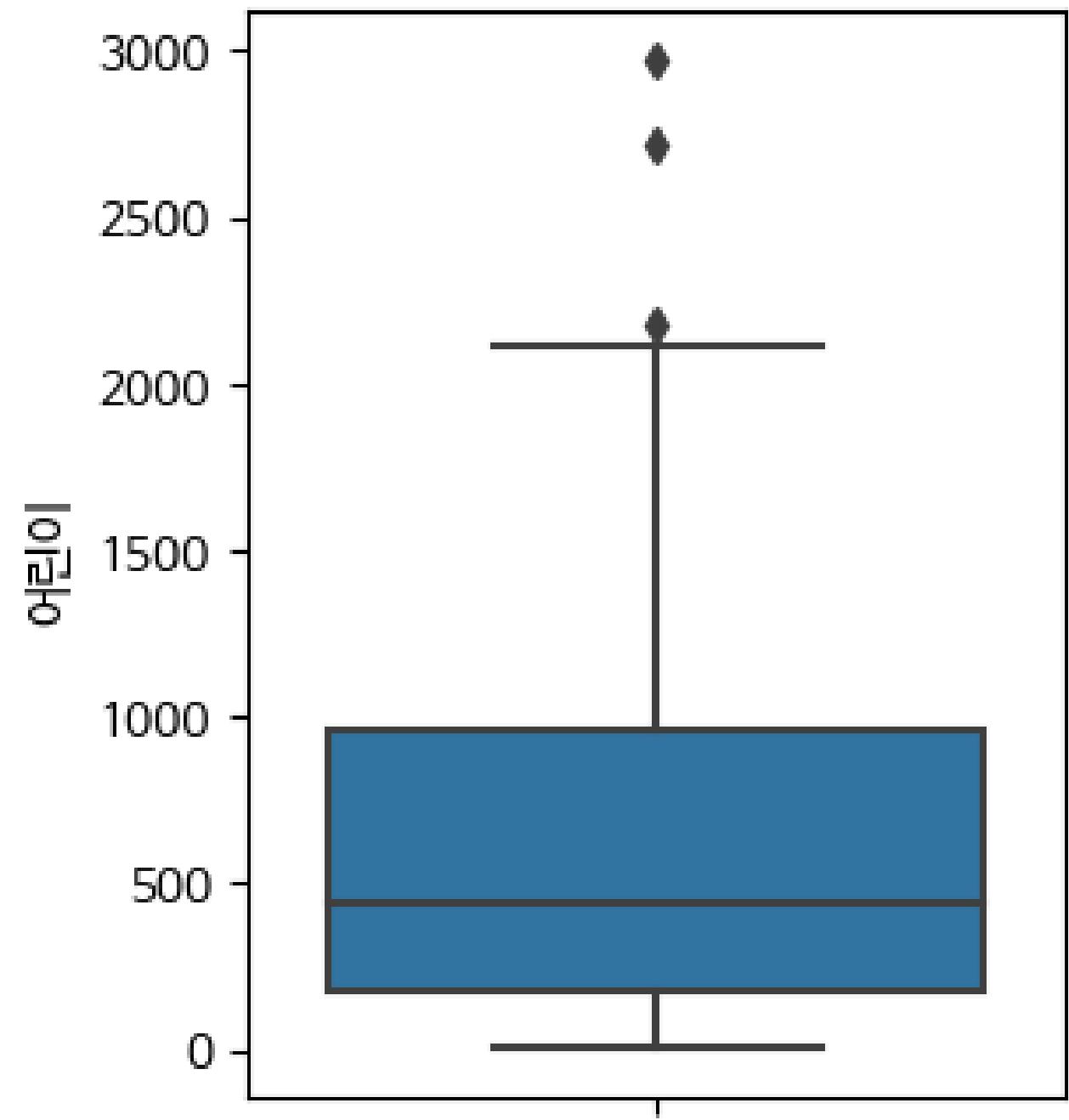
- data에 데이터 프레임을 전달
- y에 표시할 칼럼이름을 전달



⑤ 2018년 데이터 가져오기

```
df_2018 = df[df["연"] == 2018]
```

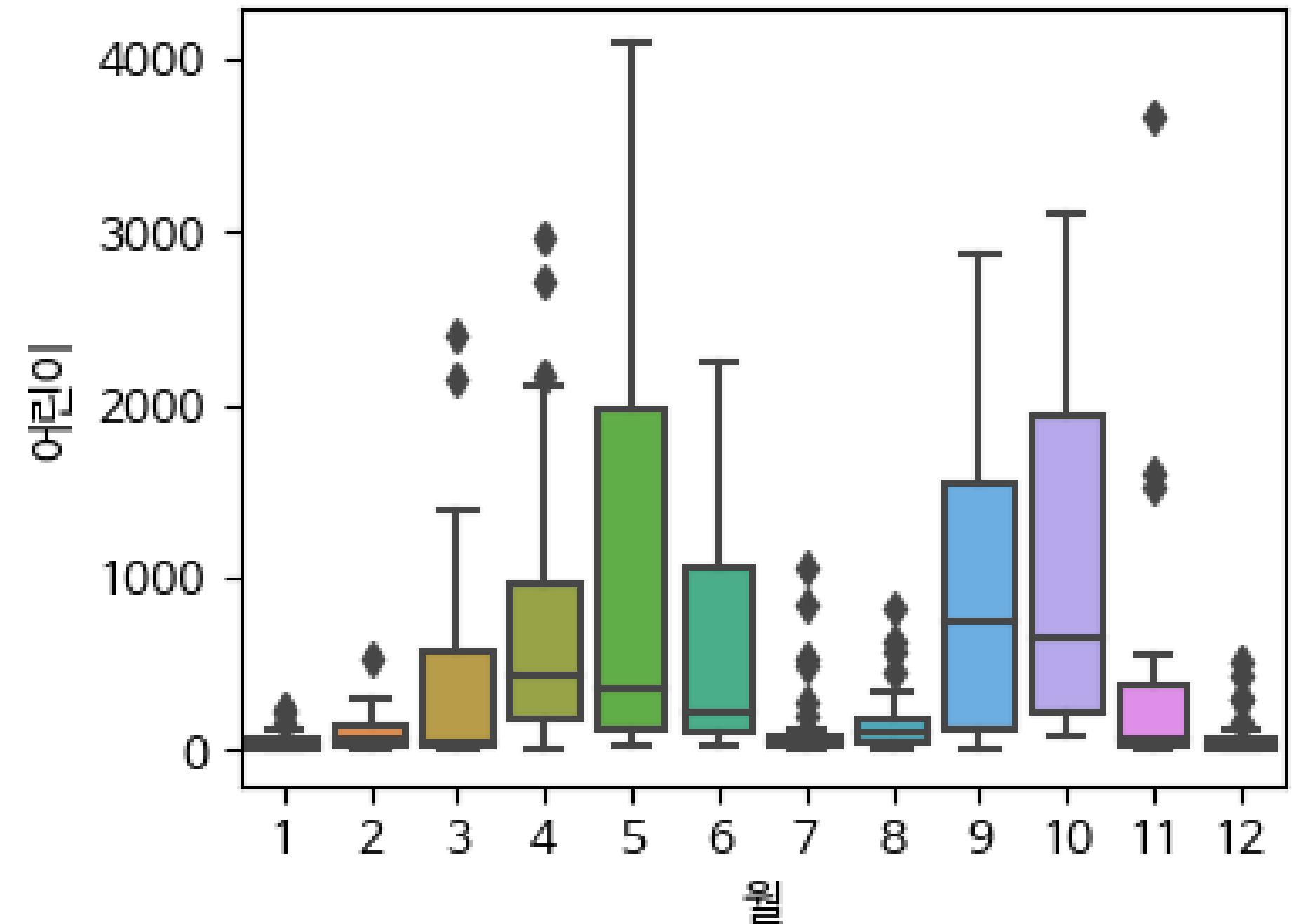
- df에서 "연" 열이 2018인 조건을 만족하는 데이터를 선택
- df_2018라는 새로운 데이터프레임을 저장



✓ 2018년 데이터 가져오기

```
sns.boxplot(data=df_2018, x="월", y="어린이")
```

- x축은 월별 데이터를 표시
- y축은 어린이 데이터를 표시
- 월별 어린이 입장객의 추세와 분포를 표시



06

산점도와 히트맵

⑤ 산점도 그래프

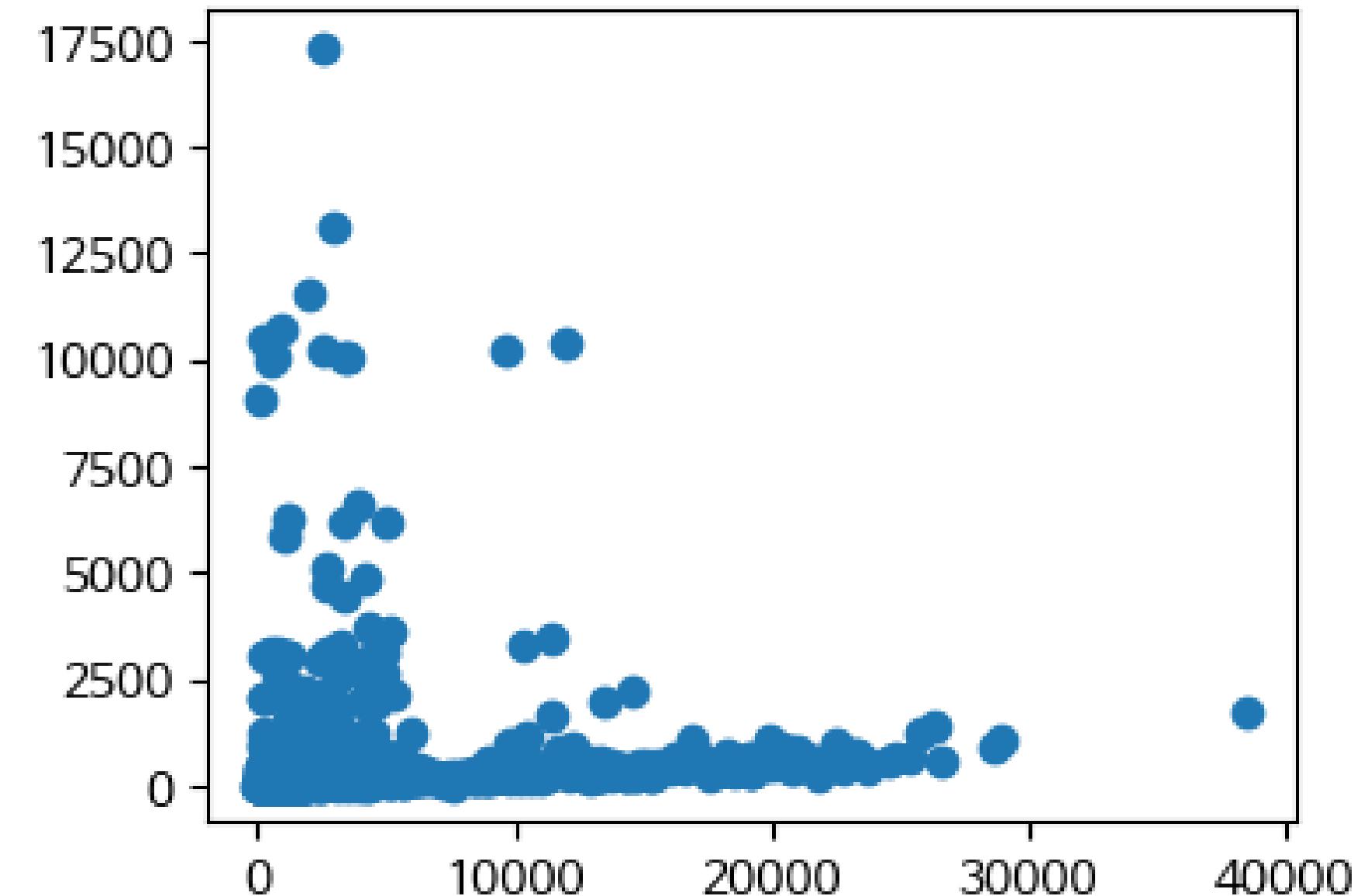
두 변수 간의 관계를 시각화하는 데 사용되는 통계적 그래프

- 각각의 데이터 포인트는 그래프의 x축과 y축 상에서 위치
- 두 변수의 상관관계를 직관적으로 확인할 수 있음

✓ Matplotlib로 그리기

```
plt.scatter(df['어른'], df['청소년'])
```

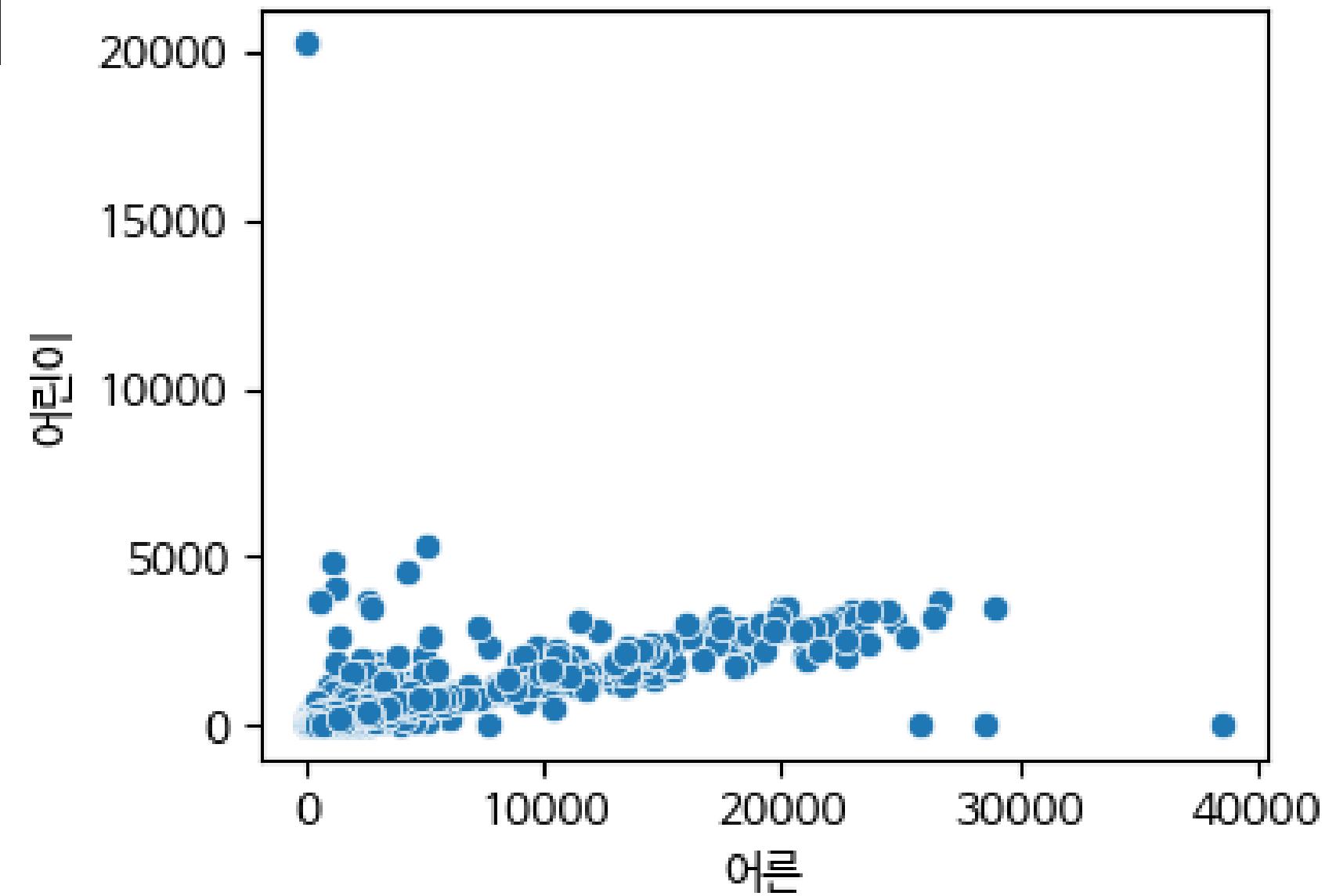
- X좌표는 어른 입장객의 수를 표시
- Y좌표는 청소년 입장객의 수를 표시
- 점 하나는 하루에 해당



✓ Seaborn으로 그리기

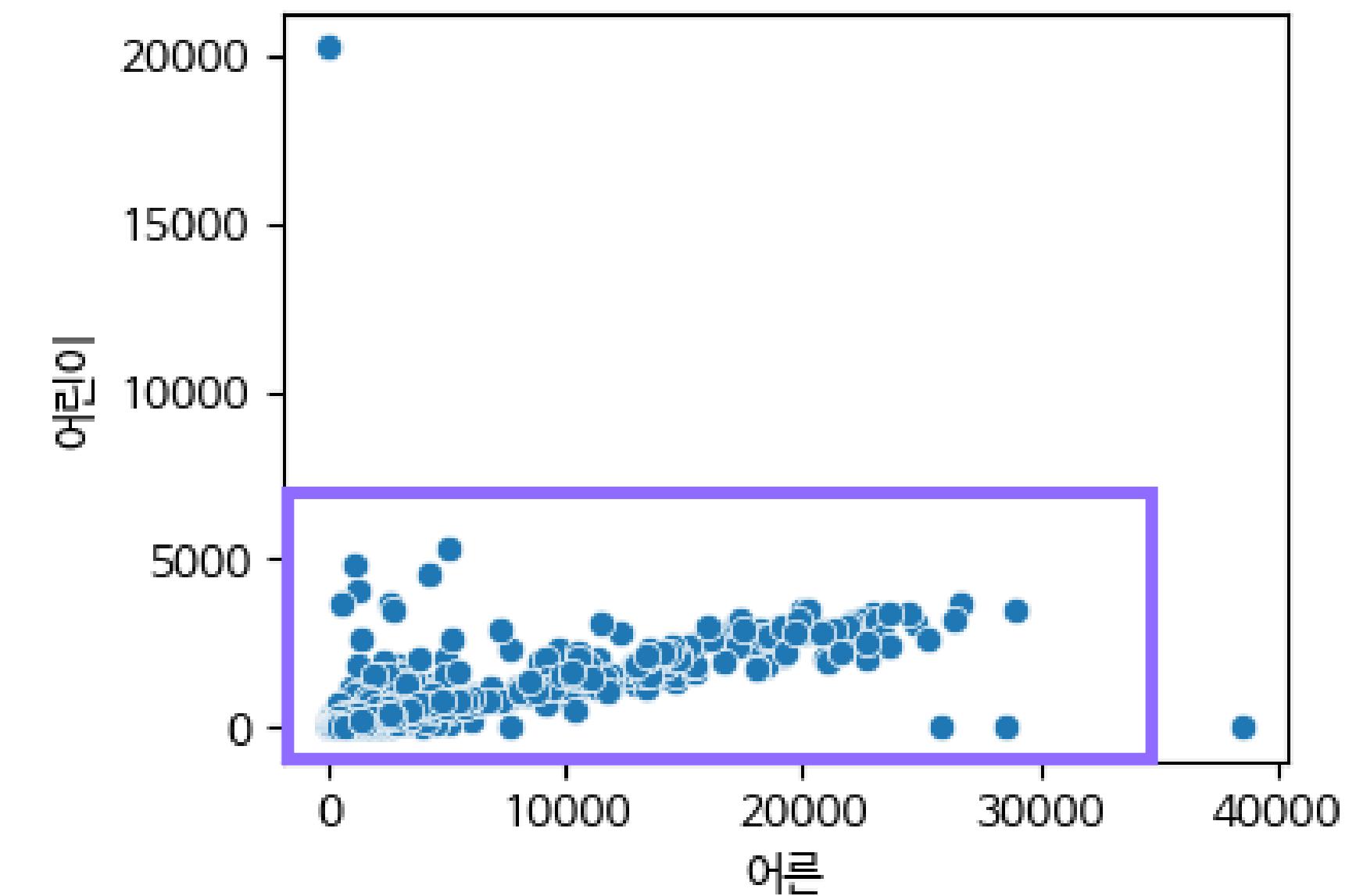
```
sns.scatterplot(data=df, x='어른', y='어린이')
```

- data에 데이터 프레임을 전달
- x, y에 각각의 칼럼 이름을 전달



✓ 특정 범위만 보기

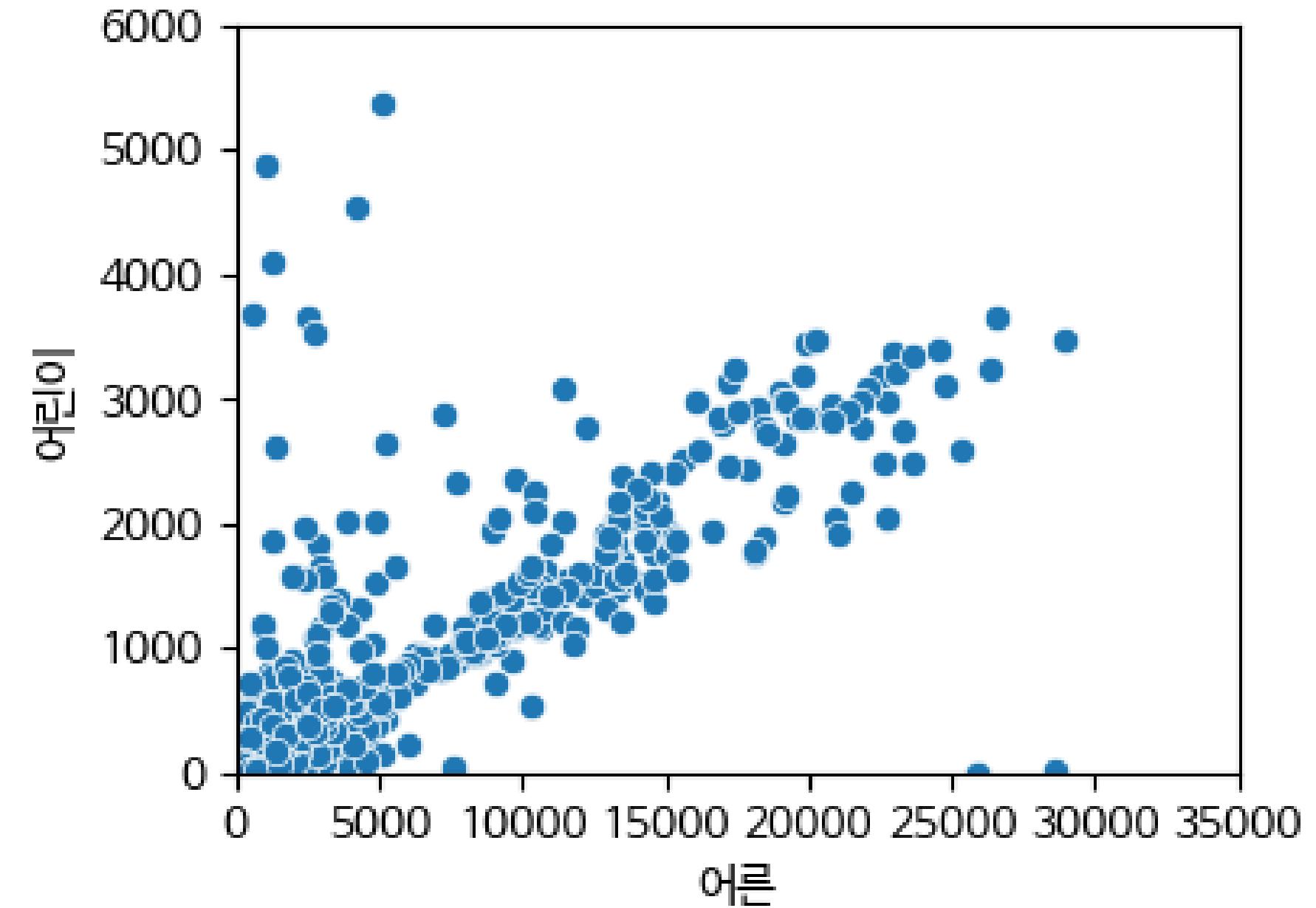
- 대부분 점은 보라색 범위 내에 존재함
- 보라색 상자를 벗어난 점은 극히 소수라서 무시할 수 있음



✓ 특정 범위만 보기

```
sns.scatterplot(data=df, x="어른", y="어린이")  
  
plt.ylim(0, 6000) # y축 범위를 0에서 6000으로 제한  
plt.xlim(0, 35000) # x축 범위를 0에서 6000으로 제한
```

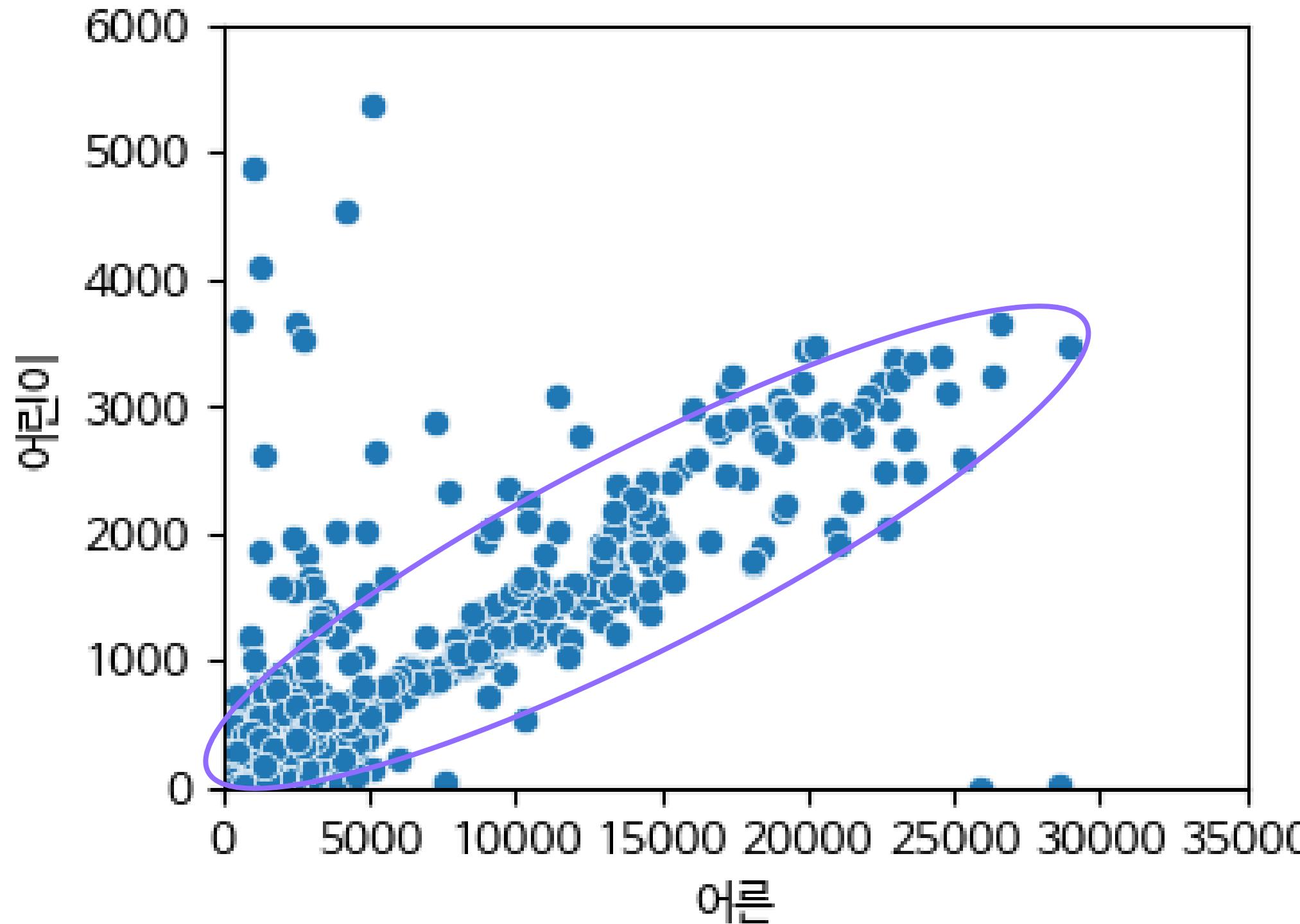
- xlim과 ylim을 이용해 범위를 제한할 수 있음



✓ 상관관계 분석

두 변수 사이의 관련성 또는 연관성

- 두 변수가 함께 움직이는 경향이 있는 경우 상관관계가 있음.
- 양의 상관관계: 한 변수가 증가할 때 다른 변수도 증가하는 경우
- 음의 상관관계: 한 변수가 증가할 때 다른 변수가 감소하는 경우

 상관관계 분석

- 데이터의 분포가 일직선에 가까울수록 상관관계가 있음
- 왼쪽의 그래프에서도 상관관계가 있다고 추측됨

⑤ 상관계수

두 변수 사이의 선형적인 관계의 강도를 나타냄

- 상관계수 값이 1에 가까울수록 양의 상관관계가 강함
- 상관계수 값이 -1에 가까울수록 음의 상관관계가 강함
- 상관계수 값이 0에 가까울수록 두 변수 사이에는 거의 관계가 없거나, 선형적인 관계가 매우 약함

 상관관계 구하기

```
df_corr = df.corr(numeric_only=True)
```

	Unnamed: 0	어른	청소년	어린이	외국인	단체	총입장객수	연	월	일	매출액
Unnamed: 0	1.000000	-0.129602	-0.123936	-0.112757	-0.263132	-0.110094	-0.158224	0.951891	0.101772	0.029936	-0.156220
어른	-0.129602	1.000000	0.079024	0.643471	0.697684	0.094074	0.964853	-0.124890	0.020178	-0.060220	0.964623
청소년	-0.123936	0.079024	1.000000	0.130108	0.073324	0.774788	0.276590	-0.120612	-0.000923	0.022924	0.307690
어린이	-0.112757	0.643471	0.130108	1.000000	0.408734	0.245117	0.701560	-0.122339	0.050375	0.002829	0.731451
외국인	-0.263132	0.697684	0.073324	0.408734	1.000000	0.014294	0.667474	-0.258235	0.023805	-0.067818	0.670458
단체	-0.110094	0.094074	0.774788	0.245117	0.014294	1.000000	0.272871	-0.117731	0.038911	0.085037	0.292723
총입장객수	-0.158224	0.964853	0.276590	0.701560	0.667474	0.272871	1.000000	-0.153188	0.021460	-0.041487	0.988557
연	0.951891	-0.124890	-0.120612	-0.122339	-0.258235	-0.117731	-0.153188	1.000000	-0.197236	0.009130	-0.152929
월	0.101772	0.020178	-0.000923	0.050375	0.023805	0.038911	0.021460	-0.197236	1.000000	-0.016435	0.024640
일	0.029936	-0.060220	0.022924	0.002829	-0.067818	0.085037	-0.041487	0.009130	-0.016435	1.000000	-0.044896
매출액	-0.156220	0.964623	0.307690	0.731451	0.670458	0.292723	0.988557	-0.152929	0.024640	-0.044896	1.000000

⑤ 히트맵

데이터 값의 상대적인 크기를 색상으로 표시하여 시각화하는 그래프

- 2차원 표 형태의 데이터를 사용하여 생성
- 값의 크기에 따라 다른 색상 맵을 사용하며, 값이 클수록 진한 색, 작을수록 연한 색으로 표현됨

 히트맵 그리기

```
sns.heatmap(df_corr)
```

- heatmap 함수에 데이터 프레임을 전달
- 값이 1이나 -1에 가까울수록 상관관계가 있다는 것을 나타냄

