06_그래프 그리기

<2가지 라이브러리 필요>

1) seaborn : 학습 데이터 제공 및 분석에 필요한 시각화 잘해줌
   import seaborn as sns

2) matplotlib.pyplot as plt
   그래프 + 그래프를 꾸미는 것 , 크기 등등 기본,도움 + pandas 자체 그래프

3) plotly 지도, 3D 그래프 입체그래프

4) folium 지도 그래프 강점

그래프 그리기

- 수많은 데이터를 한 눈에 파악하는 방법

> 그래프

- Histogram
- Scatter
- Density plot
- Bar plot
- Box plot
- Line chart
- …

집계, 축약해서 보여주는 것이 통계의 핵심
: 수치와 시각화도 중요

> 통계량

- Min, Max, Sum
- Mean, Std
- 사분위수
- 검정 통계량
- P-value
- …

[분석방법]
EDA:눈으로 확인하는/그래프
CDA:확증적 데이터분석/전통적 방식

그래프 그리기

• 수많은 데이터를 한 눈에 파악하는 방법

> 데이터 요약의 목적

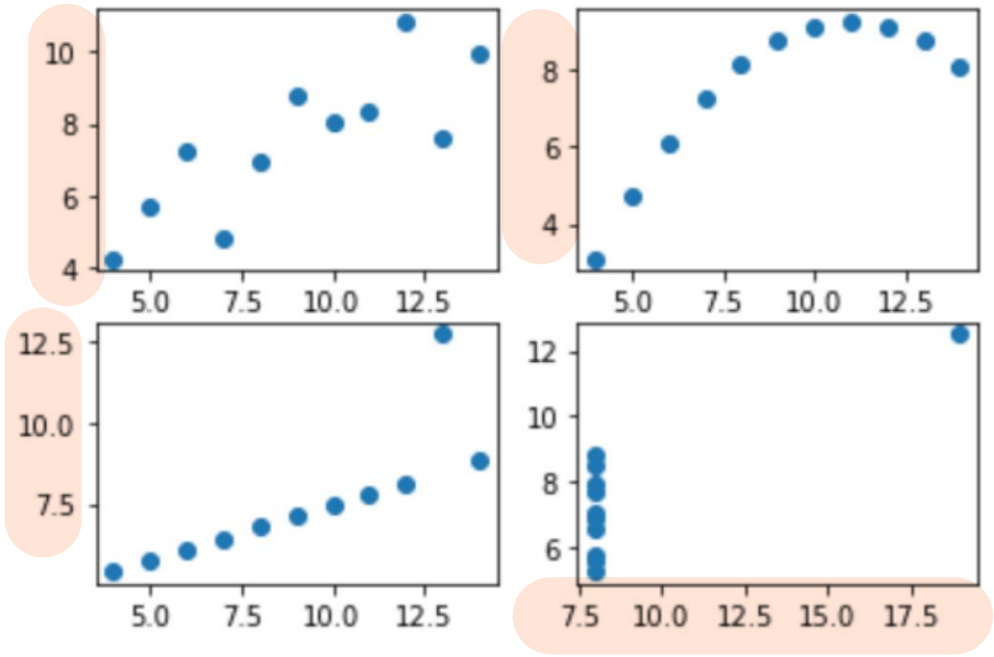
- 정보화시대에 데이터는 비즈니스를 담고 있다.
- 시각화, 통계량을 통해 방대한 데이터를 요약하여
- 비즈니스의 인사이트를 파악할 수 있다.

그래프, 통계량을 통한 데이터 요약은 정보의 손실이 발생함.

• 데이터 시각화가 필요한 이유

> 앤스콤 4분할 그래프

- 데이터 시각화의 전형적인 사례
- 시각화 하지 않고 수치만 확인할 때 발생할 수 있는 문제점을 보여줌
- 4개의 데이터 그룹으로 구성되며, 평균, 분산, 상관관계, 회귀선이 모두 같은 데이터의 특성이 있다.
  - 4개의 특성이 같으므로 4개의 데이터는 비슷한 유형이다 라는 착각
- 앤스콤 그래프



• 데이터 시각화가 필요한 이유

> **앤스콤 데이터 집합 불러오기**
  - seaborn 라이브러리 데이터셋

```
import seaborn as sns
anscombe = sns.load_dataset('anscombe')
anscombe.head()
```

Out :

	dataset	x	y
0	I	10.0	8.04
1	I	8.0	6.95
2	I	13.0	7.58
3	I	9.0	8.81
4	I	11.0	8.33

Seaborn에서 데이터학습에 도움을 주는 데이터셋 제공함

• 데이터 시각화가 필요한 이유

> **데이터 살펴보기**
  - 데이터셋 별로 평균과 분산 구하기

```
anscombe.groupby('dataset').mean()
```

Out :

dataset	x	y
I	9.0	7.500909
II	9.0	7.500909
III	9.0	7.500000
IV	9.0	7.500909

```
anscombe.groupby('dataset').var()
```

Out :

dataset	x	y
I	11.0	4.127269
II	11.0	4.127629
III	11.0	4.122620
IV	11.0	4.123249

• 데이터 시각화가 필요한 이유

> **데이터 살펴보기**

- 데이터셋 별로 상관관계 구하기 – corr() 함수

```
anscombe.groupby('dataset').corr()
```

Out :

		x	y
dataset			
I	x	1.000000	0.816421
	y	0.816421	1.000000
II	x	1.000000	0.816237
	y	0.816237	1.000000
III	x	1.000000	0.816287
	y	0.816287	1.000000
IV	x	1.000000	0.816521
	y	0.816521	1.000000

그래프 그리기

- 데이터 시각화가 필요한 이유

한장에 볼 수 있게
sns.scatterplot(data=anscombe, x='x',y='y', hue='dataset')

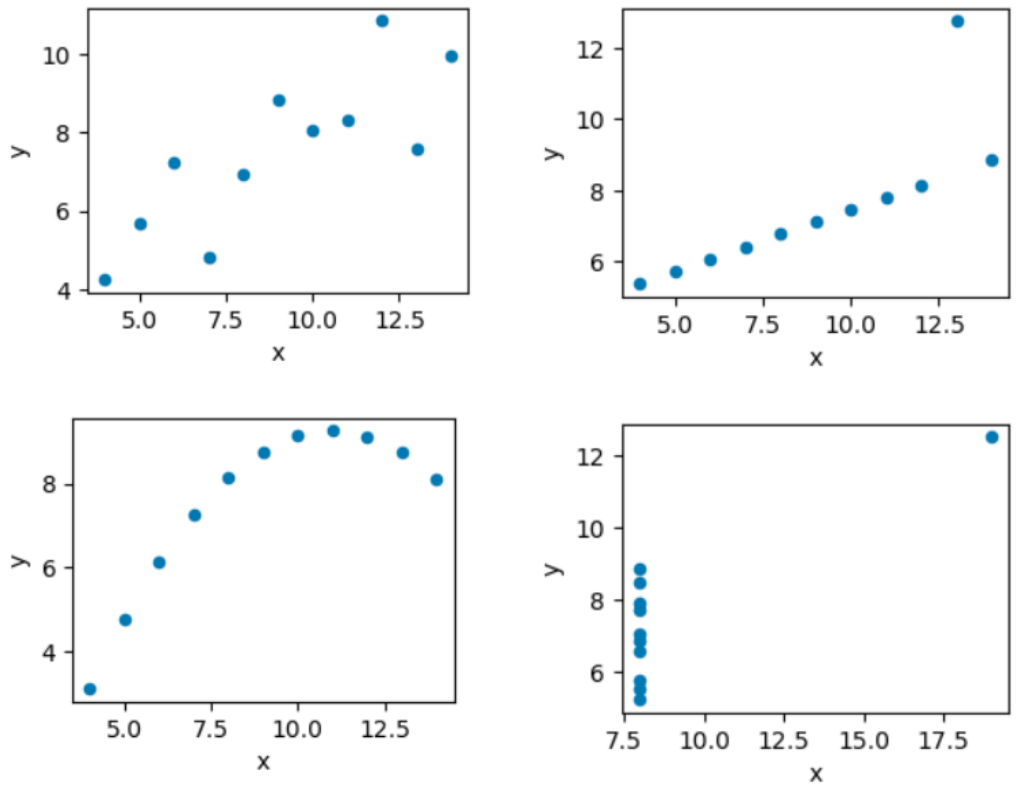
> 그래프 그리기

- 데이터셋 별로 그래프 그리기

dataset의 x,y의 컬럼을 의미/ 그러므로 'x''y'

```
anscombe.groupby('dataset').plot('x', 'y', kind='scatter')
```

Out :



그래프 그리기

- Matplotlib

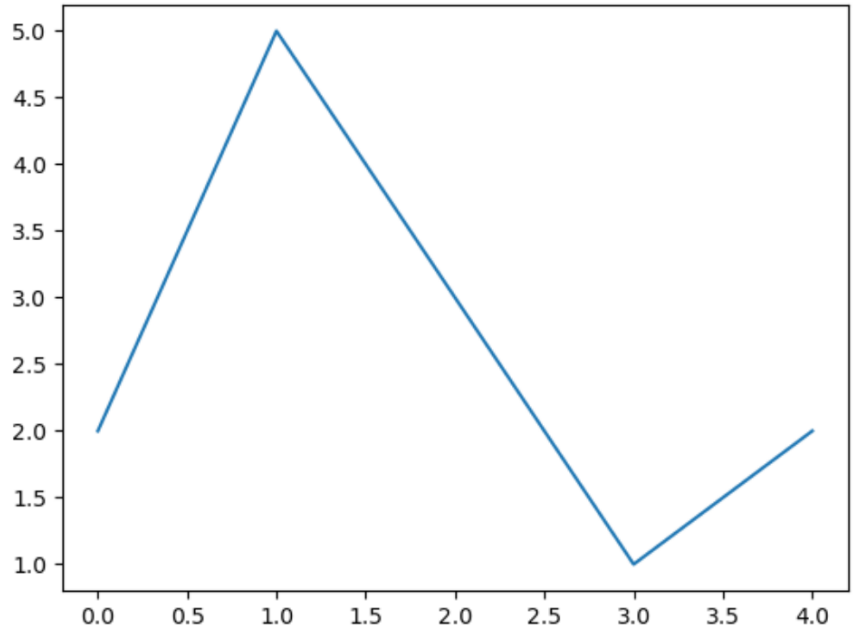
> 기본기

```
import matplotlib.pyplot as plt
```
기본 라이브러리

```
plt.plot([2, 5, 3, 1, 2])
```
plot는 라인 차트를 의미

```
plt.show()
```
y값만 넣었지만 x값은 인덱스값을 자동으로 배치됨 [0,1,2,3...]

Out :



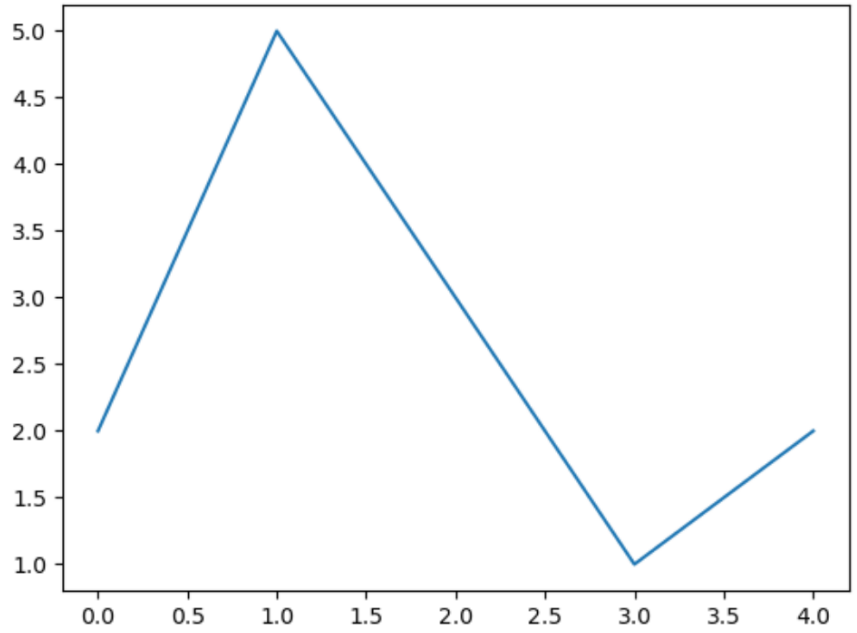
그래프 그리기

- Matplotlib

> 이미지 저장하기

```
plt.plot([2, 5, 3, 1, 2])
plt.savefig('a')
```

Out :



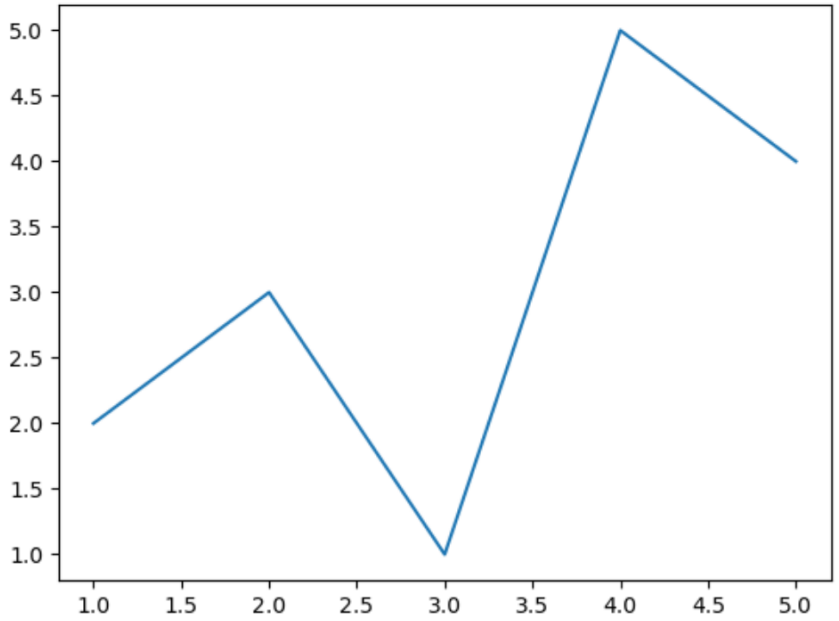
그래프 그리기

- Matplotlib

> X, y축 지정

```
X = [1, 2, 3, 4, 5]
y = [2, 3, 1, 5, 4]
plt.plot(X, y)
```

Out :



그래프 그리기

- Matplotlib

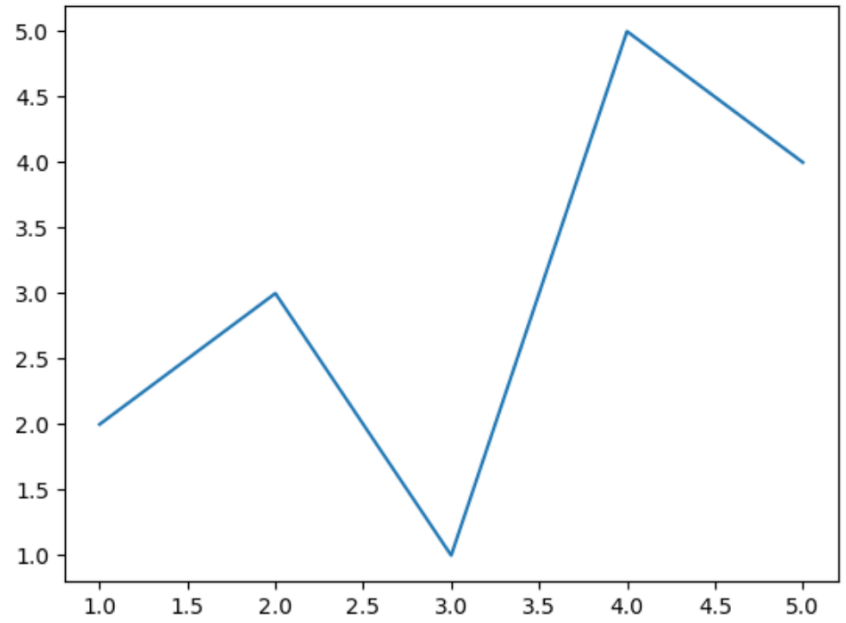
> X, y축 지정

```
import pandas as pd
df = pd.DataFrame({'X':[1, 2, 3, 4, 5],
                   'y':[2, 3, 1, 5, 4]})
plt.plot('X', 'y', data=df)
```

원래는 이렇게 써야 함

```
plt.plot(df['x'],df['y'])
```

Out :



'y'변수 이름 바꾸라고 경고함
: 바꿈

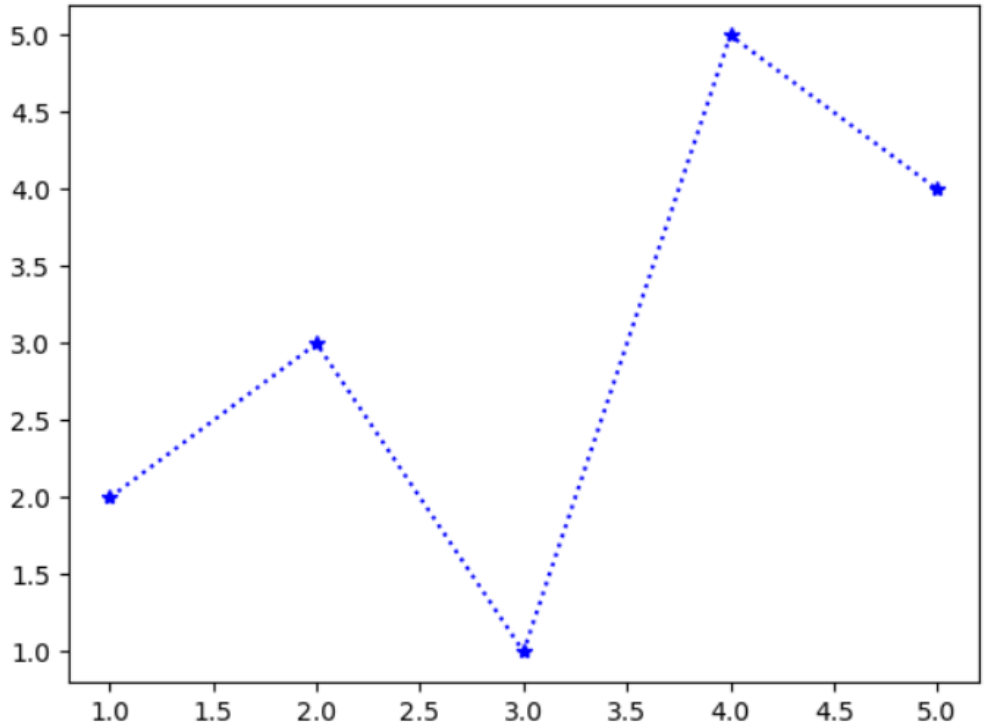
그래프 그리기

- Matplotlib

> 라인 스타일 조정

```
plt.plot('X', 'y', 'b*:', data=df)
```

Out :



그래프 그리기

- Matplotlib

plot 그래프에서만 적용됨

> 라인 스타일 조정

character	color
‘b’	blue
‘g’	green
‘r’	red
‘c’	cyan
‘m’	magenta
‘y’	yellow
‘k’	black
‘w’	white

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

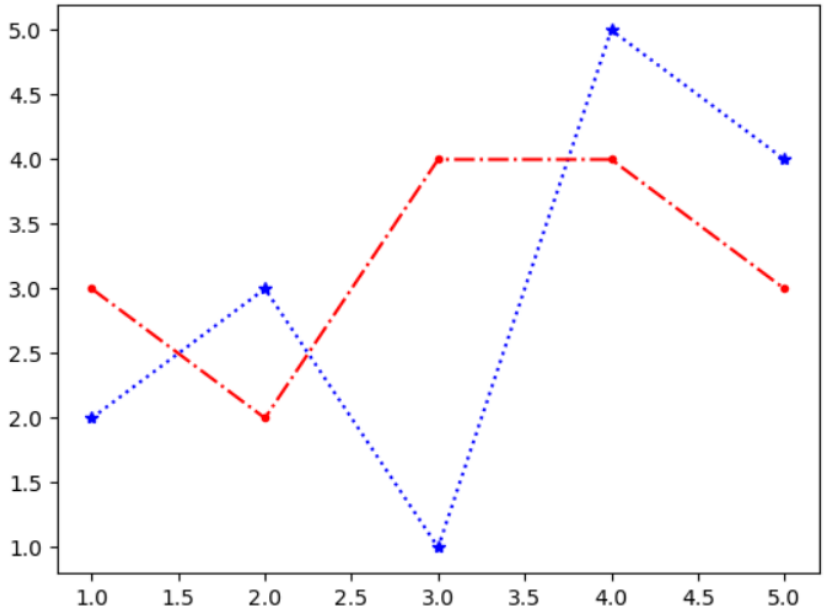
그래프 그리기

- Matplotlib

> 그래프 겹치기

```
df['y2'] = [3, 2, 4, 4, 3]
plt.plot('X', 'y', 'b*:', data=df)
plt.plot('X', 'y2', 'r.-.', data=df)
```

Out :



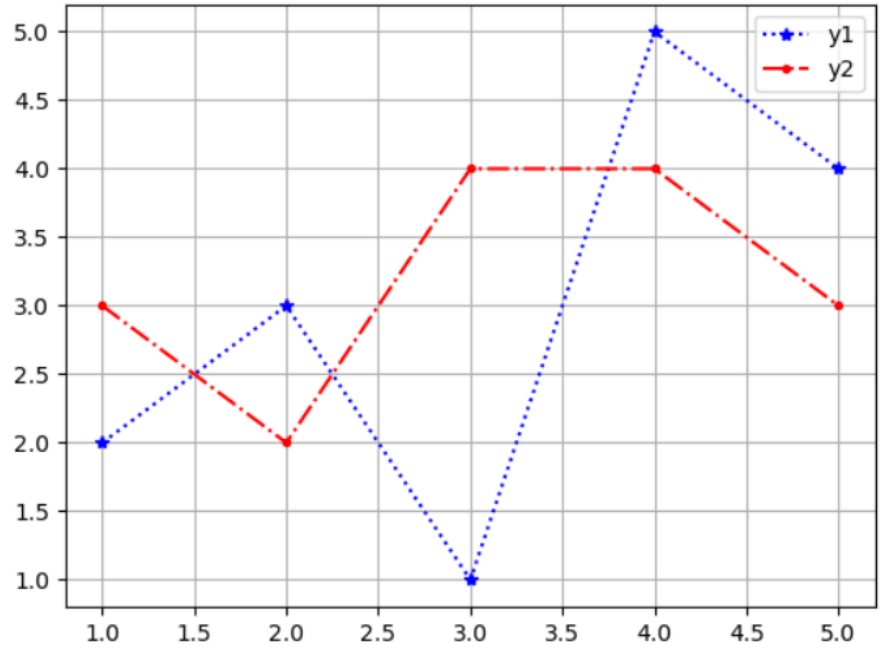
그래프 그리기

- Matplotlib

> 범례, 격자 그래프

```
plt.plot('X', 'y', 'b*:', data=df, label='y1')
plt.plot('X', 'y2', 'r.-.', data=df, label='y2')
plt.grid()
plt.legend()
```

Out :

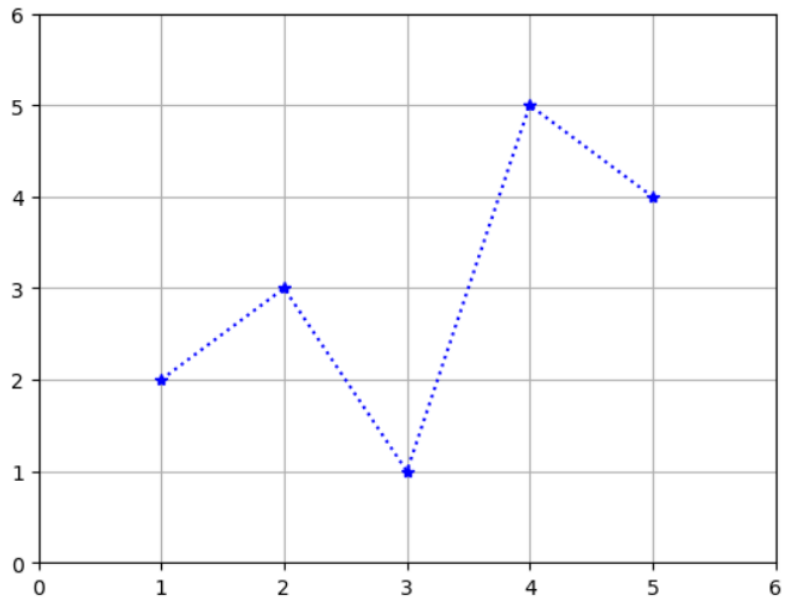


- Matplotlib

> 축 범위 지정

```
plt.plot('X', 'y', 'b*:', data=df)
plt.xlim(0, 6)
plt.ylim(0, 6)
plt.grid()
```

Out :



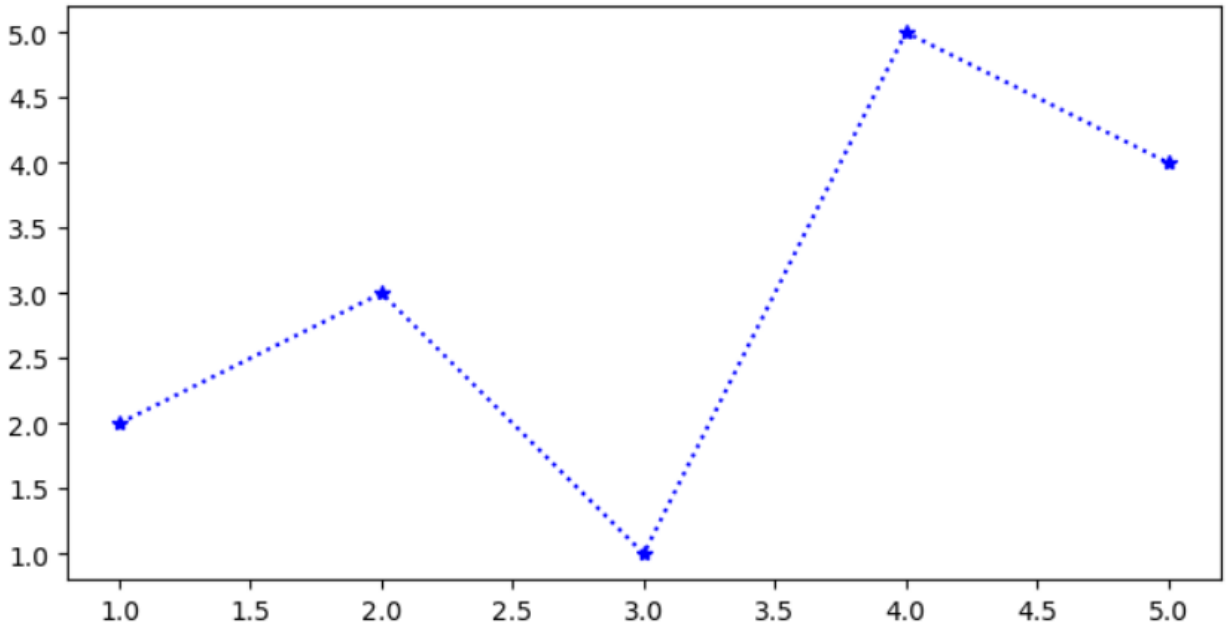
그래프 그리기

- Matplotlib

> 그래프 크기 조정

```
plt.figure(figsize = (8, 4))
plt.plot('X', 'y', 'b*:', data=df)
```

Out :



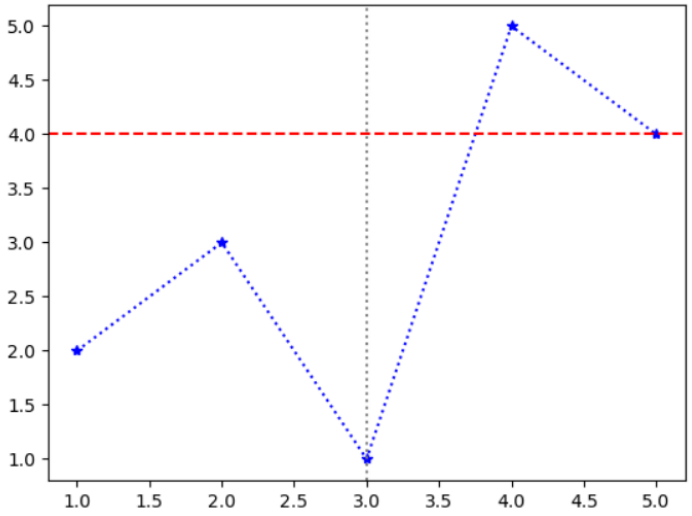
- Matplotlib

> 수평선, 수직선 추가하기

- axhline(y축값, color, linestyle)
- axvline(x축값, color, linestyle)

```
plt.plot('X', 'y', 'b*:', data=df)
plt.axhline(4, color='red', linestyle = '--')
plt.axvline(3, color='grey', linestyle = ':')
```

Out :



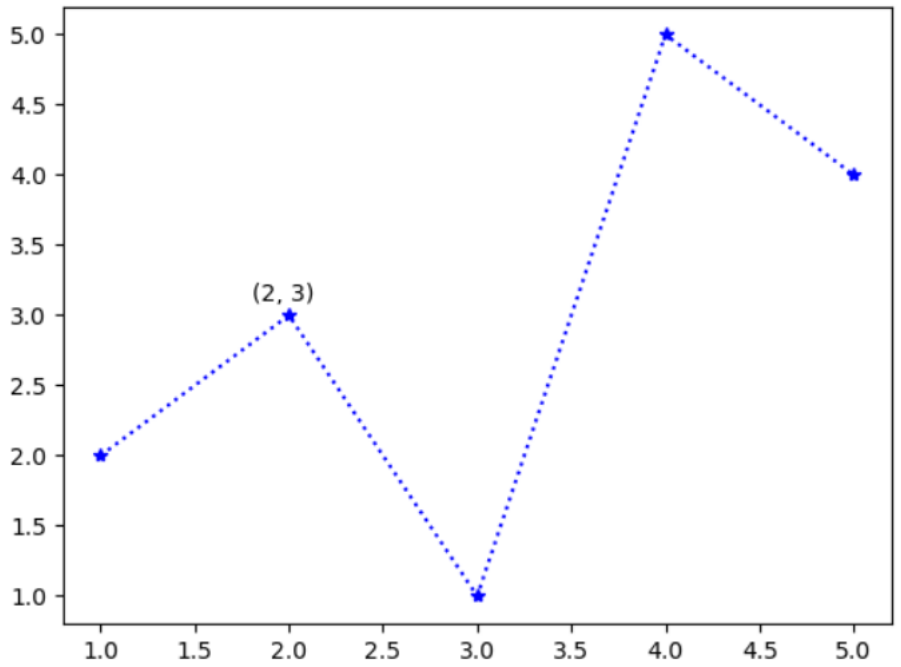
그래프 그리기

- Matplotlib

> 그래프에 텍스트 추가

```
plt.plot('X', 'y', 'b*:', data=df)
plt.text(1.8, 3.1, '(2, 3)')
```

Out :



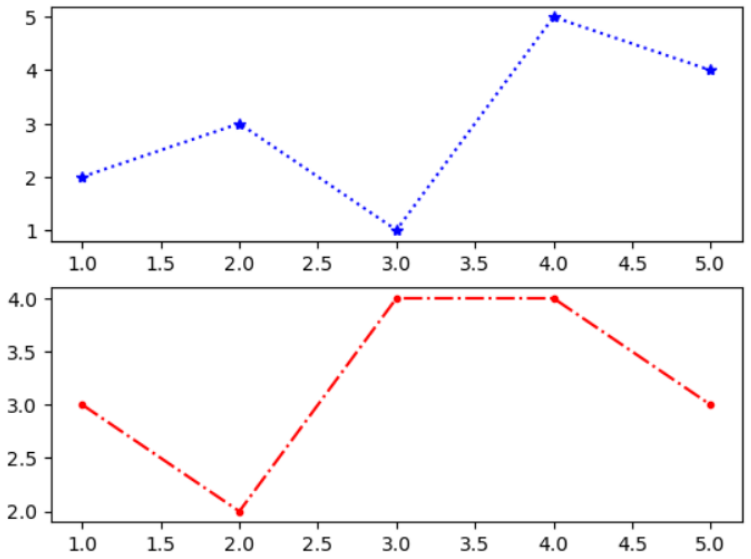
그래프 그리기

- Matplotlib

> 그래프 나누기

```
plt.subplot(2, 1, 1)
plt.plot('X', 'y', 'b*:', data=df)
plt.subplot(2, 1, 2)
plt.plot('X', 'y2', 'r.-.', data=df)
```

Out :



- Matplotlib

> 그래프 나누기

- subplot(rows, cols, index)

그래프 그리기

- Matplotlib

> 그래프 나누기

- title() – 그래프의 title
- suptitle() – 그래프의 큰 title
- tight_layout() – 그래프 간격 조정

```
plt.subplot(2, 1, 1)
plt.plot('X', 'y', 'b*:', data=df)
plt.title('Graph1')
plt.subplot(2, 1, 2)
plt.plot('X', 'y2', 'r.-.', data=df)
plt.title('Graph2')
plt.suptitle('Test Graph')
plt.tight_layout()
```

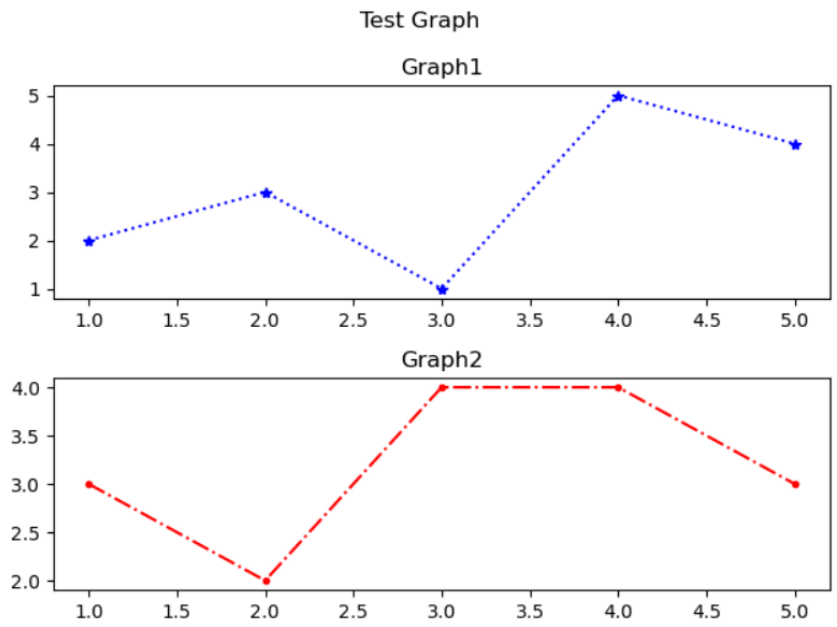
그래프 그리기

- Matplotlib

> 그래프 나누기

- title() – 그래프의 title
- suptitle() – 그래프의 큰 title
- tight_layout() – 그래프 간격 조정

Out :



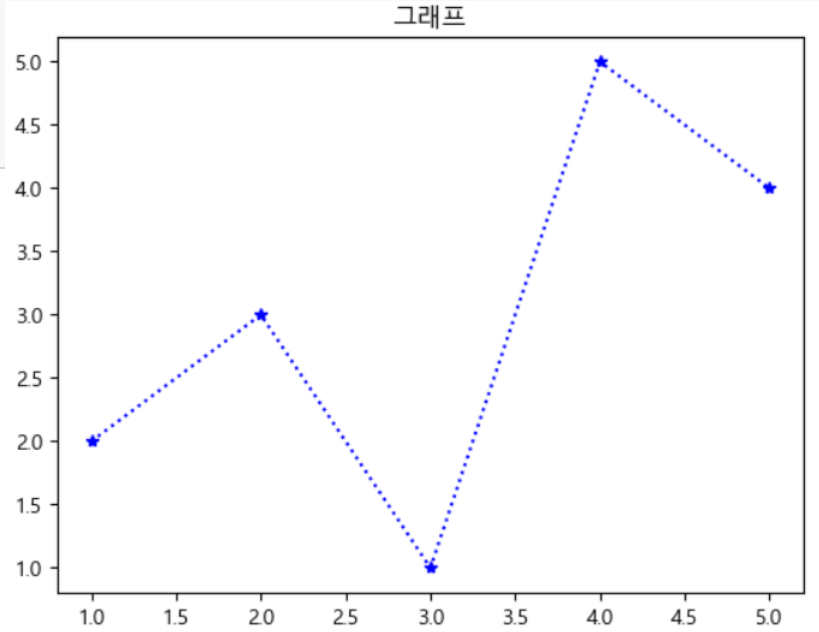
• Matplotlib

> 한글 표현하기

```
from matplotlib import font_manager, rc
font = 'C:/Windows/Fonts/Malgun.ttf'
font_name = font_manager.FontProperties(fname=font).get_name()
rc('font', family=font_name)
plt.plot('X', 'y', 'b*:', data=df)
plt.title('그래프')
```

Out :

python은 영어만 지원
(특수문자 등 포함)



• 다양한 그래프

> 데이터 준비

- seaborn 라이브러리의 tips 데이터셋

```
tips = sns.load_dataset('tips')
tips
```

Out :

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

그래프 그리기

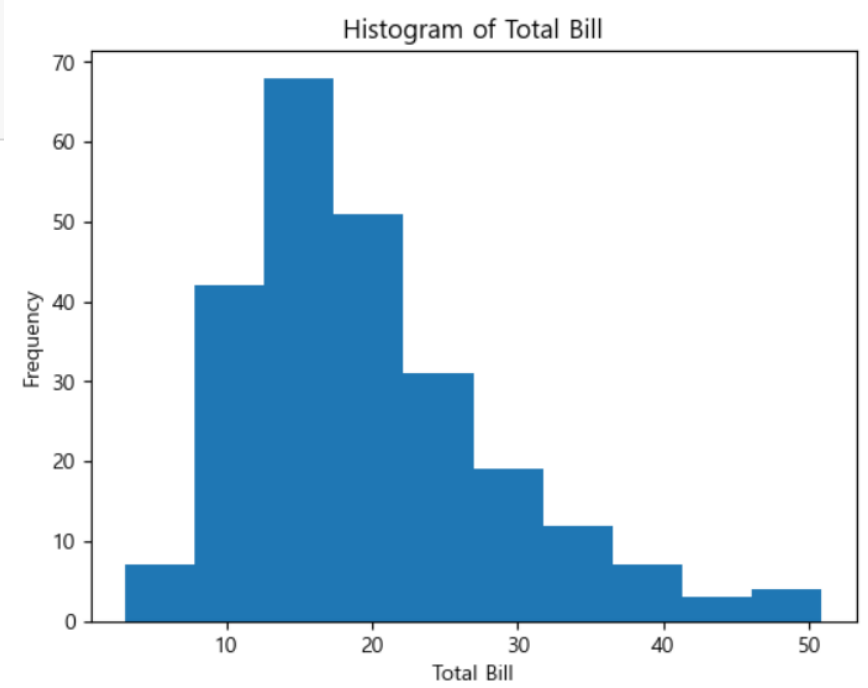
• 다양한 그래프

> **히스토그램** 일변량 그래프

- 숫자형 데이터의 분포와 빈도를 표현

```
hist = plt.hist(tips['total_bill'], bins=10)
plt.title('Histogram of Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Frequency')
```

Out :



수치형/ 연속형(실수:) 데이터가 가장 적합

• 다양한 그래프

> 히스토그램

- 숫자형 데이터의 분포와 빈도를 표현

```
print('빈도수 :', hist[0])
print('구간값 :', hist[1])
```

구간의 경계값

Out : x,y의 count값

```
빈도수 : [ 7. 42. 68. 51. 31. 19. 12.  7.  3.  4.]
구간값 : [ 3.07   7.844 12.618 17.392 22.166 26.94
          31.714 36.488 41.262 46.036 50.81 ]
```

• 다양한 그래프

> 박스 그래프

- 집단 간의 분포 차이를 표현

```
series = []
for group in tips.groupby('sex')['total_bill']:
    series.append(group[1])   group의 1번 인덱스는 남,여
series          남,여로 그룹 지워진 리스트 생성
```

Out :

```
[1      10.34
 2      21.01
 3      23.68
 5      25.29
 6       8.77
        ...
236     12.60
237     32.83
239     29.03
241     22.67
242     17.82
Name: total_bill, Length: 157, dtype: float64,
```

```
0      16.99
4      24.59
11     35.26
14     14.83
16     10.33
       ...
226    10.09
229    22.12
238    35.83
240    27.18
243    18.78
Name: total_bill, Length: 87, dtype: float64]
```

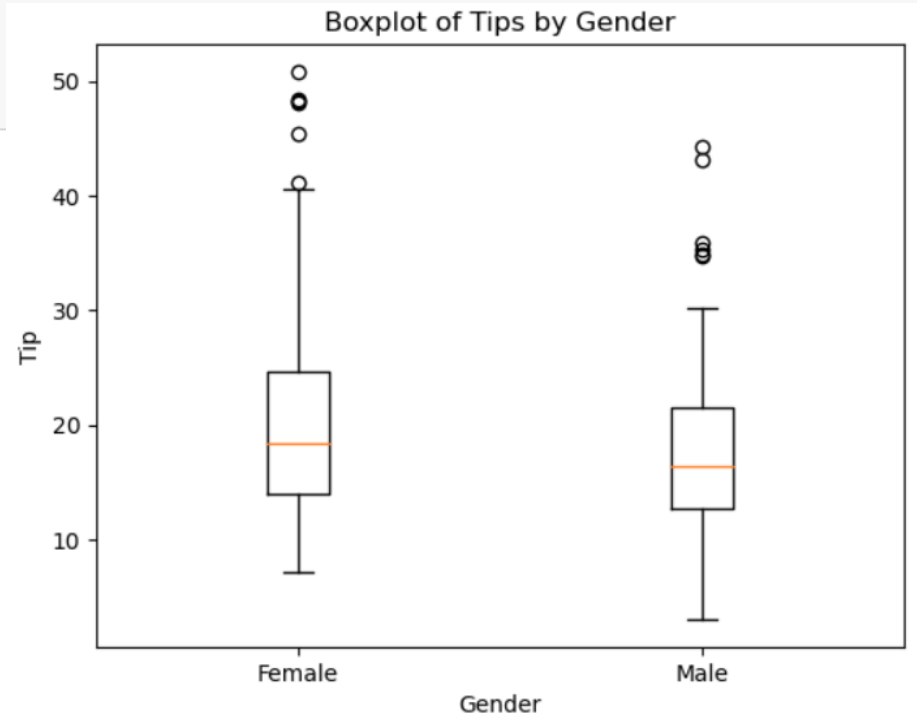
• 다양한 그래프

> 박스 그래프

- 집단 간의 분포 차이를 표현

```
box1 = plt.boxplot(series, labels=['Female', 'Male'])
plt.title('Boxplot of Tips by Gender')
plt.xlabel('Gender')
plt.ylabel('Tip')
```

Out :



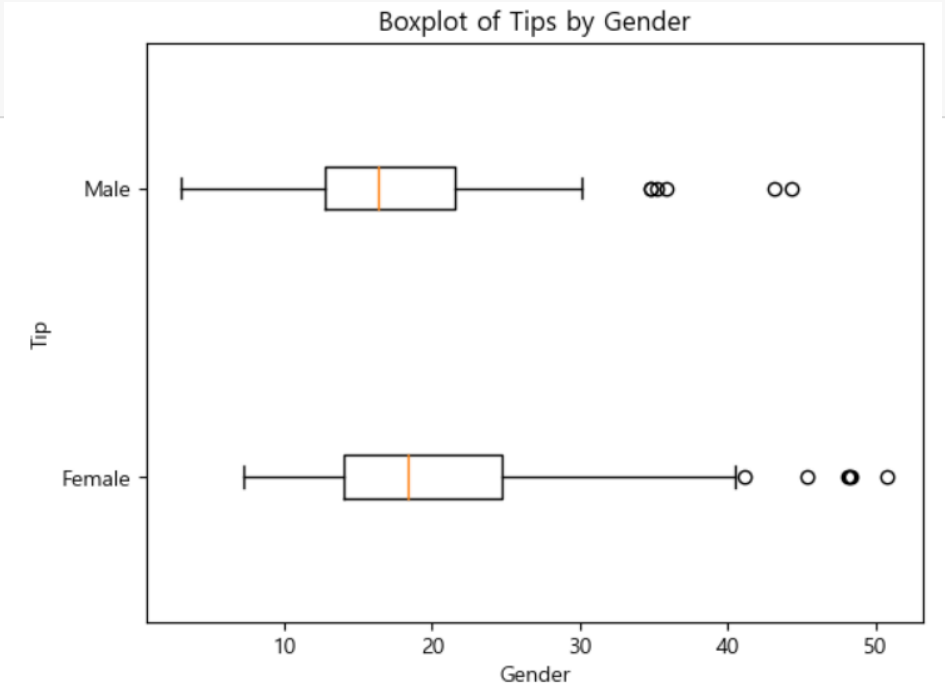
• 다양한 그래프

> 박스 그래프

- vert를 통해 가로로 그리기 가능

```
box2 = plt.boxplot(series, labels=['Female', 'Male'], vert = False)
plt.title('Boxplot of Tips by Gender')
plt.xlabel('Gender')
plt.ylabel('Tip')
```

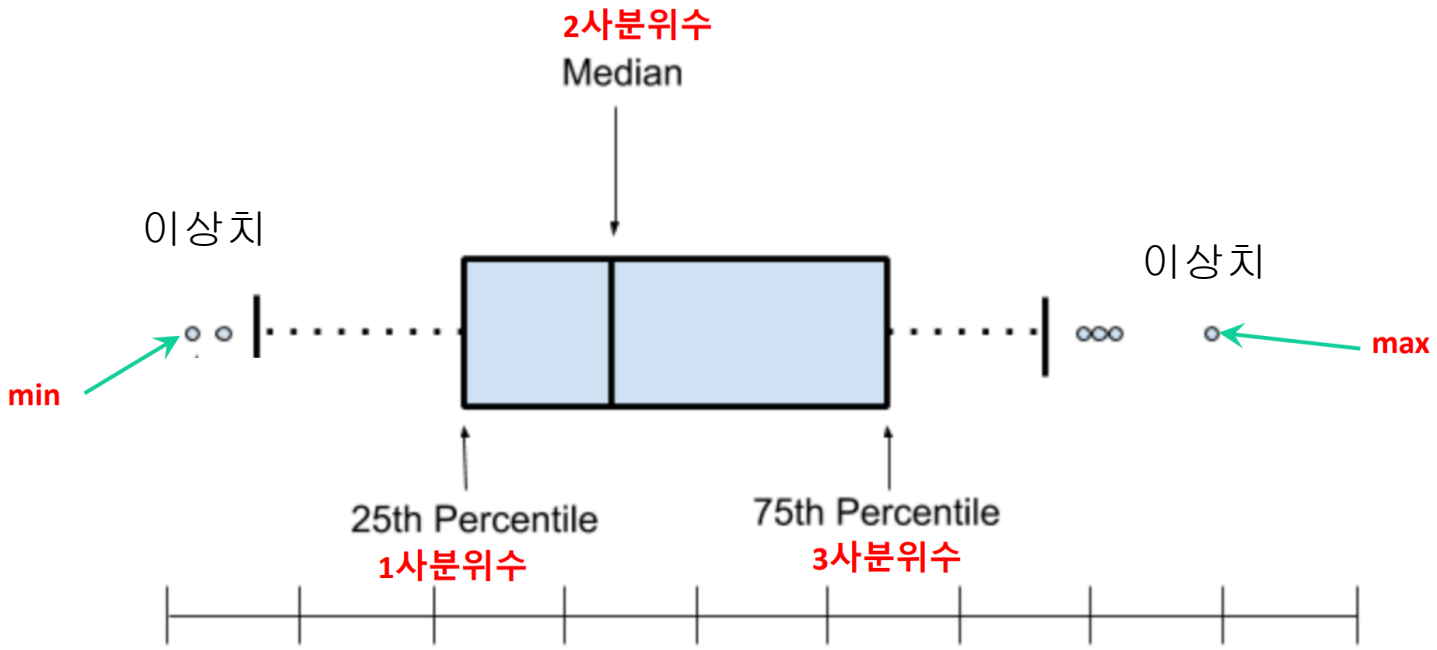
Out :



• 다양한 그래프

> 박스 그래프

- 집단 간의 분포 차이를 표현



• 다양한 그래프

> 박스 그래프

- whiskers(사분위값)

```
print(box1['whiskers'][0].get_ydata()) # 아래쪽 max, min
print(box1['whiskers'][1].get_ydata()) # 위쪽 min, max
print(series[0].describe())
```

Out : [14. 7.25]
[24.71 40.55]

```
count    157.000000
mean      20.744076
std        9.246469
min        7.250000
25%       14.000000
50%       18.350000
75%       24.710000
max       50.810000
Name: total_bill, dtype: float64
```

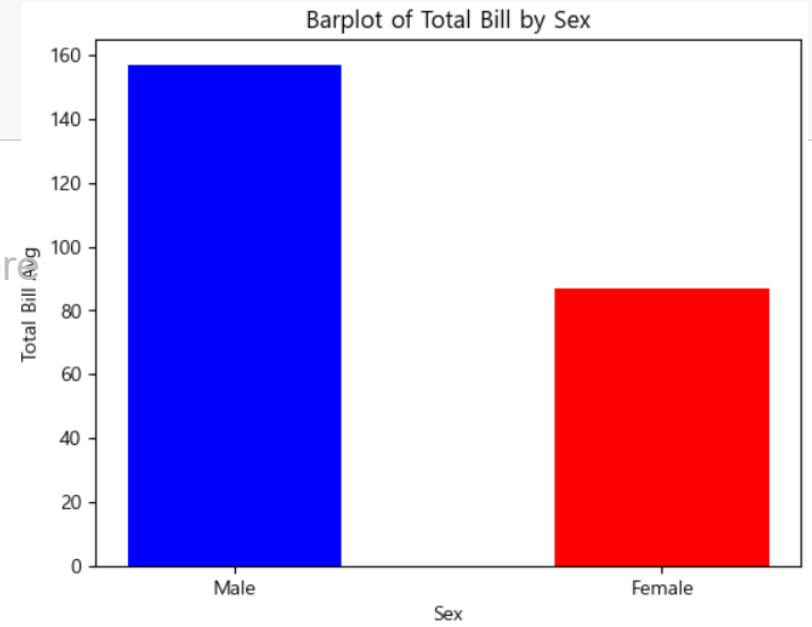
• 다양한 그래프

> 막대 그래프

- 집단 간의 차이를 표현

```
s_tips = tips['sex'].value_counts()
plt.bar(s_tips.index, s_tips.values, color = ['b', 'r'], width = .5)
plt.title('Barplot of Total Bill by Sex')
plt.xlabel('Sex')
plt.ylabel('Total Bill Count')
```

Out :



• 다양한 그래프

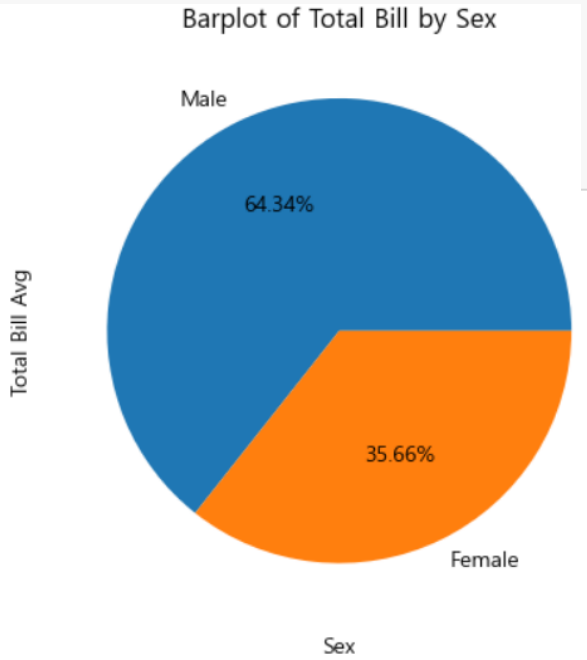
> 막대 그래프

- 집단 간의 비율을 표현

```
s_tips = tips['sex'].value_counts()
plt.pie(s_tips.values, labels = s_tips.index, autopct = '%.2f%%')
plt.title('Pie chart of Total Bill by Sex')
plt.xlabel('Sex')
plt.ylabel('Total Bill Ratio')
```

비율 지정: 소숫점 2자리표현 %%는 %

Out :



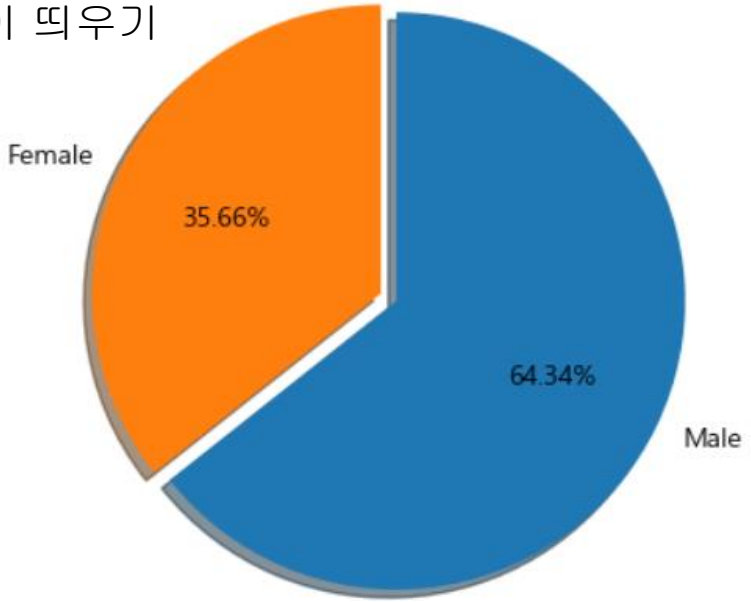
• 다양한 그래프

> 막대 그래프

- startangle, counterclock, explode, shadow

```
plt.pie(s_tips.values, labels = s_tips.index, autopct = '%.2f%%',
        startangle = 90, counterclock = False, 시계반대 방향
시작각도 explode = [0.03, 0.03], shadow = True)
```

Out : 사이 띄우기



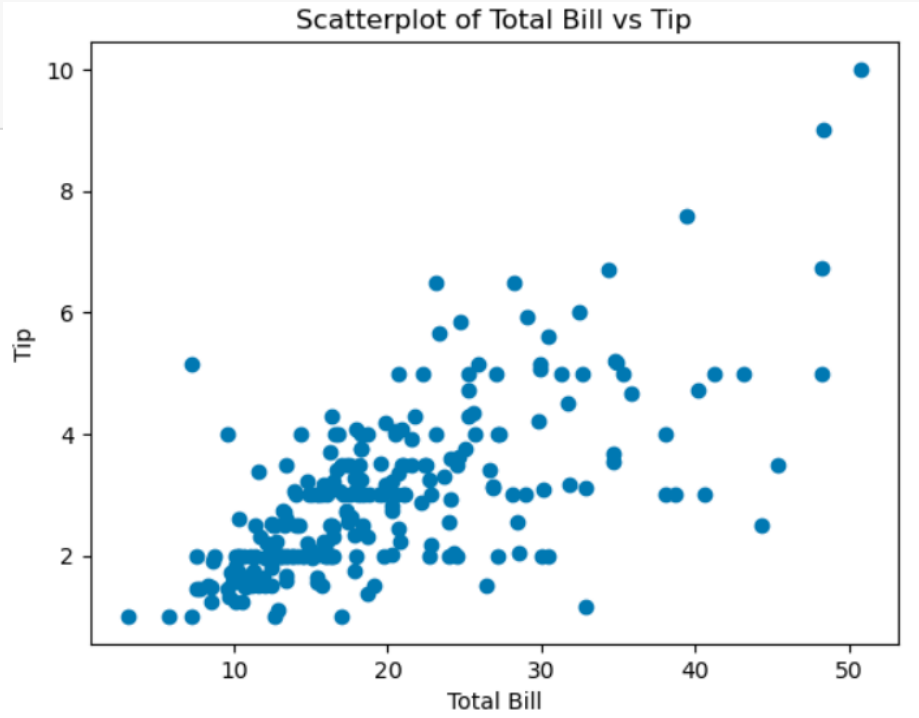
• 다양한 그래프

> 산점도

- 데이터의 분포를 표현

```
plt.scatter(tips['total_bill'], tips['tip'])
plt.title('Scatterplot of Total Bill vs Tip')
plt.xlabel('Total Bill')
plt.ylabel('Tip')
```

Out :



• 다양한그래프

> 3개 이상의 변수를 사용한 그래프

- x축, y축을 제외한 색상, 크기 등의 차이로 데이터를 구분

```
def recode_gender(gender):
    if gender == 'Female':
        return 'red'
    else:
        return 'blue'
tips['color'] = tips['sex'].apply(recode_gender)
tips.head()
```

Out :

	total_bill	tip	sex	smoker	day	time	size	color
0	16.99	1.01	Female	No	Sun	Dinner	2	red
1	10.34	1.66	Male	No	Sun	Dinner	3	blue
2	21.01	3.50	Male	No	Sun	Dinner	3	blue
3	23.68	3.31	Male	No	Sun	Dinner	2	blue
4	24.59	3.61	Female	No	Sun	Dinner	4	red

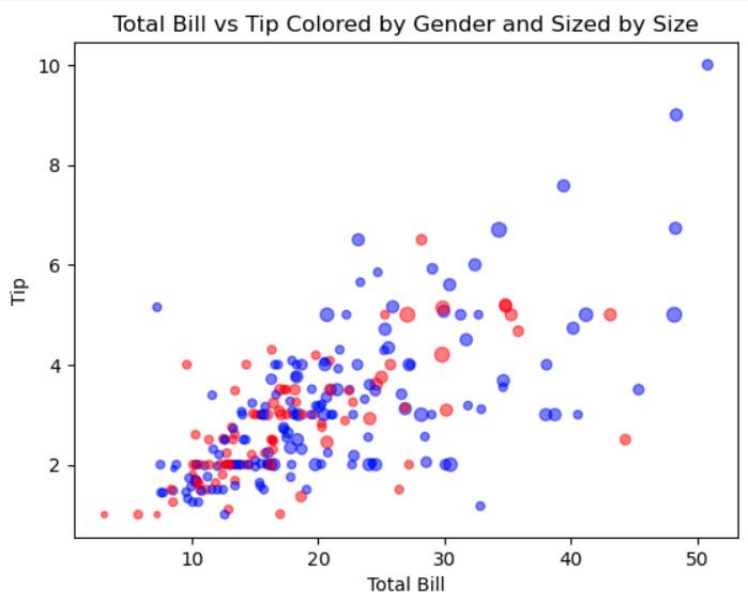
• 다양한그래프

> 3개 이상의 변수를 사용한 그래프

- x축, y축을 제외한 색상, 크기 등의 차이로 데이터를 구분

```
plt.scatter(x = tips['total_bill'], y = tips['tip'], s = tips['size'] * 10,
            c = tips['color'], alpha = 0.5)
plt.title('Total Bill vs Tip Colored by Gender and Sized by Size')
plt.xlabel('Total Bill')
plt.ylabel('Tip')
```

Out :



그래프 그리기

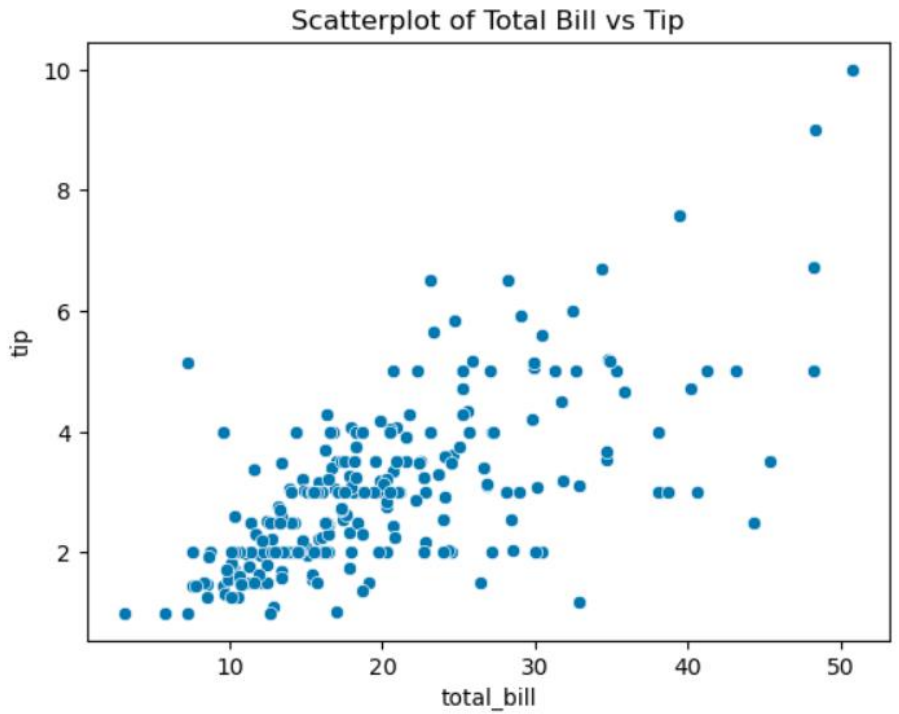
• Seaborn 라이브러리

> 기본 그래프

- 산점도 scatterplot

```
sns.scatterplot(x = 'total_bill', y='tip', data=tips)
plt.title('Scatterplot of Total Bill vs Tip')
```

Out :



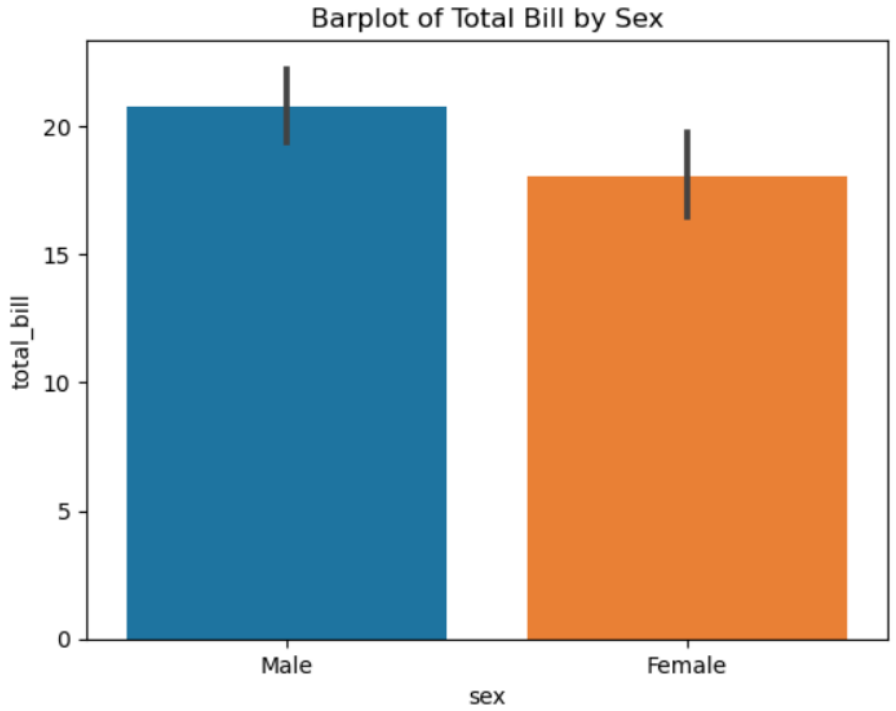
• Seaborn 라이브러리

> 기본 그래프

- 막대 barplot, countplot

```
sns.barplot(data = tips, x = 'sex', y = 'total_bill')
plt.title('Barplot of Total Bill by Sex')
```

Out :



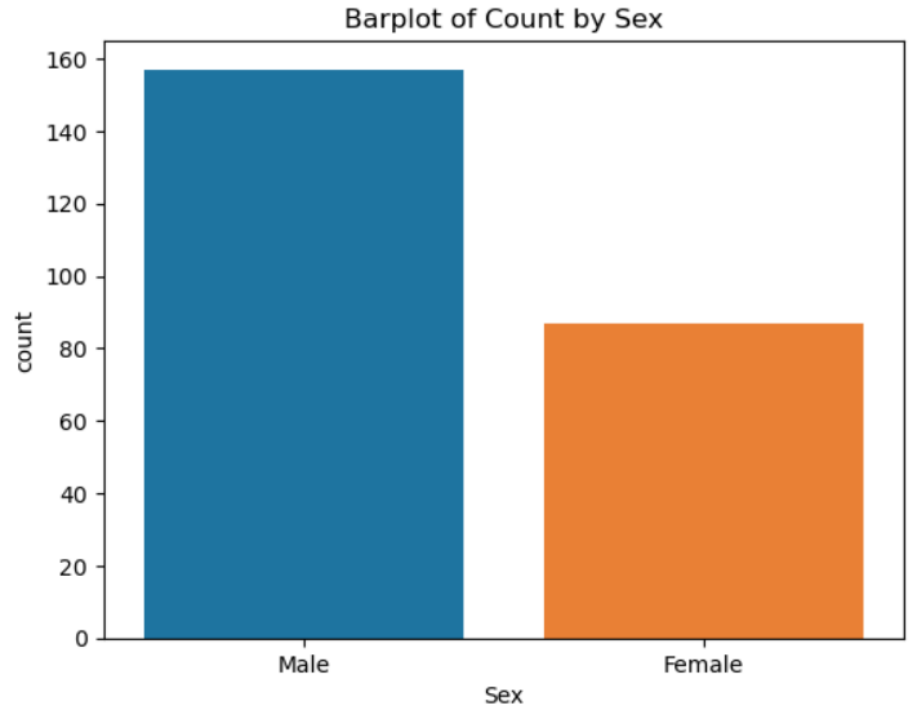
• Seaborn 라이브러리

> 기본 그래프

- 막대 barplot, countplot

```
sns.countplot(data = tips, x = 'sex')
plt.title('Barplot of Count by Sex')
```

Out :



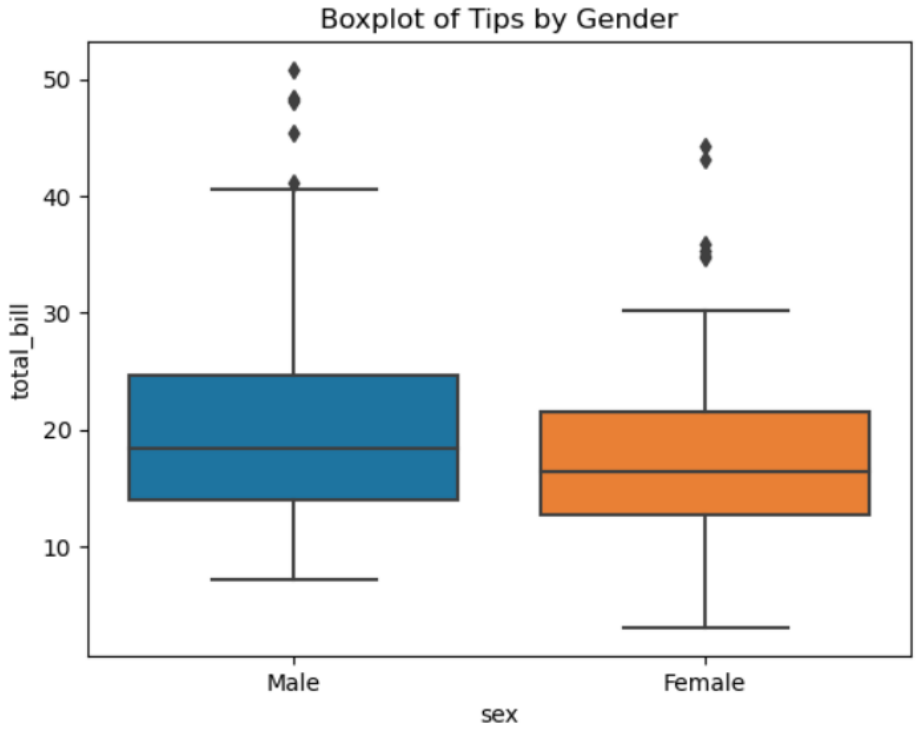
• Seaborn 라이브러리

> 기본 그래프

- 박스 boxplot

```
sns.boxplot(x = 'sex', y = 'total_bill', data = tips)
plt.title('Boxplot of Tips by Gender')
```

Out :



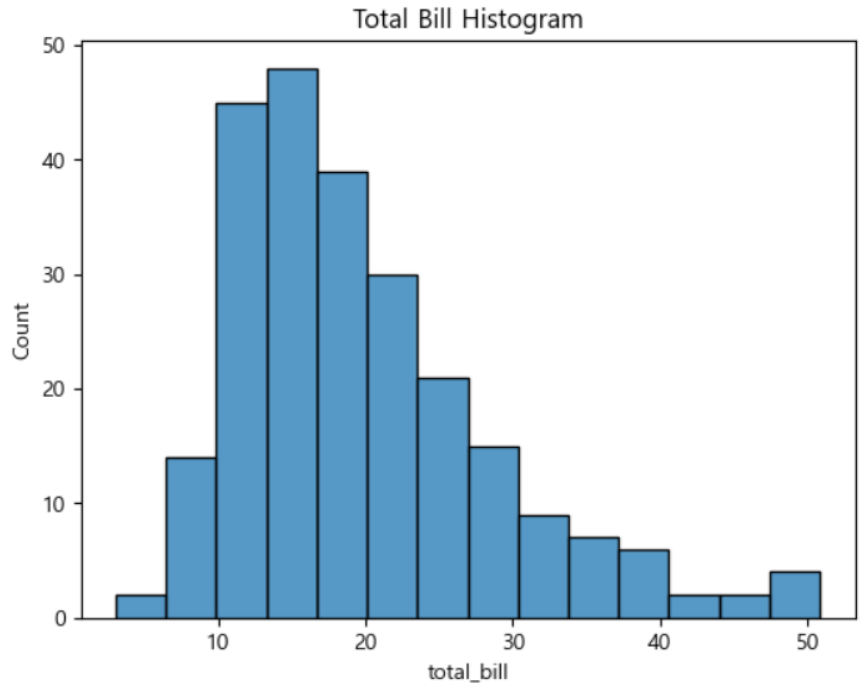
• Seaborn 라이브러리

> 기본 그래프

- histplot

```
sns.histplot(tips['total_bill'])
plt.title('Total Bill Histogram')
```

Out :



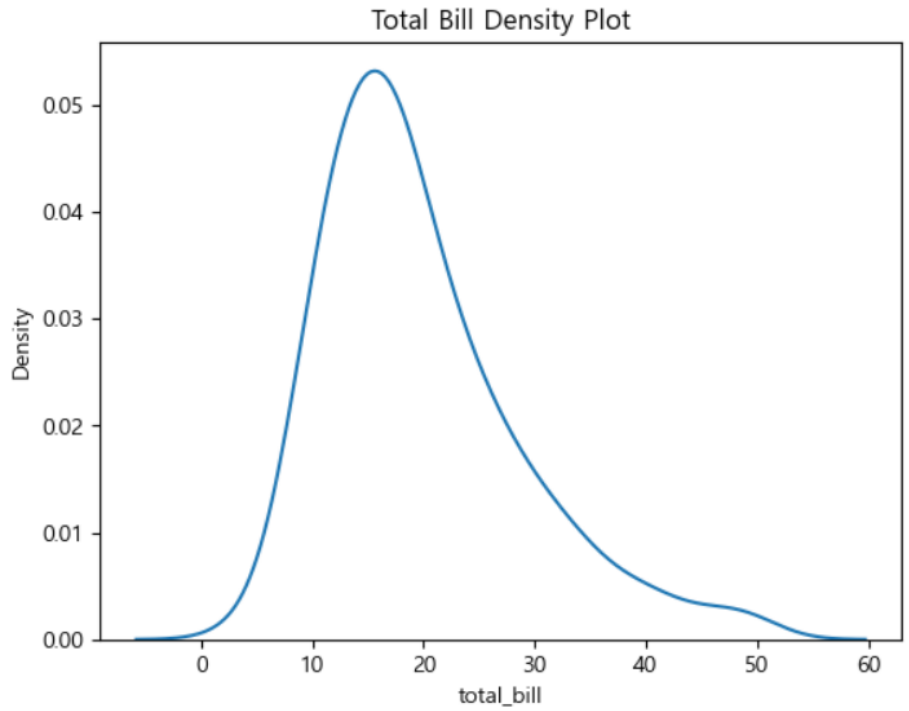
• Seaborn 라이브러리

> 새로운 그래프

- Density

```
sns.kdeplot(tips['total_bill']) # Kernel density estimation
plt.title('Total Bill Density Plot')
```

Out :



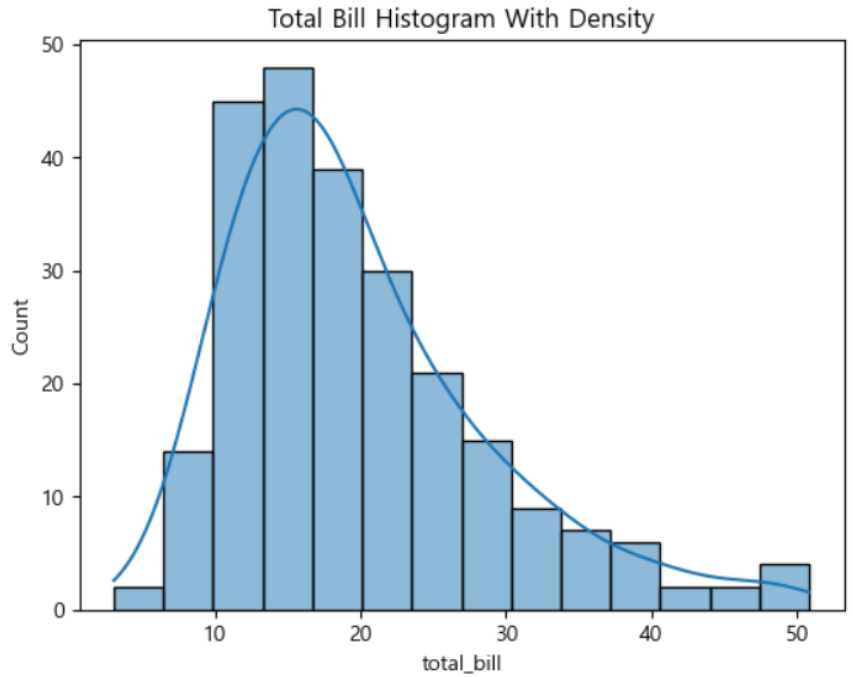
• Seaborn 라이브러리

> 새로운 그래프

- histplot(Histogram + Density)

```
sns.histplot(tips['total_bill'], kde=True)
plt.title('Total Bill Histogram With Density')
```

Out :



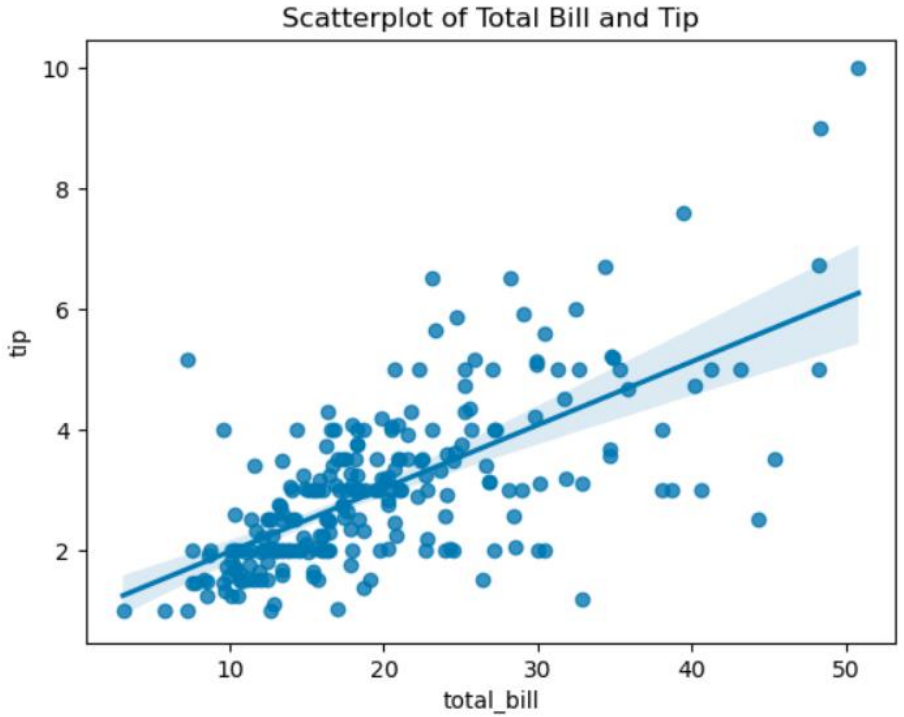
• Seaborn 라이브러리

> 새로운 그래프

- regplot(Scatter + Regression)

```
sns.regplot(x = 'total_bill', y = 'tip', data = tips)
plt.title('Scatterplot of Total Bill and Tip')
```

Out :



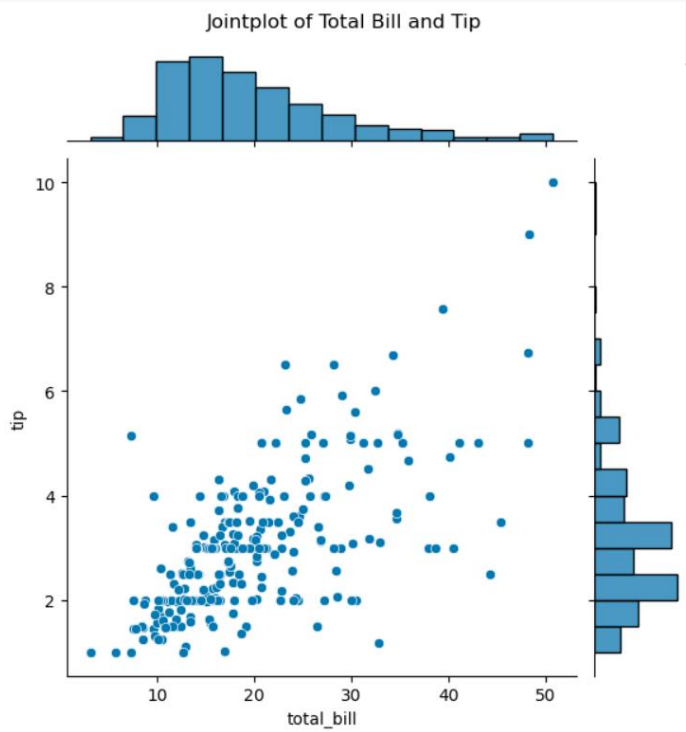
• Seaborn 라이브러리

> 새로운 그래프

- jointplot(Scatter + Histogram)

```
sns.jointplot(x = 'total_bill', y = 'tip', data = tips)
plt.suptitle('Jointplot of Total Bill and Tip')
plt.tight_layout()
```

Out :



그래프 그리기

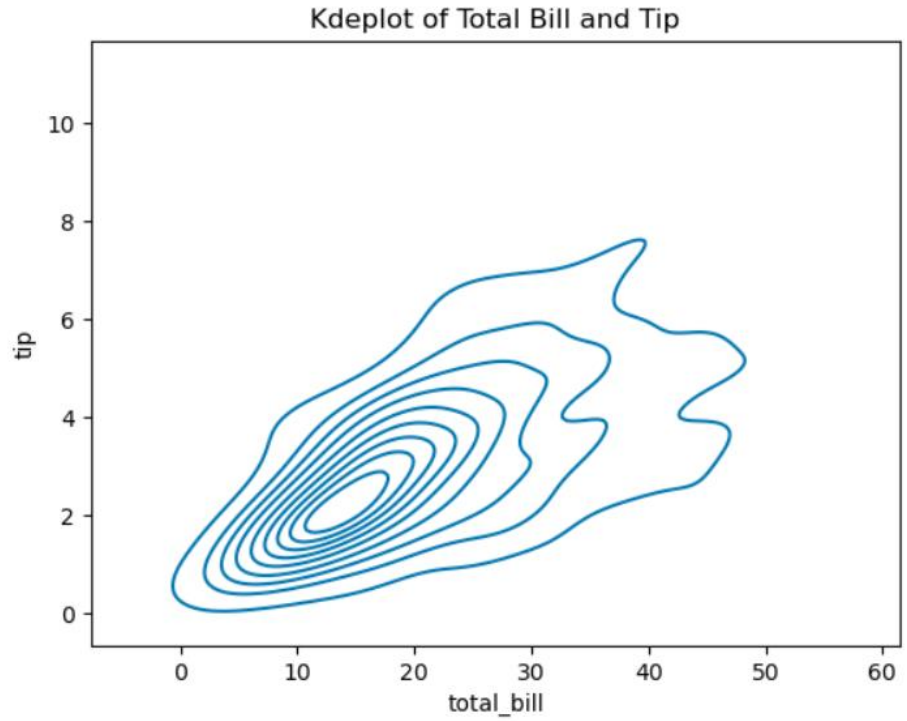
• Seaborn 라이브러리

> 새로운 그래프

- kdeplot

```
sns.kdeplot(x = 'total_bill', y = 'tip', data = tips)
plt.title('Kdeplot of Total Bill and Tip')
```

Out :



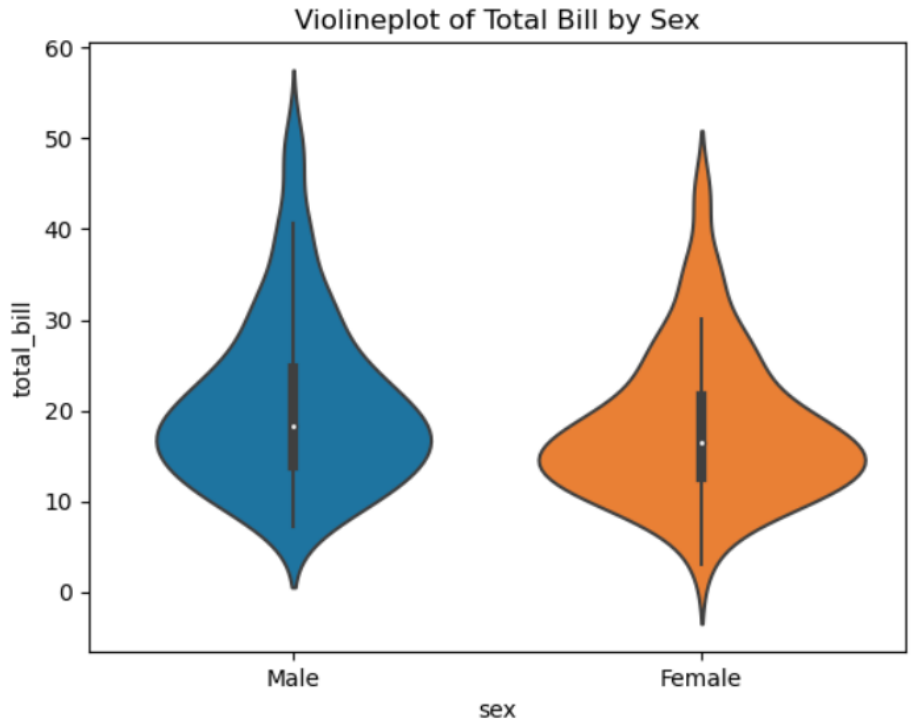
• Seaborn 라이브러리

> 새로운 그래프

- violinplot

```
sns.violinplot(x = 'sex', y = 'total_bill', data = tips)
plt.title('Violineplot of Total Bill by Sex')
```

Out :



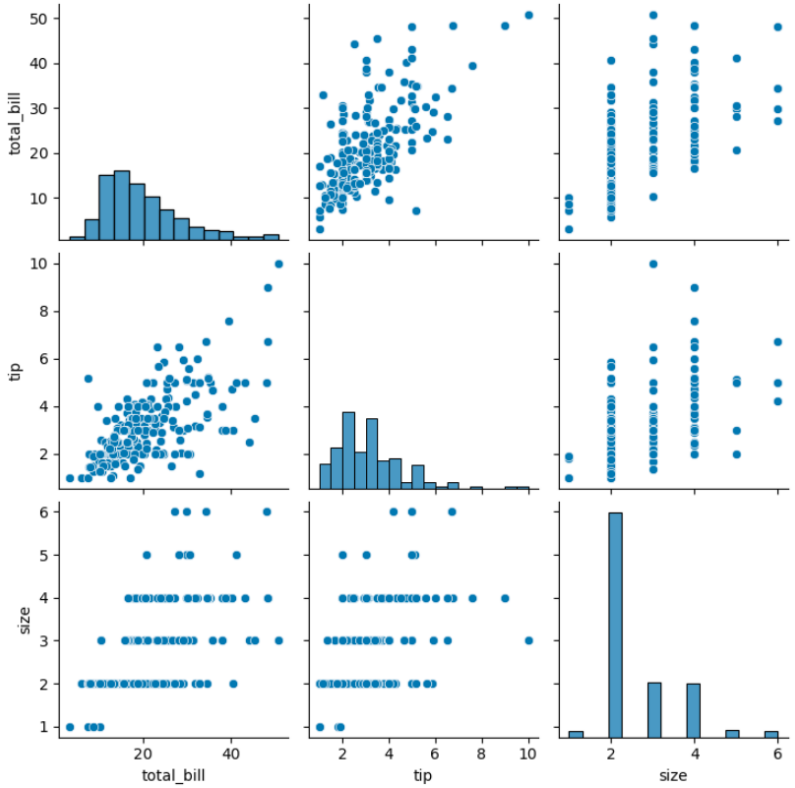
• Seaborn 라이브러리

> 새로운 그래프

- pairplot – 각 컬럼(숫자) 간의 히스토그램, 산점도 그래프

```
sns.pairplot(tips)
```

Out :



• Seaborn 라이브러리

> **새로운 그래프**

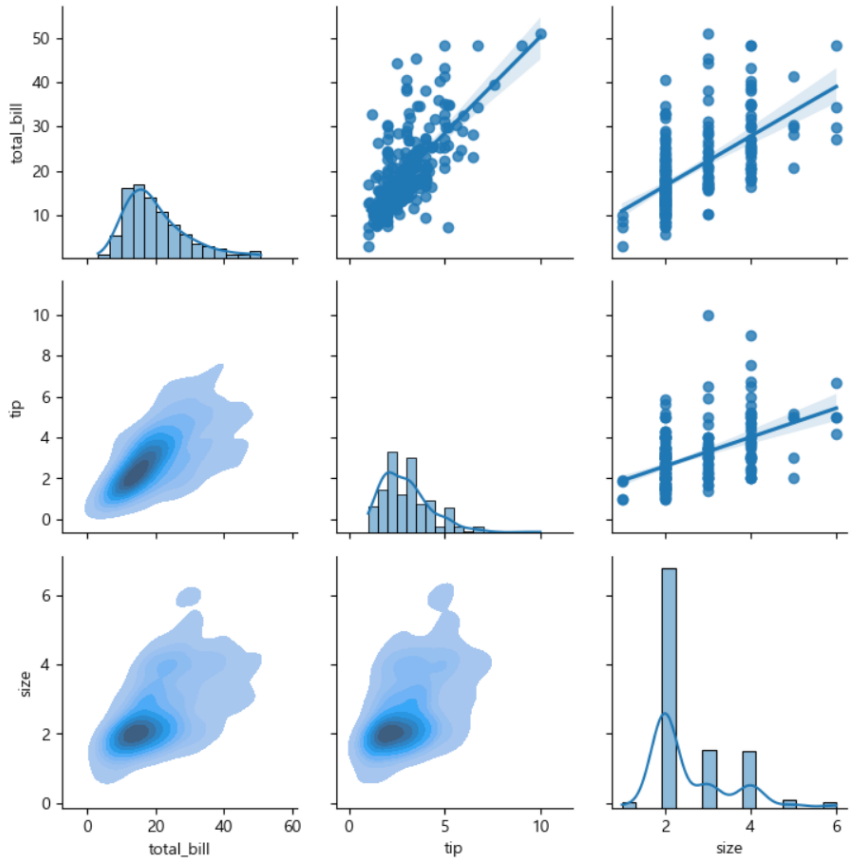
- pairplot – 각 컬럼(숫자) 간의 히스토그램, 산점도 그래프

```
pair_grid = sns.PairGrid(tips)
pair_grid = pair_grid.map_upper(sns.regplot)
pair_grid = pair_grid.map_lower(sns.kdeplot, fill = True)
pair_grid = pair_grid.map_diag(sns.histplot, kde=True)
```

• Seaborn 라이브러리

> 새로운 그래프

- pairplot – 각 컬럼(숫자) 간의 히스토그램, 산점도 그래프



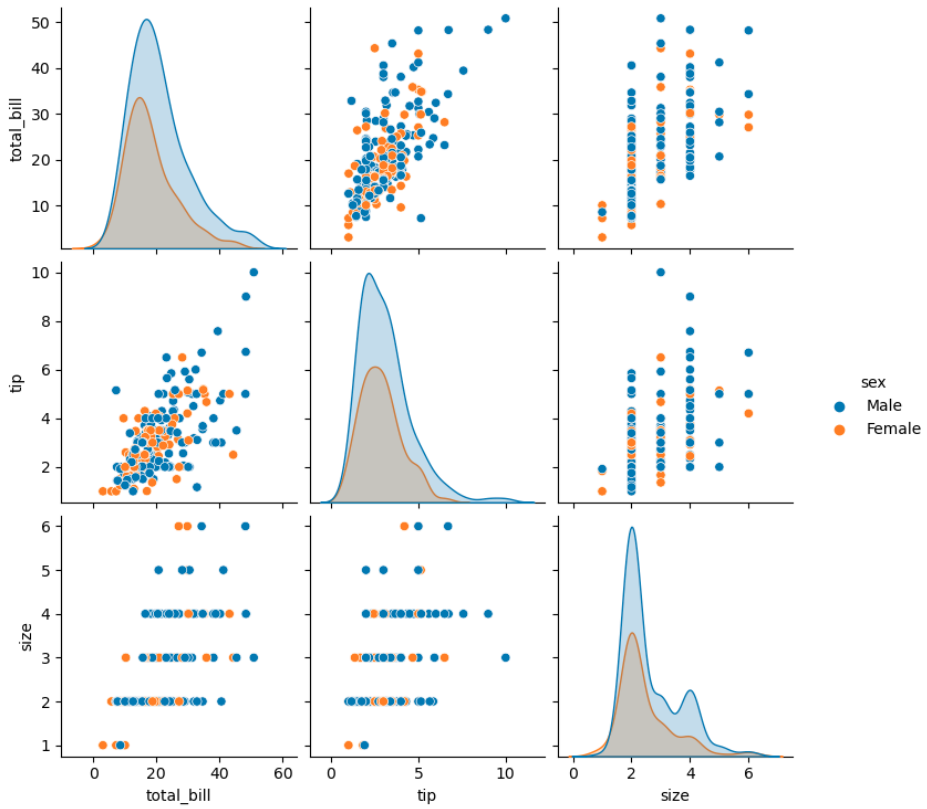
• Seaborn 라이브러리

> 새로운 그래프

- pairplot – 각 컬럼(숫자) 간의 히스토그램, 산점도 그래프

```
sns.pairplot(tips, hue='sex')
```

Out :



그래프 그리기

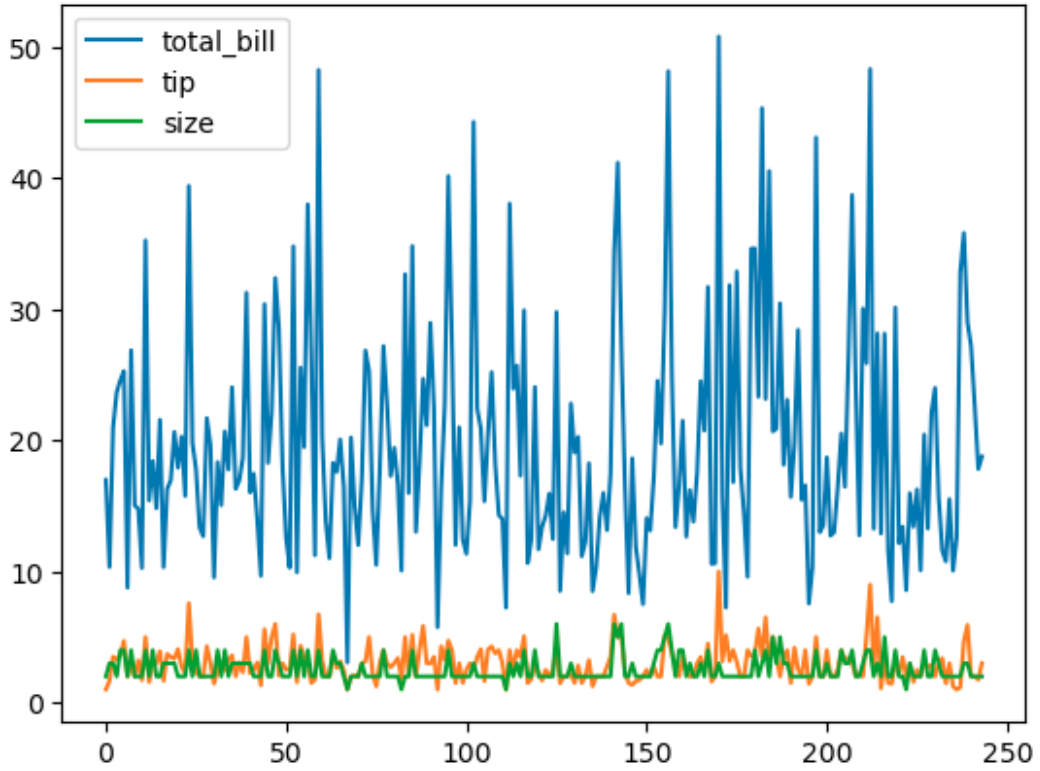
- Pandas 내장 그래프

> 판다스 내 그래프 함수를 사용

- df.plot()

```
tips.plot()
```

Out :



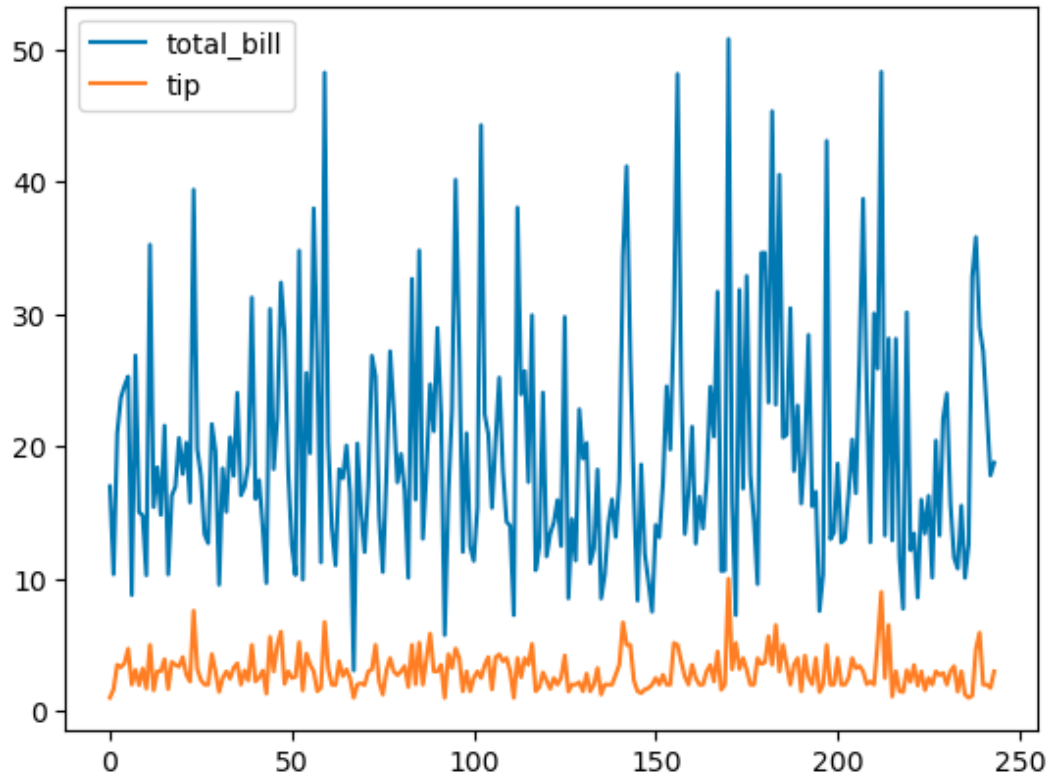
• Pandas 내장 그래프

> 판다스 내 그래프 함수를 사용

- df.plot()

```
tips.plot(y=['total_bill', 'tip'])
```

Out :



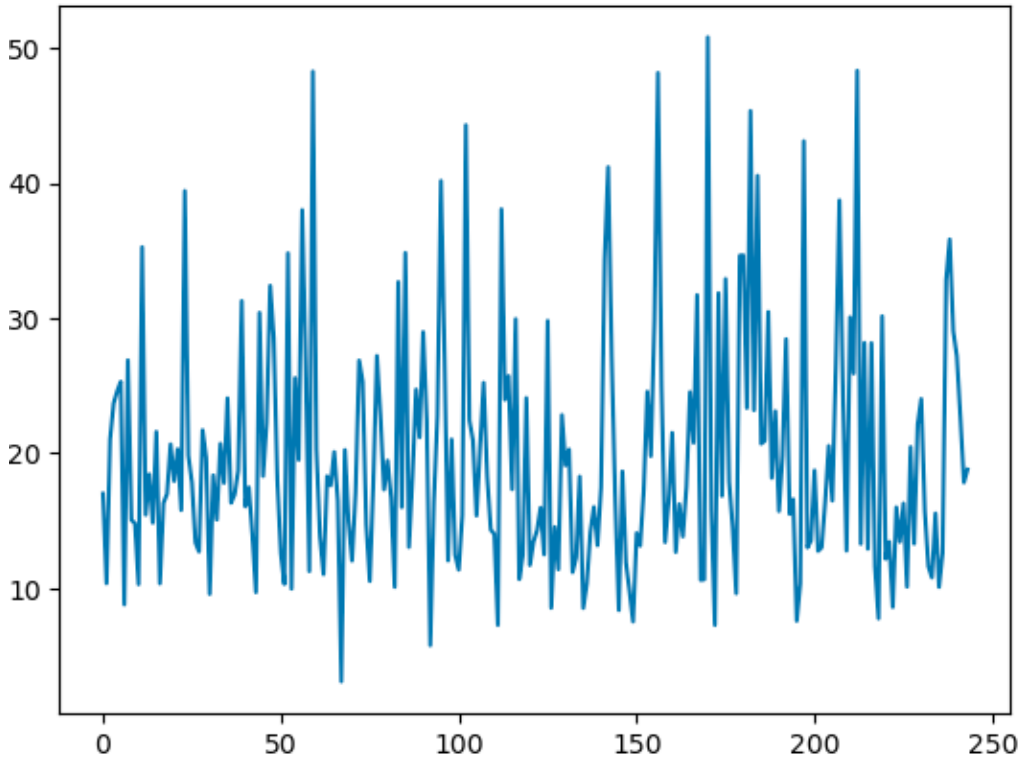
• Pandas 내장 그래프

> 판다스 내 그래프 함수를 사용

- df.plot()

```
tips['total_bill'].plot()
```

Out :



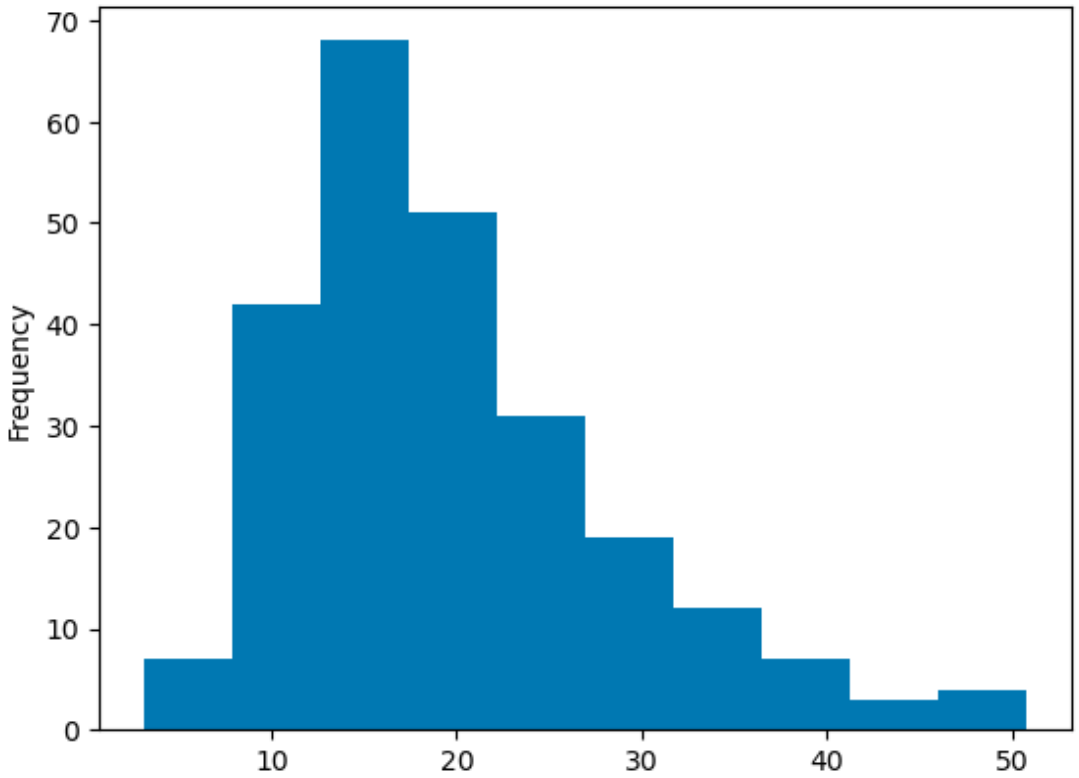
• Pandas 내장 그래프

> 판다스 내 그래프 함수를 사용

- df.plot()

```
tips['total_bill'].plot(kind = 'hist')
```

Out :



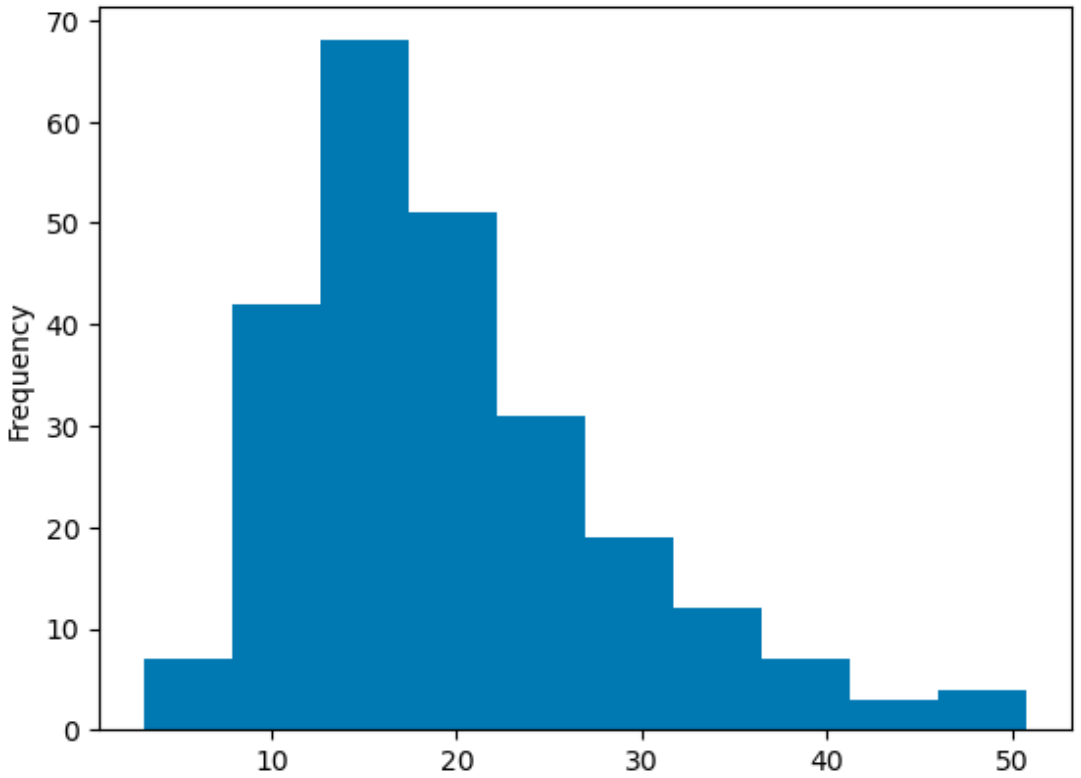
• Pandas 내장 그래프

> 판다스 내 그래프 함수를 사용

- df.plot

```
tips['total_bill'].plot.hist(bins = 10)
```

Out :



• Pandas 내장 그래프

> 판다스 내 그래프 함수를 사용

- df.plot

```
tips.plot.scatter(x='total_bill', y='tip')
```

Out :