

10

# 데이터 시각화

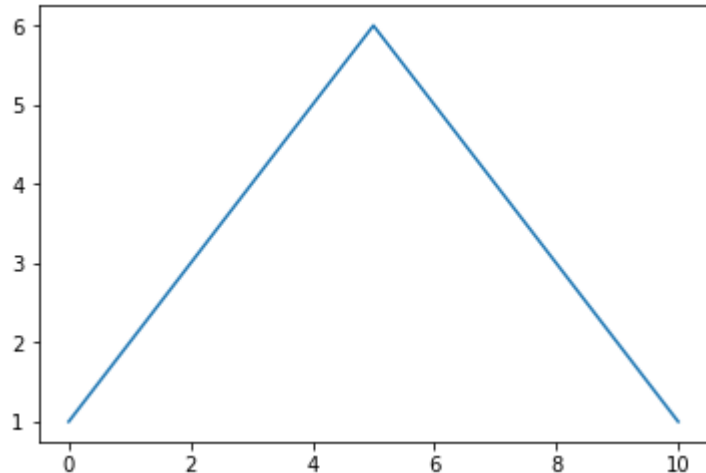
# Python 시각화

```
%matplotlib inline
```

```
s=[1,2,3,4,5,6,5,4,3,2,1]
```

```
s1=pd.Series(s)
```

```
s1.plot()
```



```
import matplotlib.pyplot as plt  
%matplotlib inline
```

```
s=[1,2,3,4,5,6,5,4,3,2,1]
```

```
plt.plot(s)
```

```
plt.show()
```

- % 명령은 Cell Magic 명령어
- 파이썬 코드가 아니라 Jupyter에게 특정 기능을 수행하도록 하는 명령
- %matplotlib inline 명령은 Jupyter에게 matplotlib 그래프를 출력 영역에 표시할 것을 지시하는 명령

# Pandas 내장함수 시각화

- 시리즈 또는 데이터프레임 객체에 plot() 메소드를 적용
- kind 옵션으로 그래프의 종류 선택

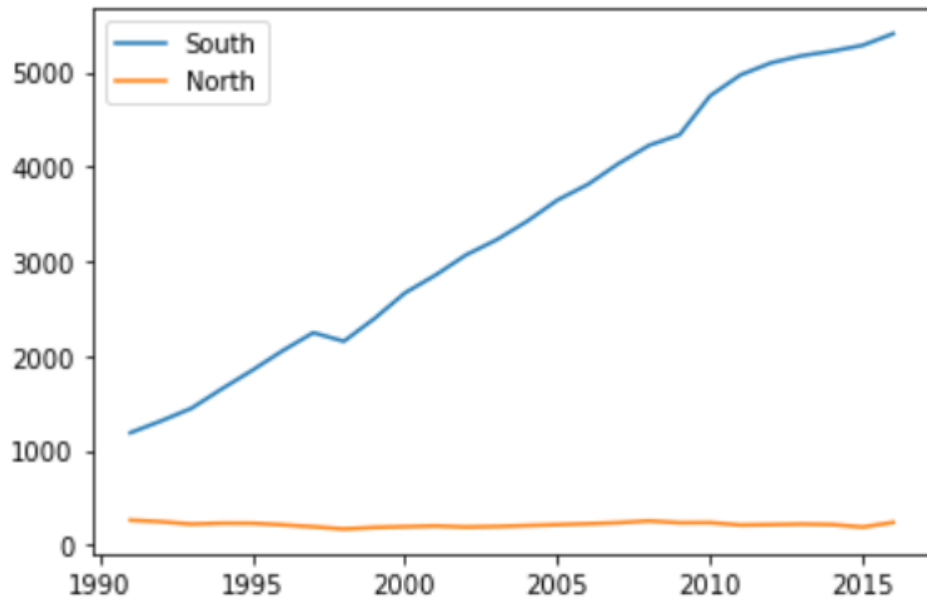
kind 옵션	설명	kind 옵션	설명
'line'	선 그래프	'kde'	커널 밀도 그래프
'bar'	수직 막대 그래프	'area'	면적 그래프
'barh'	수평 막대 그래프	'pie'	파이 그래프
'his'	히스토그램	'scatter'	산점도 그래프
'box'	박스 플롯	'hexbin'	고밀도 산점도 그래프

< Pandas 내장 함수 plot() 메소드 >

# Pandas 내장함수 시각화

## ■ 선그래프

DataFrame 객체.plot()



```
import pandas as pd
df = pd.read_excel('data/남북한발전전력량.xlsx')
# 남한, 북한 발전량 합계 데이터만 추출
df_ns = df.iloc[[0, 5], 3:]
# 행인덱스 변경
df_ns.index = ['South', 'North']
# 열이름의 자료형을 정수형으로 변경
df_ns.columns = df_ns.columns.map(int)
print(df_ns.head())
```

```
df_ns.plot() #선 그래프 그리기
```

```
tdf_ns = df_ns.T #행, 열 전치하여 다시 그리기
print(tdf_ns.head())
```

```
tdf_ns.plot()
```

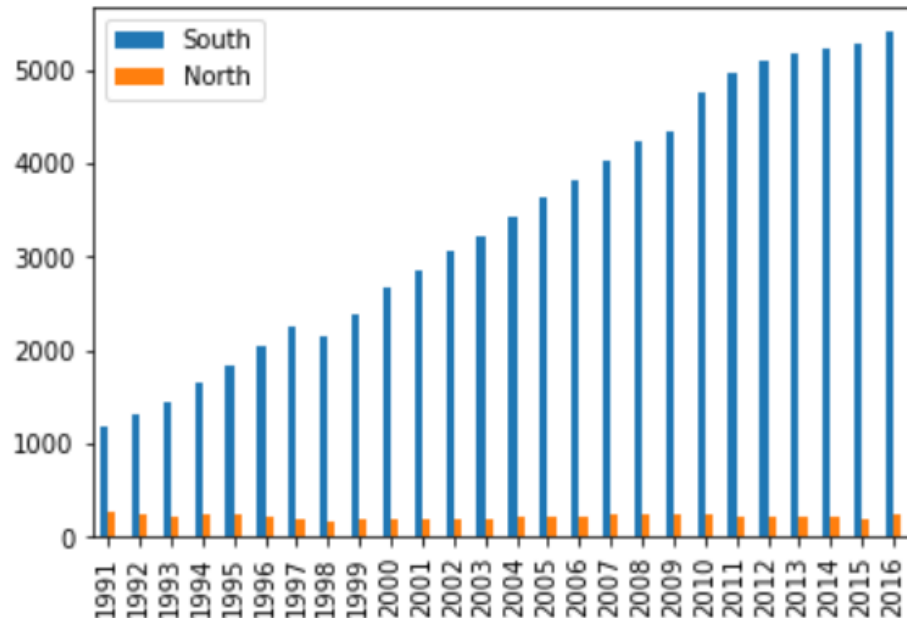
# Pandas 내장함수 시각화

## ■ 막대그래프

- kind 옵션에 그래프 종류 지정

DataFrame 객체.plot(kind='bar')

DataFrame 객체.plot(kind='barh')



```
import pandas as pd
```

```
df = pd.read_excel('data/남북한발전전력량.xlsx')
```

```
# 남한, 북한 발전량 합계 데이터만 추출
```

```
df_ns = df.iloc[[0, 5], 3:]
```

```
# 행 인덱스 변경
```

```
df_ns.index = ['South', 'North']
```

```
# 열 이름의 자료형을 정수형으로 변경
```

```
df_ns.columns = df_ns.columns.map(int)
```

```
# 행, 열 전치하여 막대 그래프 그리기
```

```
tdf_ns = df_ns.T
```

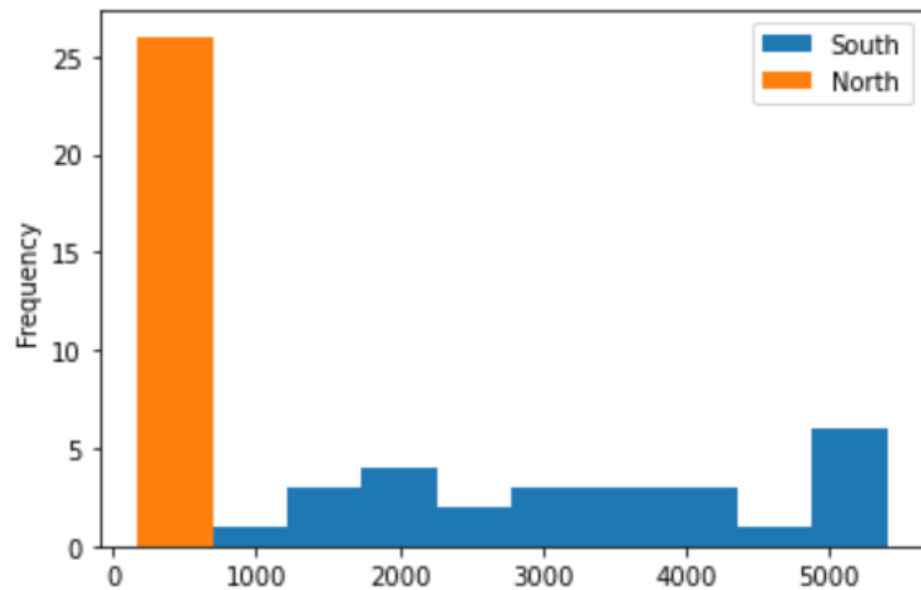
```
tdf_ns.plot(kind='bar')
```

```
tdf_ns.plot(kind='barh')
```

# Pandas 내장함수 시각화

## ■ 히스토그램

DataFrame 객체.plot(kind='hist')



```
import pandas as pd
```

```
df = pd.read_excel('data/남북한발전전력량.xlsx')
```

```
# 남한, 북한 발전량 합계 데이터만 추출
```

```
df_ns = df.iloc[[0, 5], 3:]
```

```
# 행 인덱스 변경
```

```
df_ns.index = ['South', 'North']
```

```
# 열 이름의 자료형을 정수형으로 변경
```

```
df_ns.columns = df_ns.columns.map(int)
```

```
# 행, 열 전치하여 막대 그래프 그리기
```

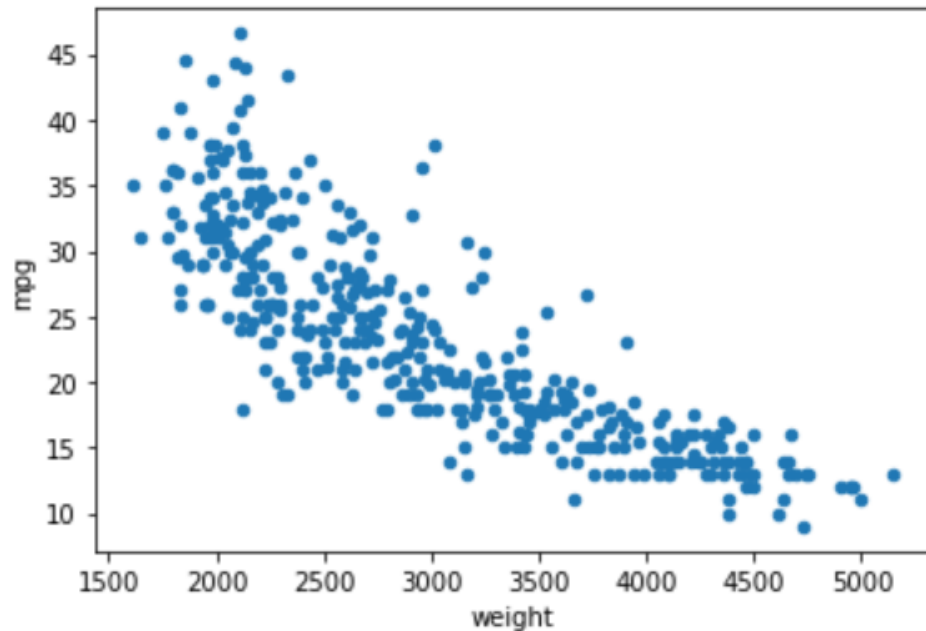
```
tdf_ns = df_ns.T
```

```
tdf_ns.plot(kind='hist')
```

# Pandas 내장함수 시각화

## ■ 산점도

DataFrame 객체.plot(x,y,kind='scatter')



```
import pandas as pd
```

```
df = pd.read_csv('data/auto-mpg.csv', header=None)
```

```
df.columns = ['mpg', 'cylinders', 'displacement', 'horsepower',  
              'weight', 'acceleration', 'model year', 'origin',  
              'name']
```

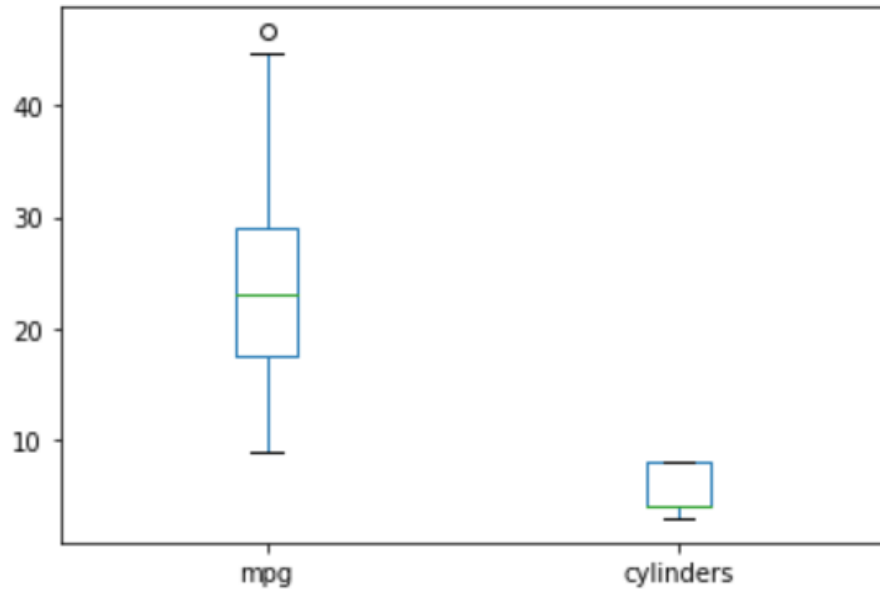
```
# 2개의 열을 선택하여 산점도 그리기
```

```
df.plot(x='weight', y='mpg', kind='scatter')
```

# Pandas 내장함수 시각화

## ■ 박스플롯

DataFrame 객체.plot(kind='box')



```
import pandas as pd
```

```
df = pd.read_csv('data/auto-mpg.csv', header=None)
```

```
df.columns = ['mpg', 'cylinders', 'displacement', 'horsepower',  
              'weight', 'acceleration', 'model year', 'origin',  
              'name']
```

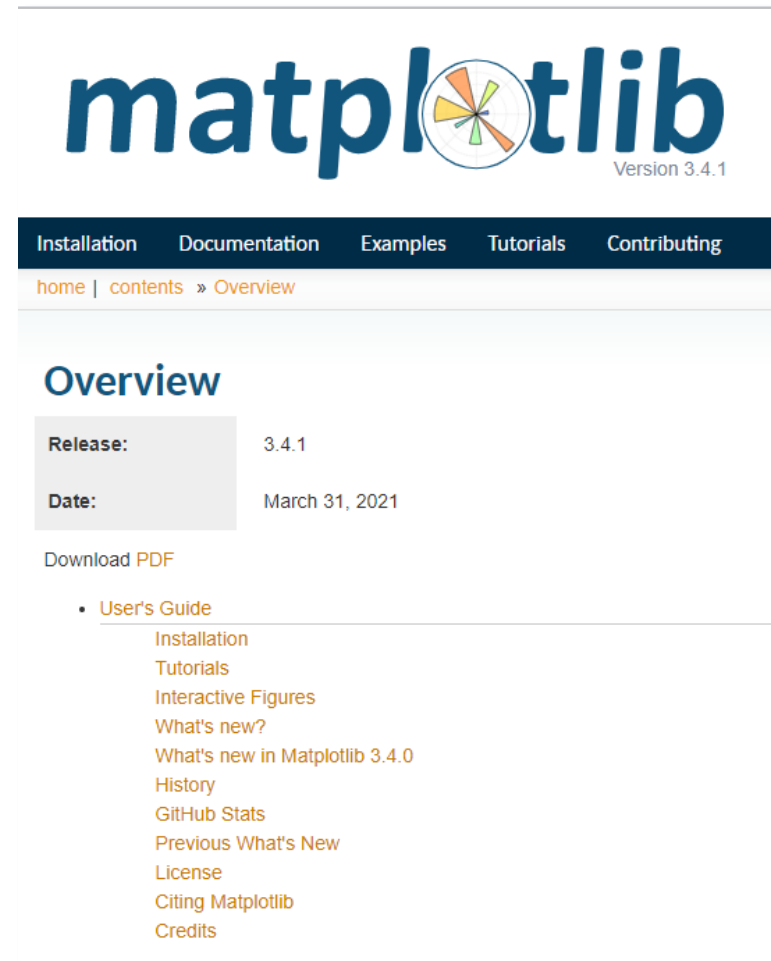
```
# 열을 선택하여 박스 플롯 그리기
```

```
df[['mpg', 'cylinders']].plot(kind='box')
```



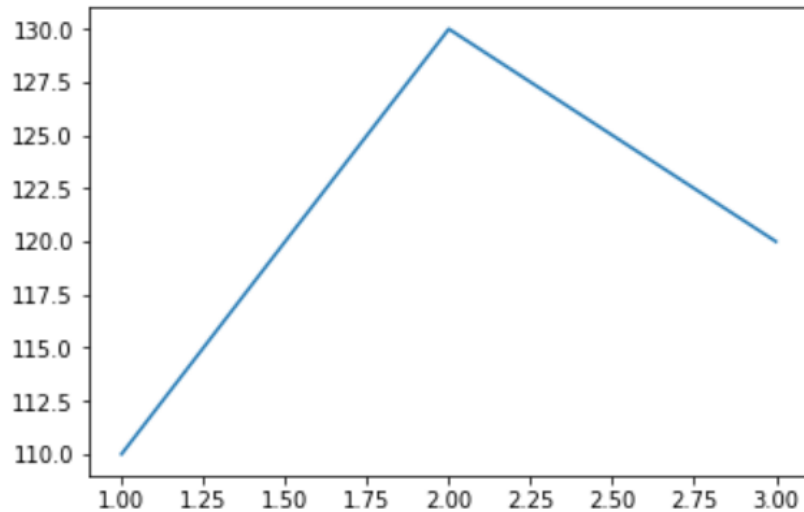
# Matplotlib 개요

- 파이썬에서 데이터를 차트나 플롯(Plot)으로 그려주는 라이브러리 패키지
- 가장 많이 사용되는 데이터 시각화(Data Visualization) 패키지
- 라인 플롯, 바 차트, 파이차트, 히스토그램, Box Plot, Scatter Plot 등을 비롯하여 다양한 차트와 플롯 스타일을 지원



# Matplotlib 기본 사용법

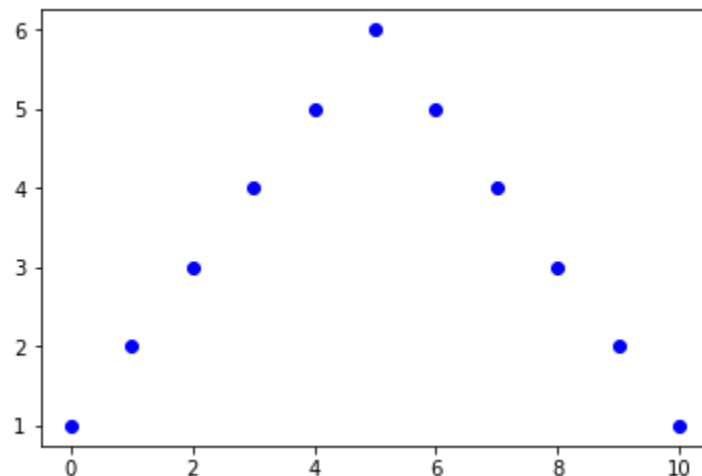
```
plt.plot([1,2,3], [110,130,120])  
plt.show()
```



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

```
%matplotlib inline
```

```
plt.figure()  
plt.plot([1,2,3,4,5,6,5,4,3,2,1], "ob")  
plt.show()
```



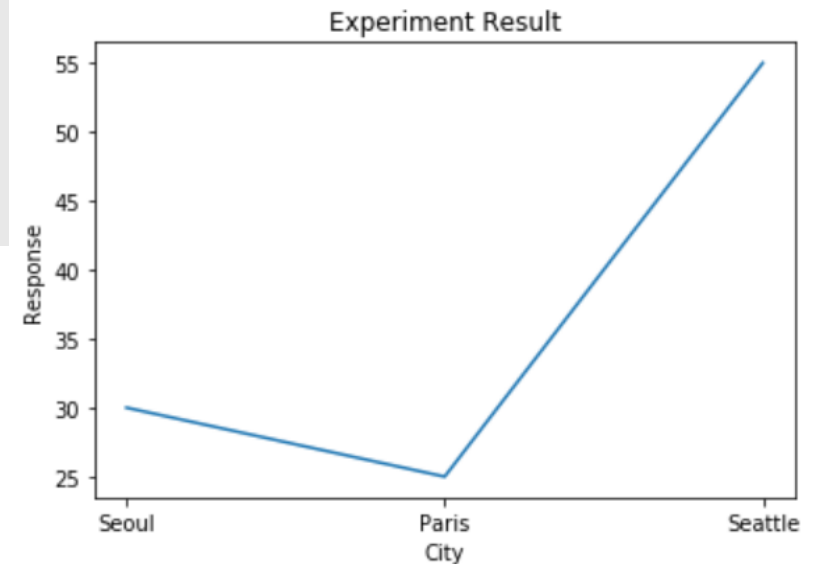
# Matplotlib 기본 사용법

---

## ■ 제목과 축 레이블

- 플롯에 X,Y 축 레이블이나 제목을 붙이기 위해서는 plt.xlabel(축이름), plt.ylabel(축이름), plt.title(제목) 등의 함수를 사용

```
plt.plot(["Seoul", "Paris", "Seattle"], [30, 25, 55])  
plt.xlabel('City')  
plt.ylabel('Response')  
plt.title('Experiment Result')  
plt.show()
```

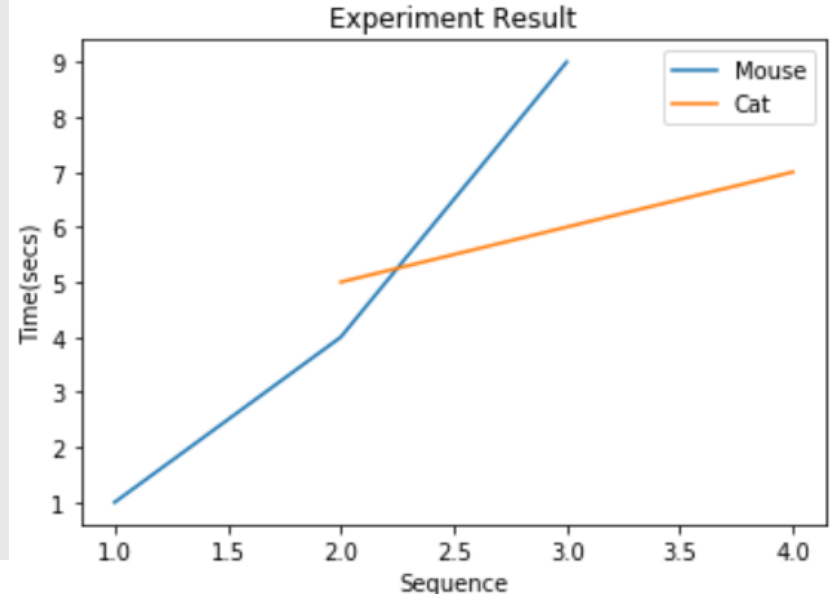


# Matplotlib 기본 사용법

## ■ 범례 추가

- 플롯에 여러 개의 라인들을 추가하기 위해서는 plt.plot()을 plt.show() 이전에 여러 번 사용
- 각 라인에 대한 범례를 추가하기 위해서는 plt.legend([라인1범례, 라인2범례]) 함수를 사용하여 각 라인에 대한 범례를 순서대로 지정

```
plt.plot([1,2,3], [1,4,9])  
plt.plot([2,3,4],[5,6,7])  
plt.xlabel('Sequence')  
plt.ylabel('Time(secs)')  
plt.title('Experiment Result')  
plt.legend(['Mouse', 'Cat'])  
plt.show()
```



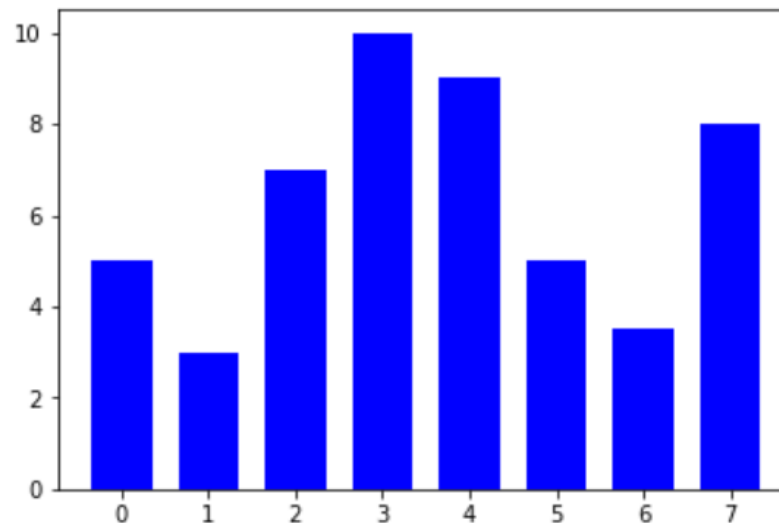
# Matplotlib 기본 사용법

---

## ■ 다양한 차트 및 플롯

- Bar 차트를 그리기 위해서는 plt.bar() 함수
- Pie 차트를 그리기 위해서는 plt.pie() 함수
- 히스토그램을 그리기 위해선 plt.hist() 함수

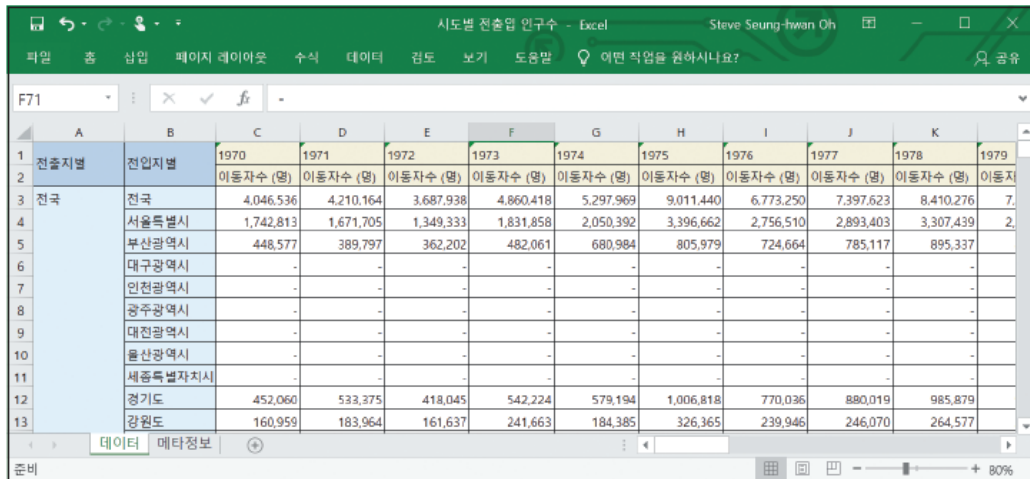
```
y = [5, 3, 7, 10, 9, 5, 3.5, 8]
x = range(len(y))
plt.bar(x, y, width=0.7, color="blue")
plt.show()
```



# Matplotlib 실습 예제

## ■ 선그래프

```
import matplotlib.pyplot as plt
plt.plot()
```



The screenshot shows an Excel spreadsheet titled '시도별 전출입 인구수 - Excel'. The data is organized in columns for years (1970-1979) and rows for regions. The first two columns are '전출지별' (Origin Region) and '전입지별' (Destination Region). The subsequent columns show population counts for each year, with sub-headers like '이동자수 (명)' (Number of Movers). The regions listed include 전국 (National), 서울특별시 (Seoul), 부산광역시 (Busan), 대구광역시 (Daegu), 인천광역시 (Incheon), 광주광역시 (Gwangju), 대전광역시 (Daejeon), 울산광역시 (Ulsan), 세종특별자치시 (Sejong), 경기도 (Gyeonggi), and 강원도 (Gangwon). The data shows a general trend of population movement from rural areas to urban centers like Seoul and Gyeonggi over the decade.

전출지별	전입지별	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979
전국	전국	4,046,536	4,210,164	3,687,938	4,860,418	5,297,969	9,011,440	6,773,250	7,397,623	8,410,276	7,...
	서울특별시	1,742,813	1,671,705	1,349,333	1,831,858	2,050,392	3,396,662	2,756,510	2,893,403	3,307,439	2,...
	부산광역시	448,577	389,797	362,202	482,061	680,984	805,979	724,664	785,117	895,337	
	대구광역시	-	-	-	-	-	-	-	-	-	-
	인천광역시	-	-	-	-	-	-	-	-	-	-
	광주광역시	-	-	-	-	-	-	-	-	-	-
	대전광역시	-	-	-	-	-	-	-	-	-	-
	울산광역시	-	-	-	-	-	-	-	-	-	-
	세종특별자치시	-	-	-	-	-	-	-	-	-	-
	경기도	452,060	533,375	418,045	542,224	578,194	1,006,818	770,036	880,019	985,879	
	강원도	160,959	183,964	161,637	241,663	184,385	326,365	239,946	246,070	264,577	

< 시도별 전출입 인구수(시도간 인구 이동)>

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
df = pd.read_excel('data/시도별 전출입 인구수.xlsx')
df = df.fillna(method='ffill') #누락값(NaN)을 앞 데이터로 채움
```

```
# 서울에서 다른 지역으로 이동한 데이터만 추출하여 정리
mask = (df['전출지별'] == '서울특별시') & (df['전입지별'] != '서울특별시')
df_seoul = df[mask]
df_seoul = df_seoul.drop(['전출지별'], axis=1)
df_seoul.rename({'전입지별': '전입지'}, axis=1, inplace=True)
df_seoul.set_index('전입지', inplace=True)
```

```
# 서울에서 경기도로 이동한 인구 데이터 값만 선택
sr_one = df_seoul.loc['경기도']
```

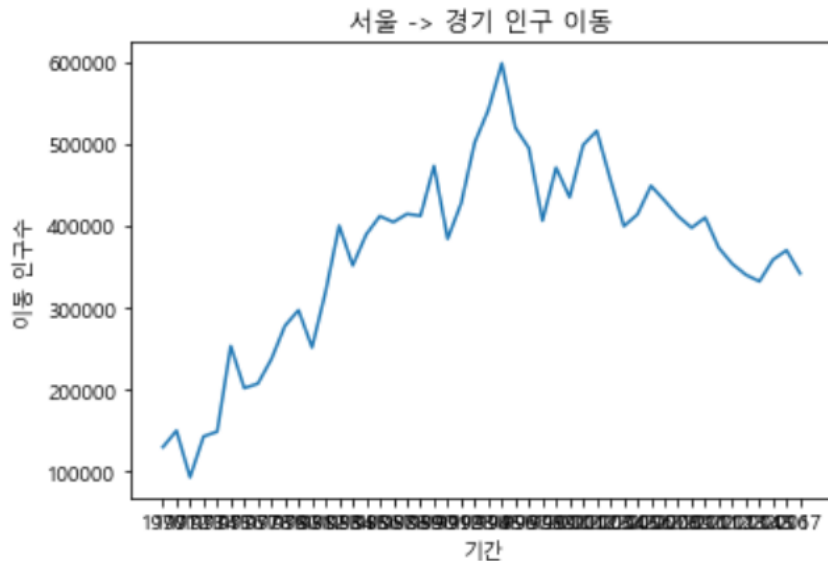
```
plt.plot(sr_one.index, sr_one.values) # x, y축 데이터를 plot 함수에 입력
```

```
plt.plot(sr_one) # 판다스 객체를 plot 함수에 입력
```

# Matplotlib 실습 예제

- 차트 제목, 축 이름 추가

제목 : title()  
x축 이름 : xlabel()  
y축 이름 : ylabel()



```
plt.plot(sr_one) # 판다스 객체를 plot 함수에 입력
```

```
# 차트 제목 추가
```

```
plt.title('서울 -> 경기 인구 이동')
```

```
# 축이름 추가
```

```
plt.xlabel('기간')
```

```
plt.ylabel('이동 인구수')
```

```
plt.show() # 변경사항 저장하고 그래프 출력
```

```
# matplotlib 한글 폰트 오류 문제 해결
```

```
from matplotlib import font_manager, rc
```

```
font_path = "c:/Windows/Fonts/malgun.ttf" #폰트파일의 위치
```

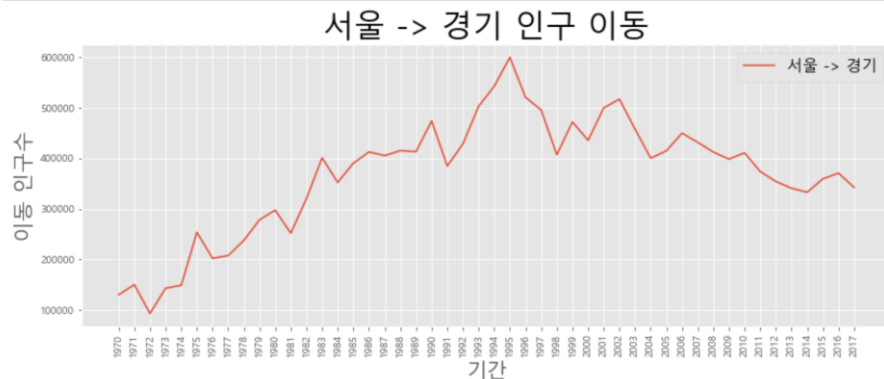
```
font_name = font_manager.FontProperties(fname=font_path).get_name()
```

```
rc('font', family=font_name)
```

# Matplotlib 실습 예제

## ■ 그래프 꾸미기

style.use() : 스타일 서식  
size 속성 : 크기  
marker : 마커



```
sr_one = df_seoul.loc['경기도']
```

```
plt.style.use('ggplot') # 스타일 서식 지정
```

```
plt.figure(figsize=(14, 5))
```

```
plt.xticks(size=10, rotation='vertical')
```

```
# 마커 표시 추가
```

```
plt.plot(sr_one, marker='o', markersize=10)
```

```
plt.title('서울 -> 경기 인구 이동', size=30) #차트 제목
```

```
plt.xlabel('기간', size=20) #x축 이름
```

```
plt.ylabel('이동 인구수', size=20) #y축 이름
```

```
plt.legend(labels=['서울 -> 경기'], loc='best', fontsize=15)
```

```
plt.show()
```



# Matplotlib 실습 예제

## ■ 스타일 리스트

# y축 범위 지정 (최소값, 최대값)

```
plt.ylim(50000, 800000)
```

# 주식 표시 - 화살표

```
plt.annotate('',  
             xy=(20, 620000), #화살표의 머리 부분(끝점)  
             xytext=(2, 290000), #화살표의 꼬리 부분(시작점)  
             xycoords='data',      #좌표체계  
             arrowprops=dict(arrowstyle='->', color='skyblue', lw=5), #화살표 서식  
             )
```

```
plt.annotate('',  
             xy=(47, 450000),      #화살표의 머리 부분(끝점)  
             xytext=(30, 580000), #화살표의 꼬리 부분(시작점)  
             xycoords='data',      #좌표체계  
             arrowprops=dict(arrowstyle='->', color='olive', lw=5), #화살표 서식  
             )
```

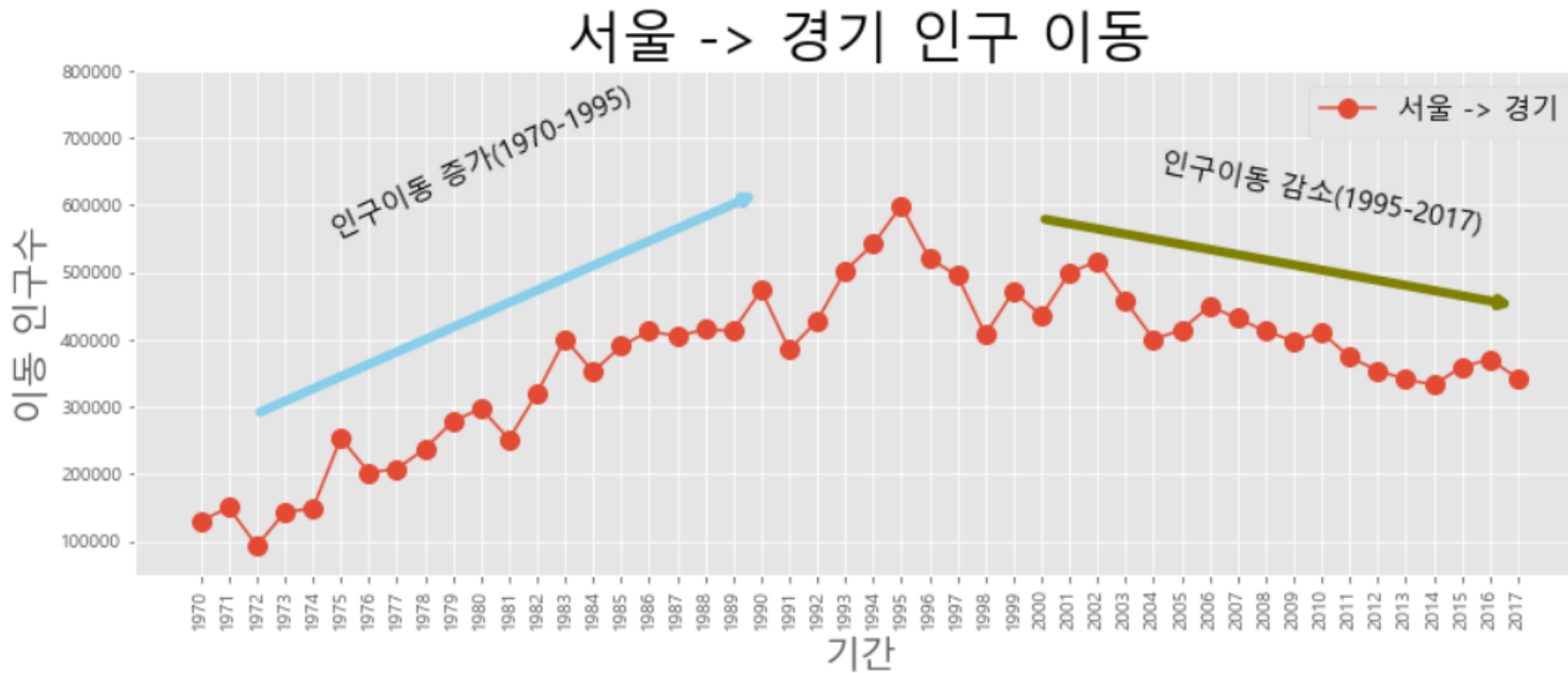
# 주식 표시 - 텍스트

```
plt.annotate('인구이동 증가(1970-1995)', #텍스트 입력  
            xy=(10, 550000),             #텍스트 위치 기준점  
            rotation=25,                  #텍스트 회전각도  
            va='baseline',                #텍스트 상하 정렬  
            ha='center',                  #텍스트 좌우 정렬  
            fontsize=15,                  #텍스트 크기  
            )
```

```
plt.annotate('인구이동 감소(1995-2017)', #텍스트 입력  
            xy=(40, 560000),             #텍스트 위치 기준점  
            rotation=-11,                 #텍스트 회전각도  
            va='baseline',                #텍스트 상하 정렬  
            ha='center',                  #텍스트 좌우 정렬  
            fontsize=15,                  #텍스트 크기  
            )
```

```
plt.show() # 변경사항 저장하고 그래프 출력
```

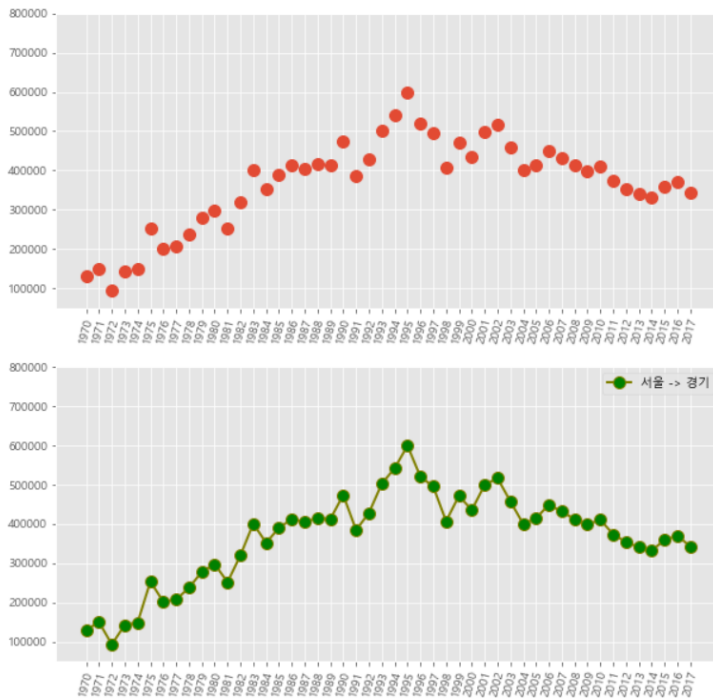
# Matplotlib 실습 예제



# Matplotlib 실습 예제

## ■ 그래프 여러 개 그리기

subplot()



```
fig = plt.figure(figsize=(10, 10))
ax1 = fig.add_subplot(2, 1, 1)
ax2 = fig.add_subplot(2, 1, 2)
```

```
# axe 객체에 plot 함수로 그래프 출력
ax1.plot(sr_one, 'o', markersize=10)
ax2.plot(sr_one, marker='o', markerfacecolor='green',
        markersize=10, color='olive', linewidth=2,
        label='서울 -> 경기')
ax2.legend(loc='best')
```

```
#y축 범위 지정 (최소값, 최대값)
ax1.set_ylim(50000, 800000)
ax2.set_ylim(50000, 800000)
```

```
# 축 눈금 라벨 지정 및 75도 회전
ax1.set_xticklabels(sr_one.index, rotation=75)
ax2.set_xticklabels(sr_one.index, rotation=75)
```

```
plt.show() # 변경사항 저장하고 그래프 출력
```

# Matplotlib 실습 예제

꾸미기 옵션	설명
'o'	선 그래프가 아니라 점 그래프로 표현
marker='o'	마커 모양 (예: 'o', '+', '*', '.')
markerfacecolor='green'	마커 배경색
markersize=10	마커 크기
color='olive'	선의 색
linewidth=2	선의 두께
label='서울 -> 경기'	라벨 지정

< 선그래프 꾸미기 옵션 >

## ■ color

- an RGB or RGBA (red, green, blue, alpha) tuple of float values in closed interval  $[0, 1]$  (e.g.,  $(0.1, 0.2, 0.5)$  or  $(0.1, 0.2, 0.5, 0.3)$ );
- a hex RGB or RGBA string (e.g., '#0f0f0f' or '#0f0f0f80'; case-insensitive);
- a shorthand hex RGB or RGBA string, equivalent to the hex RGB or RGBA string obtained by duplicating each character, (e.g., '#abc', equivalent to '#aabbcc', or '#abcd', equivalent to '#aabbccdd'; case-insensitive);
- a string representation of a float value in  $[0, 1]$  inclusive for gray level (e.g., '0.5');
- one of the characters {'b', 'g', 'r', 'c', 'm', 'y', 'k', 'w'}, which are short-hand notations for shades of blue, green, red, cyan, magenta, yellow, black, and white. Note that the colors 'g', 'c', 'm', 'y' do not coincide with the X11/CSS4 colors. Their particular shades were chosen for better visibility of colored lines against typical backgrounds.
- a X11/CSS4 color name (case-insensitive);
- a name from the **xkcd color survey**, prefixed with 'xkcd:' (e.g., 'xkcd:sky blue'; case insensitive);
- one of the Tableau Colors from the 'T10' categorical palette (the default color cycle): {'tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'tab:purple', 'tab:brown', 'tab:pink', 'tab:gray', 'tab:olive', 'tab:cyan'} (case-insensitive);
- a "CN" color spec, i.e. 'C' followed by a number, which is an index into the default property cycle (`rcParams["axes.prop_cycle"]`) (default: `cycler('color', ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf'])`); the indexing is intended to occur at rendering time, and defaults to black if the cycle does not include color.

## ■ marker

marker	symbol	description
"."	•	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	⋿	tri_down
"2"	⋈	tri_up
"3"	↙	tri_left
"4"	↘	tri_right
"8"	⬢	octagon
"s"	■	square
"p"	⬠	pentagon
"P"	⊕	plus (filled)
"*"	★	star
"h"	⬡	hexagon1
"H"	⬢	hexagon2
"+"	+	plus
"x"	×	x
"X"	⊗	x (filled)
"D"	◆	diamond
"d"	◊	thin_diamond
" "		vline

# Matplotlib 실습 예제

```
# 서울에서 '충청남도', '경상북도', '강원도', '전라남도'로 이동한 인구 데이터 값만 선택
col_years = list(map(str, range(1970, 2018)))
df_4 = df_seoul.loc[['충청남도', '경상북도', '강원도', '전라남도'], col_years]

plt.style.use('ggplot') # 스타일 서식 지정

# 그래프 객체 생성 (figure에 1개의 서브 플롯을 생성)
fig = plt.figure(figsize=(20, 10))
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)
# ax 객체에 plot 함수로 그래프 출력
ax1.plot(col_years, df_4.loc['충청남도',:], marker='o', markerfacecolor='green',
         markersize=10, color='olive', linewidth=2, label='서울 -> 충남')
ax2.plot(col_years, df_4.loc['경상북도',:], marker='o', markerfacecolor='blue',
         markersize=10, color='skyblue', linewidth=2, label='서울 -> 경북')
ax3.plot(col_years, df_4.loc['강원도',:], marker='o', markerfacecolor='red',
         markersize=10, color='magenta', linewidth=2, label='서울 -> 강원')
ax4.plot(col_years, df_4.loc['전라남도',:], marker='o', markerfacecolor='orange',
         markersize=10, color='yellow', linewidth=2, label='서울 -> 전남')
```

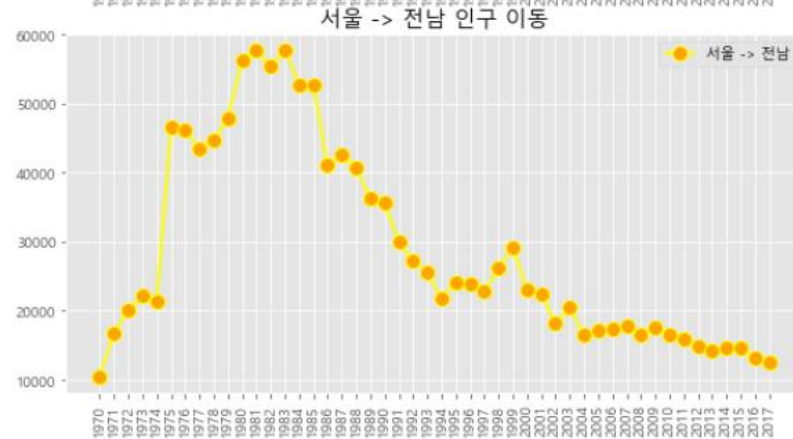
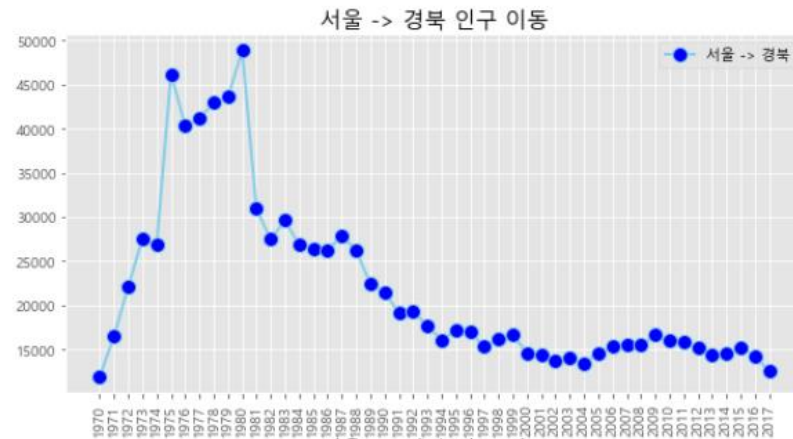
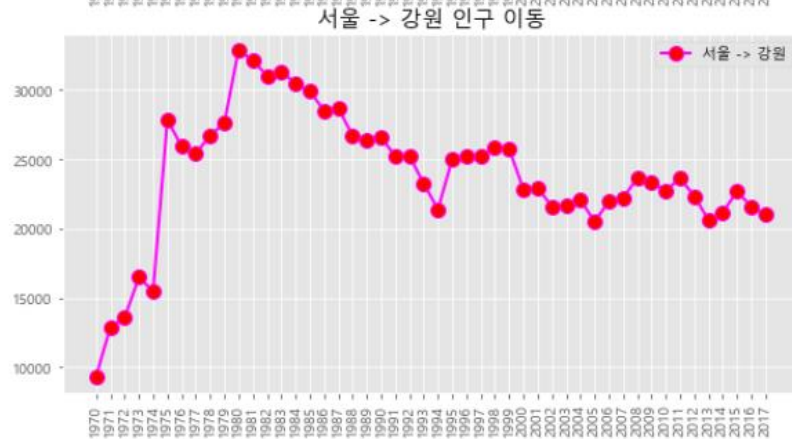
```
# 범례 표시
ax1.legend(loc='best')
ax2.legend(loc='best')
ax3.legend(loc='best')
ax4.legend(loc='best')

# 차트 제목 추가
ax1.set_title('서울 -> 충남 인구 이동', size=15)
ax2.set_title('서울 -> 경북 인구 이동', size=15)
ax3.set_title('서울 -> 강원 인구 이동', size=15)
ax4.set_title('서울 -> 전남 인구 이동', size=15)

# 축 눈금 라벨 지정 및 90도 회전
ax1.set_xticklabels(col_years, rotation=90)
ax2.set_xticklabels(col_years, rotation=90)
ax3.set_xticklabels(col_years, rotation=90)
ax4.set_xticklabels(col_years, rotation=90)

plt.show() # 변경사항 저장하고 그래프 출력
```

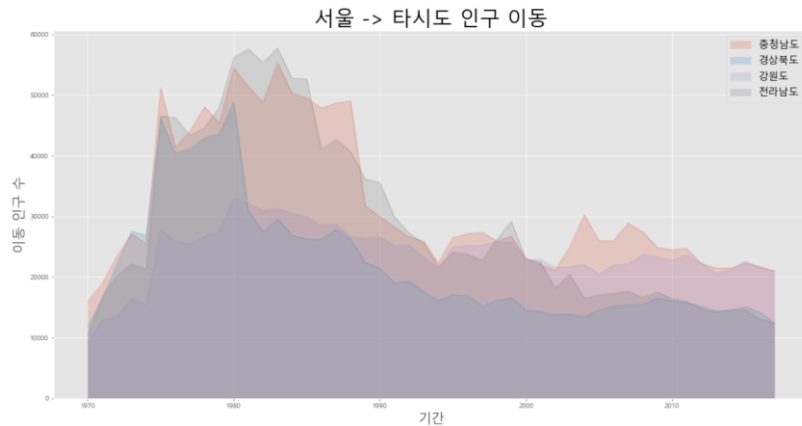
# Matplotlib 실습 예제



# Matplotlib 실습 예제

## ■ 면적그래프

```
plt.plot(kind='area')
```



```
# 서울에서 '충청남도', '경상북도', '강원도', '전라남도'로 이동한 인구 데이터 값만 선택
col_years = list(map(str, range(1970, 2018)))
df_4 = df_seoul.loc[['충청남도', '경상북도', '강원도', '전라남도'], col_years]
df_4 = df_4.transpose()
```

```
plt.style.use('ggplot')
```

```
df_4.index = df_4.index.map(int)
```

```
# 면적 그래프 그리기
```

```
df_4.plot(kind='area', stacked=False, alpha=0.2, figsize=(20, 10))
```

```
plt.title('서울 -> 타시도 인구 이동', size=30)
```

```
plt.ylabel('이동 인구 수', size=20)
```

```
plt.xlabel('기간', size=20)
```

```
plt.legend(loc='best', fontsize=15)
```

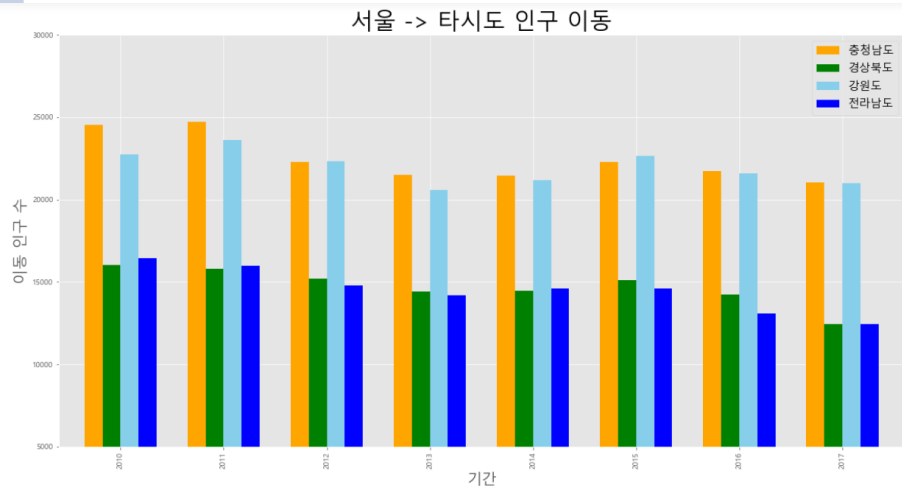
```
plt.show()
```

# Matplotlib 실습 예제

## ■ 막대그래프

```
plt.plot(kind='bar')
```

```
plt.plot(kind='barh')
```



```
# 서울에서 '충청남도', '경상북도', '강원도', '전라남도'로 이동한 인구 데이터 값만 선택
col_years = list(map(str, range(2010, 2018)))
df_4 = df_seoul.loc[['충청남도', '경상북도', '강원도', '전라남도'], col_years]
df_4 = df_4.transpose()
```

```
plt.style.use('ggplot') # 스타일 서식 지정
```

```
df_4.index = df_4.index.map(int)
```

```
# 막대 그래프 그리기
```

```
df_4.plot(kind='bar', figsize=(20, 10), width=0.7,
          color=['orange', 'green', 'skyblue', 'blue'])
```

```
plt.title('서울 -> 타시도 인구 이동', size=30)
```

```
plt.ylabel('이동 인구 수', size=20)
```

```
plt.xlabel('기간', size=20)
```

```
plt.ylim(5000, 30000)
```

```
plt.legend(loc='best', fontsize=15)
```

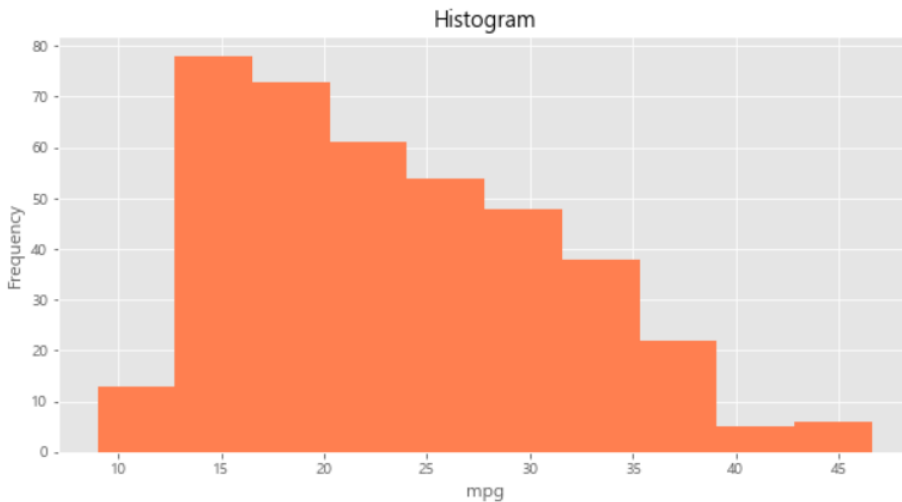
```
plt.show()
```



# Matplotlib 실습 예제

## ■ 히스토그램

```
plt.plot(kind='hist')
```



```
df = pd.read_csv('data/auto-mpg.csv', header=None)
```

```
df.columns = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',  
              'acceleration', 'model year', 'origin', 'name']
```

```
# 연비(mpg) 열에 대한 히스토그램 그리기
```

```
df['mpg'].plot(kind='hist', bins=10, color='coral', figsize=(10, 5))
```

```
# 그래프 꾸미기
```

```
plt.title('Histogram')
```

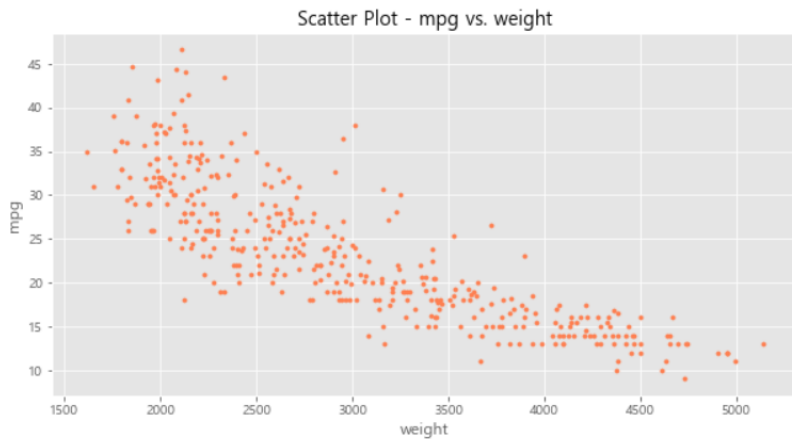
```
plt.xlabel('mpg')
```

```
plt.show()
```

# Matplotlib 실습 예제

## ■ 산점도

```
plt.plot(kind='scatter', x, y)
```



```
df = pd.read_csv('data/auto-mpg.csv', header=None)
```

```
df.columns = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',  
              'acceleration', 'model year', 'origin', 'name']
```

```
# 연비(mpg)와 차중(weight) 열에 대한 산점도 그리기
```

```
df.plot(kind='scatter', x='weight', y='mpg', c='coral', s=10, figsize=(10, 5))
```

```
plt.title('Scatter Plot - mpg vs. weight')
```

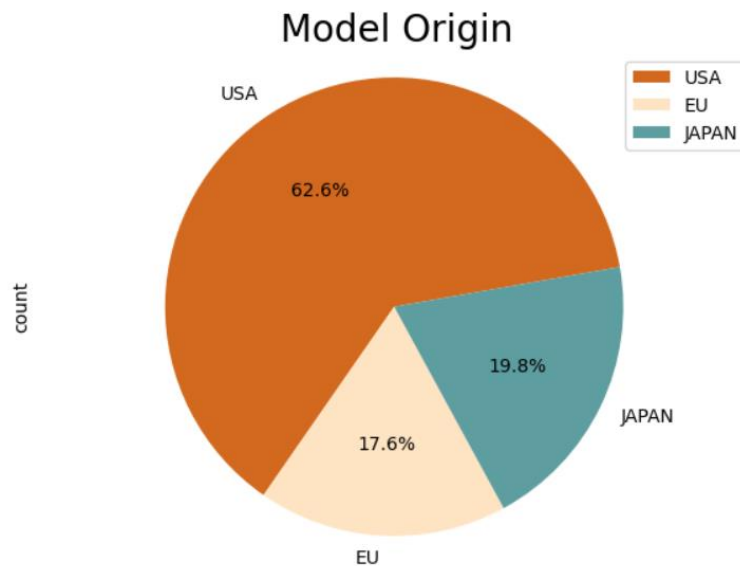
```
plt.show()
```

# Matplotlib 실습 예제

- 이전 데이터 동일(auto-mpg.csv)

## ■ 파이 차트

```
plt.plot(kind='pie', 속성들)
```



```
# 데이터 개수 카운트를 위해 값 1을 가진 열을 추가
df['count'] = 1
df_origin = df.groupby('origin').sum() # origin 열을 기준으로 그룹화, 합계 연산
print(df_origin.head())              # 그룹 연산 결과 출력
```

```
# 제조국가(origin) 값을 실제 지역명으로 변경
df_origin.index = ['USA', 'EU', 'JAPAN']
```

```
# 제조국가(origin) 열에 대한 파이 차트 그리기 - count 열 데이터 사용
```

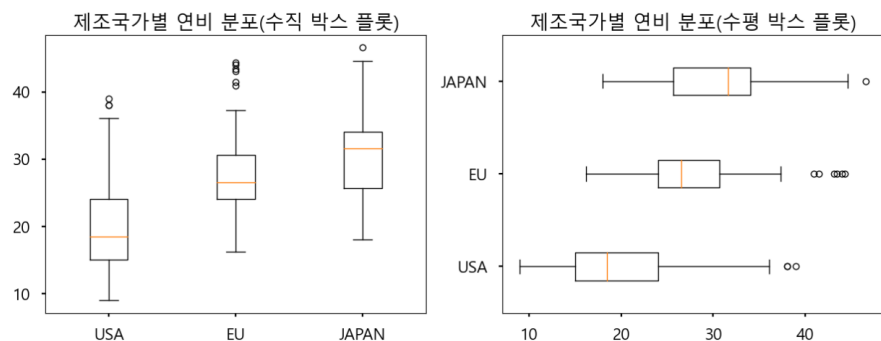
```
df_origin['count'].plot(kind='pie', figsize=(7, 5),
                        autopct='%1.1f%%', # 퍼센트 % 표시
                        startangle=10,      # 파이 조각을 나누는 시작점(각도 표시)
                        colors=['chocolate', 'bisque', 'cadetblue'] # 색상 리스트
                        )
```

```
plt.title('Model Origin', size=20)
plt.axis('equal') # 파이 차트의 비율을 같게 (원에 가깝게) 조정
plt.legend(labels=df_origin.index, loc='upper right') # 범례 표시
plt.show()
```

# Matplotlib 실습 예제

- 이전 데이터 동일(auto-mpg.csv)

## ■ 박스 플롯



```
# 그래프 객체 생성 (figure에 2개의 서브 플롯을 생성)
```

```
fig = plt.figure(figsize=(15, 5))
```

```
ax1 = fig.add_subplot(1, 2, 1)
```

```
ax2 = fig.add_subplot(1, 2, 2)
```

```
# axe 객체에 boxplot 메서드로 그래프 출력
```

```
ax1.boxplot(x=[df[df['origin']==1]['mpg'],  
              df[df['origin']==2]['mpg'],  
              df[df['origin']==3]['mpg']],  
           labels=['USA', 'EU', 'JAPAN'])
```

```
ax2.boxplot(x=[df[df['origin']==1]['mpg'],  
              df[df['origin']==2]['mpg'],  
              df[df['origin']==3]['mpg']],  
           labels=['USA', 'EU', 'JAPAN'],  
           vert=False)
```

```
ax1.set_title('제조국가별 연비 분포(수직 박스 플롯)')
```

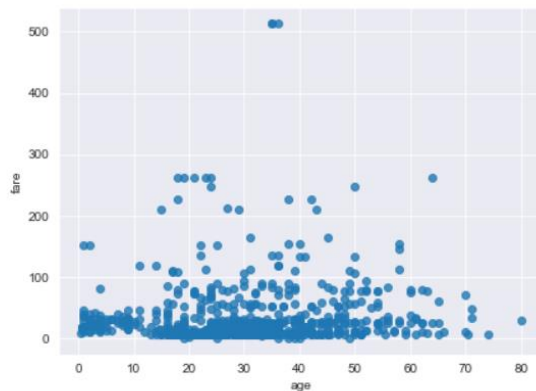
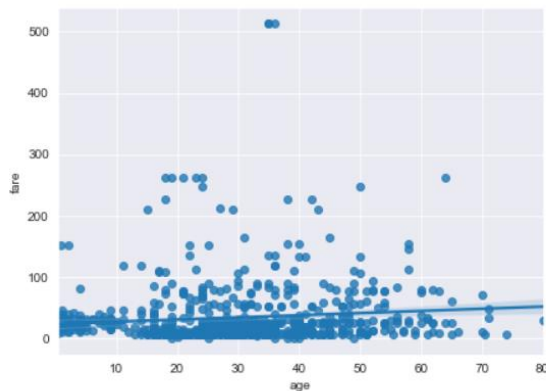
```
ax2.set_title('제조국가별 연비 분포(수평 박스 플롯)')
```

```
plt.show()
```

# 고급 그래프 도구(Seaborn)

## ■ Seaborn 라이브러리

- 파이썬 시각화 도구 고급 버전
- import 약어로 sns 사용



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

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

```
# 그래프 객체 생성 (figure에 2개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)
```

```
# 그래프 그리기 - 선형회귀선 표시(fit_reg=True)
sns.regplot(x='age',          #x축 변수
            y='fare',        #y축 변수
            data=titanic,    #데이터
            ax=ax1)          #axe 객체 - 1번째 그래프
```

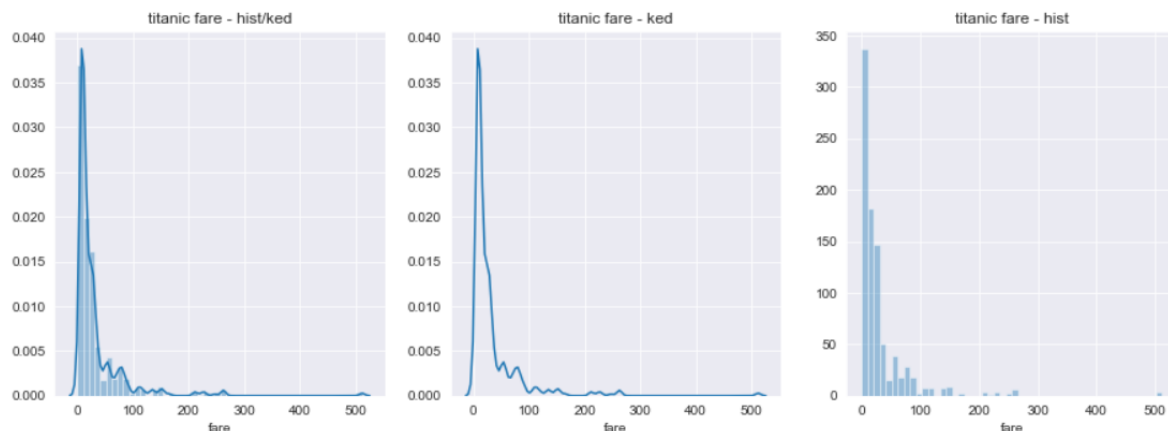
```
# 그래프 그리기 - 선형회귀선 미표시(fit_reg=False)
sns.regplot(x='age',          #x축 변수
            y='fare',        #y축 변수
            data=titanic,    #데이터
            ax=ax2,          #axe 객체 - 2번째 그래프
            fit_reg=False)    #회귀선 미표시
```

```
plt.show()
```

# 고급 그래프 도구(Seaborn)

- 히스토그램/커널 밀도 함수

`displot()`: 단일변수 데이터의 분포 확인



```
# 그래프 객체 생성 (figure에 3개의 서브 플롯을 생성)
```

```
fig = plt.figure(figsize=(15, 5))
```

```
ax1 = fig.add_subplot(1, 3, 1)
```

```
ax2 = fig.add_subplot(1, 3, 2)
```

```
ax3 = fig.add_subplot(1, 3, 3)
```

```
# 기본값
```

```
sns.distplot(titanic['fare'], ax=ax1)
```

```
# hist=False
```

```
sns.distplot(titanic['fare'], hist=False, ax=ax2)
```

```
# kde=False
```

```
sns.distplot(titanic['fare'], kde=False, ax=ax3)
```

```
# 차트 제목 표시
```

```
ax1.set_title('titanic fare - hist/ked')
```

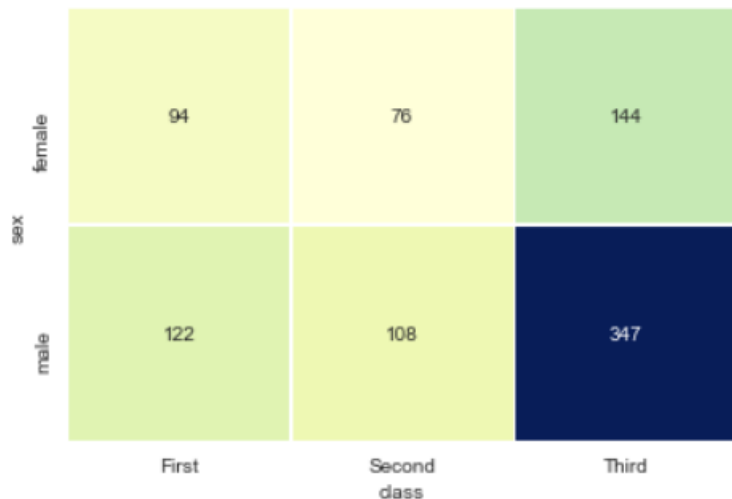
```
ax2.set_title('titanic fare - ked')
```

```
ax3.set_title('titanic fare - hist')
```

```
plt.show()
```

## 고급 그래프 도구(Seaborn)

- 히트맵(hitmap())



```
# 피벗테이블로 범주형 변수를 각각 행, 열로 재구분하여 정리
table = titanic.pivot_table(index=['sex'], columns=['class'], aggfunc='size')
```

## # 히트맵 그리기

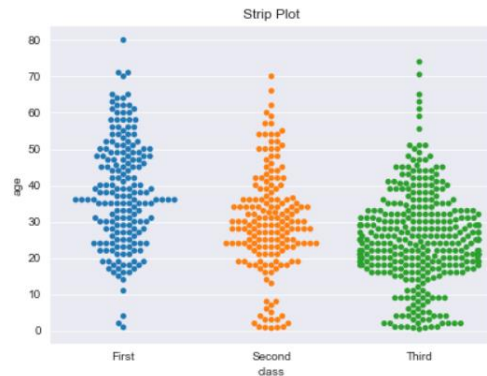
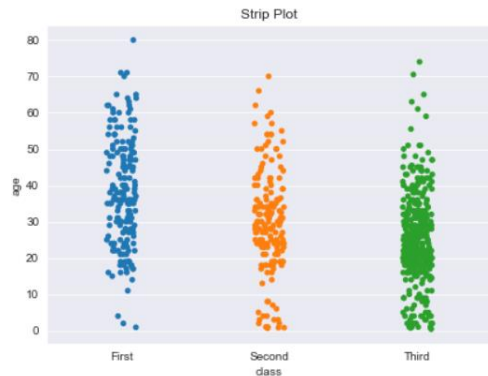
```
sns.heatmap(table,          # 데이터프레임
             annot=True,    # 데이터 값 표시 여부, 정수형 포맷
             cmap='YlGnBu', # 컬러 맵
             linewidth=.5,  # 구분 선
             cbar=False)    # 컬러 바 표시 여부
```

```
plt.show()
```

# 고급 그래프 도구(Seaborn)

- 범주형 데이터의 산점도

```
stripplot()  
swarmplot():데이터 분산 고려
```



```
# 그래프 객체 생성 (figure에 2개의 서브 플롯을 생성)  
fig = plt.figure(figsize=(15, 5))  
ax1 = fig.add_subplot(1, 2, 1)  
ax2 = fig.add_subplot(1, 2, 2)
```

```
# 이산형 변수의 분포 - 데이터 분산 미고려  
sns.stripplot(x="class",          #x축 변수  
              y="age",           #y축 변수  
              data=titanic,      #데이터셋 - 데이터프레임  
              ax=ax1)           #axe 객체 - 1번째 그래프
```

```
# 이산형 변수의 분포 - 데이터 분산 고려 (중복 x)  
sns.swarmplot(x="class",         #x축 변수  
              y="age",          #y축 변수  
              data=titanic,     #데이터셋 - 데이터프레임  
              ax=ax2)          #axe 객체 - 2번째 그래프
```

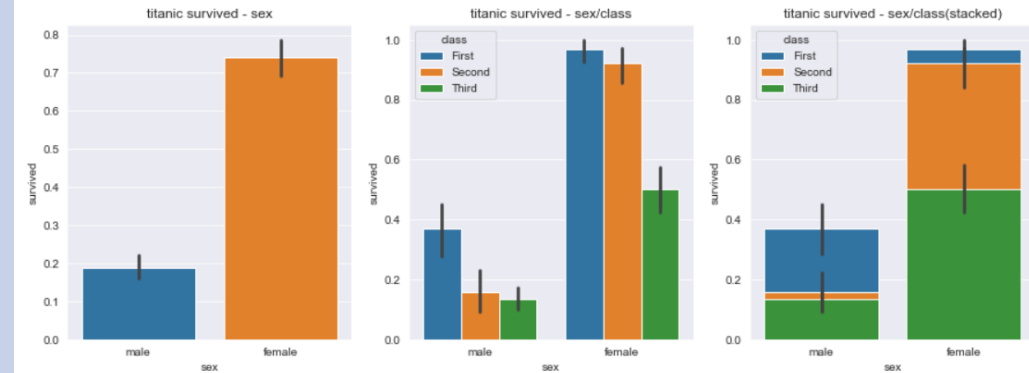
```
# 차트 제목 표시  
ax1.set_title('Strip Plot')  
ax2.set_title('Strip Plot')
```

```
plt.show()
```



# 고급 그래프 도구(Seaborn)

- 막대그래프(barplot())



```
# x축, y축에 변수 할당
```

```
sns.barplot(x='sex', y='survived', data=titanic, ax=ax1)
```

```
# x축, y축에 변수 할당하고 hue 옵션 추가
```

```
sns.barplot(x='sex', y='survived', hue='class', data=titanic, ax=ax2)
```

```
# x축, y축에 변수 할당하고 hue 옵션을 추가하여 누적 출력
```

```
sns.barplot(x='sex', y='survived', hue='class', dodge=False, data=titanic, ax=ax3)
```

```
# 차트 제목 표시
```

```
ax1.set_title('titanic survived - sex')
```

```
ax2.set_title('titanic survived - sex/class')
```

```
ax3.set_title('titanic survived - sex/class(stacked)')
```

```
plt.show()
```

---

**감사합니다**

---