DATA VISUALIZATION MATPLOTLIB

PART I

VISUALIZATION

VISUALIZATION

VISUALIZATION

시각화란

- 데이터 **분석 결과를 한눈에 이해하기 쉽게 시각적으로 표현**하는 과정
- 숫자와 표를 **그림·그래프 형태로 변환**
- **패턴·추세·이상치를 파악**하는 데이터 분석의 **핵심 단계**
- 데이터 시각화 = 데이터 해석력 + 의사소통력 + 문제 해결력

VISUALIZATION

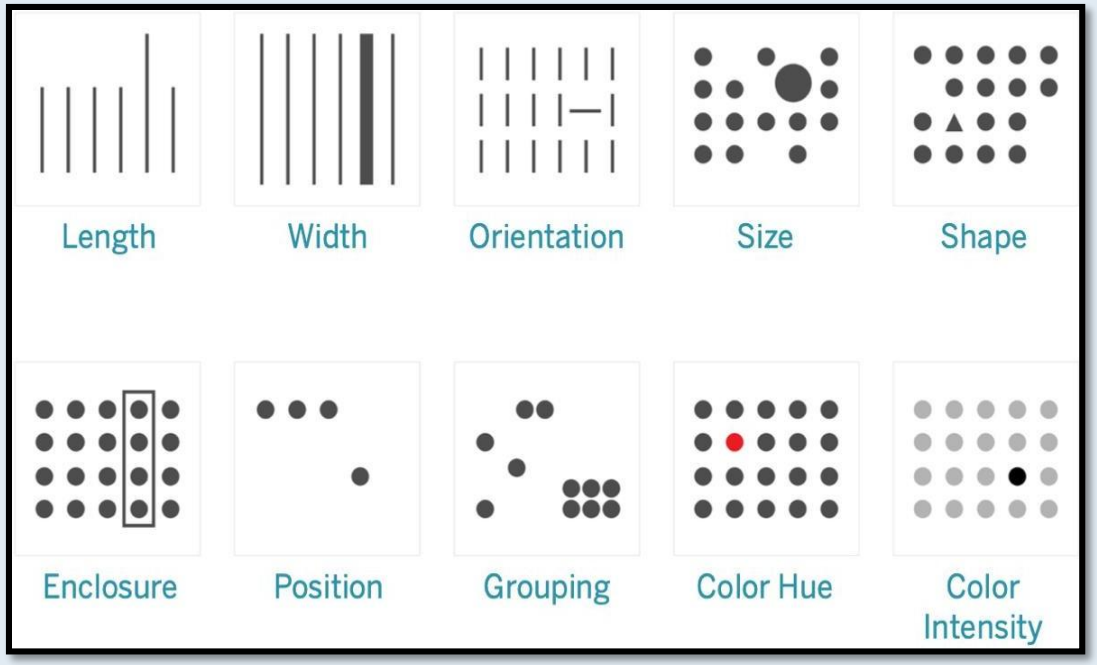
시각화 목적

▪ **데이터 이해(Understanding)**	복잡한 데이터를 시각적 패턴으로 변환해 직관적으로 이해
▪ **인사이트 발견(Insight Discovery)**	숨겨진 관계, 트렌드, 이상치(outlier)를 시각적으로 탐색
▪ **의사결정(Decision-making)**	시각적 근거를 기반으로 전략적 의사결정
▪ **소통(Communication)**	분석 결과를 비전문가나 의사결정자에게 쉽게 전달
▪ **탐색 및 예측(Exploration & Forecast)**	시각화로 패턴을 탐색해 미래 예측 모델 설계

시각화 원리

- **전주의적 속성: Pre-attentive Attribute**

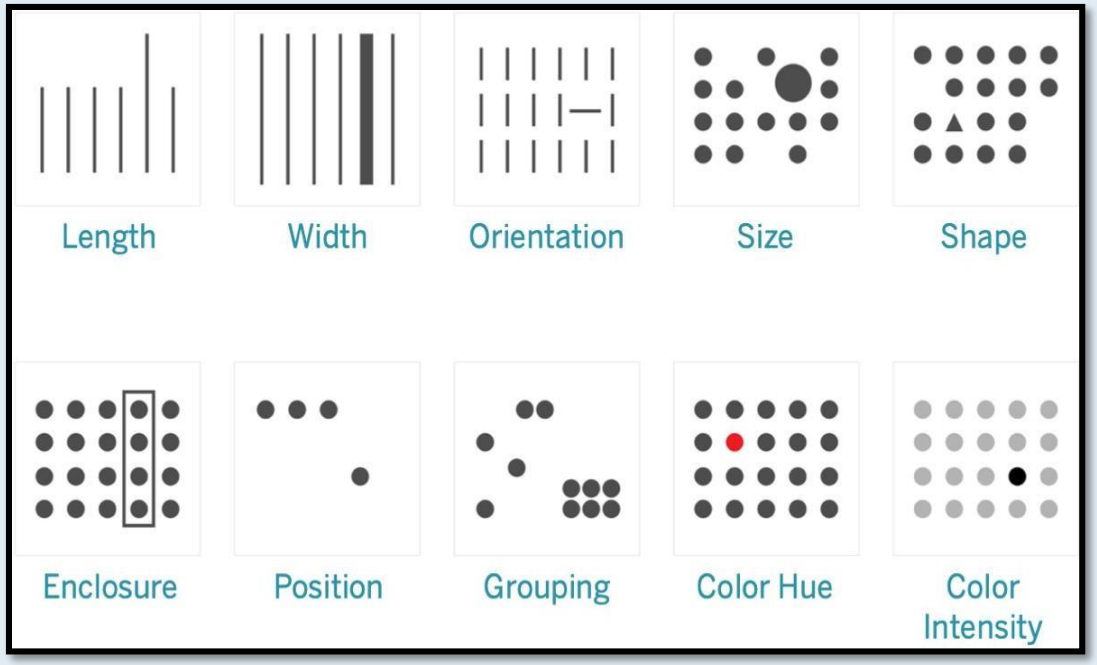
  주의를 기울이지 않아도 직관적으로 정보를 처리할 수 있는 시각적 속성



시각화 원리

- **전주의적 속성: Pre-attentive Attribute**

  주의를 기울이지 않아도 직관적으로 정보를 처리할 수 있는 시각적 속성

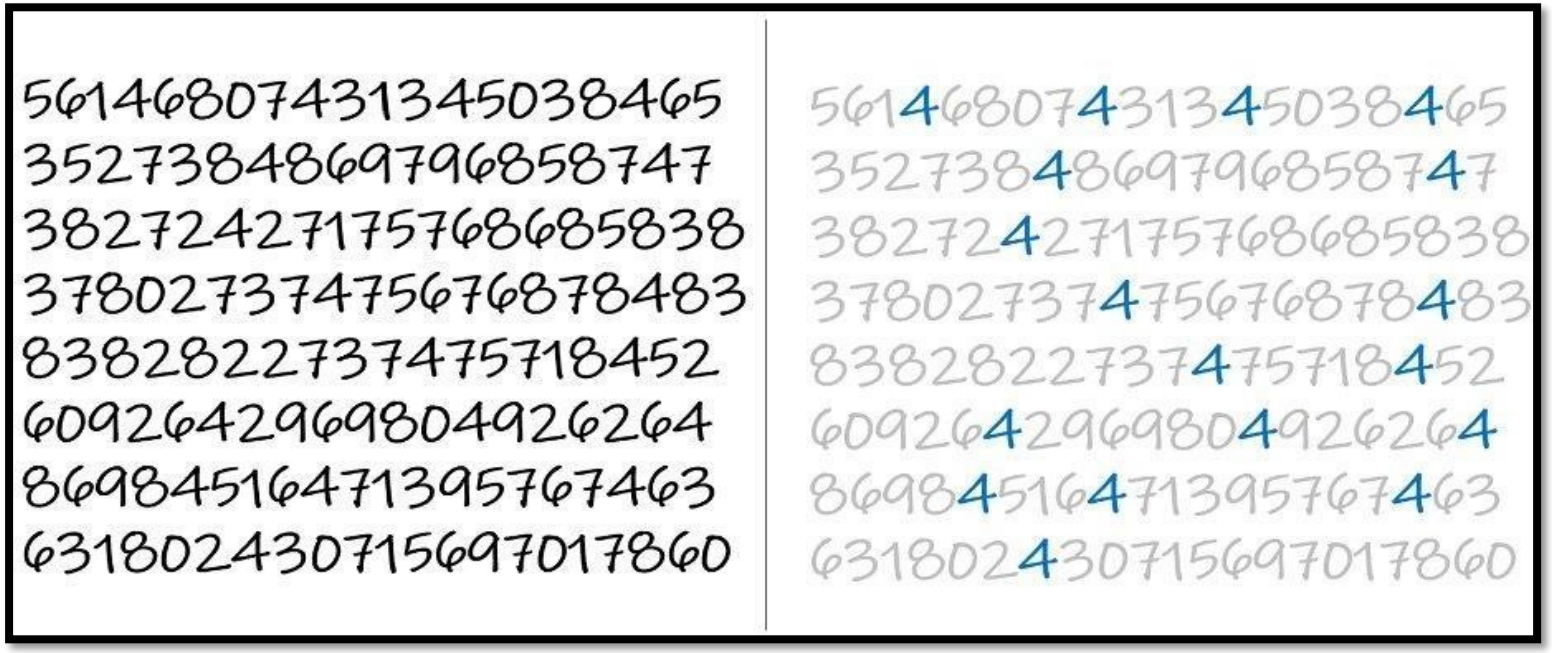


VISUALIZATION

시각화 원리

- **전주의적 속성: Pre-attentive Attribute**

  주의를 기울이지 않아도 직관적으로 정보를 처리할 수 있는 시각적 속성

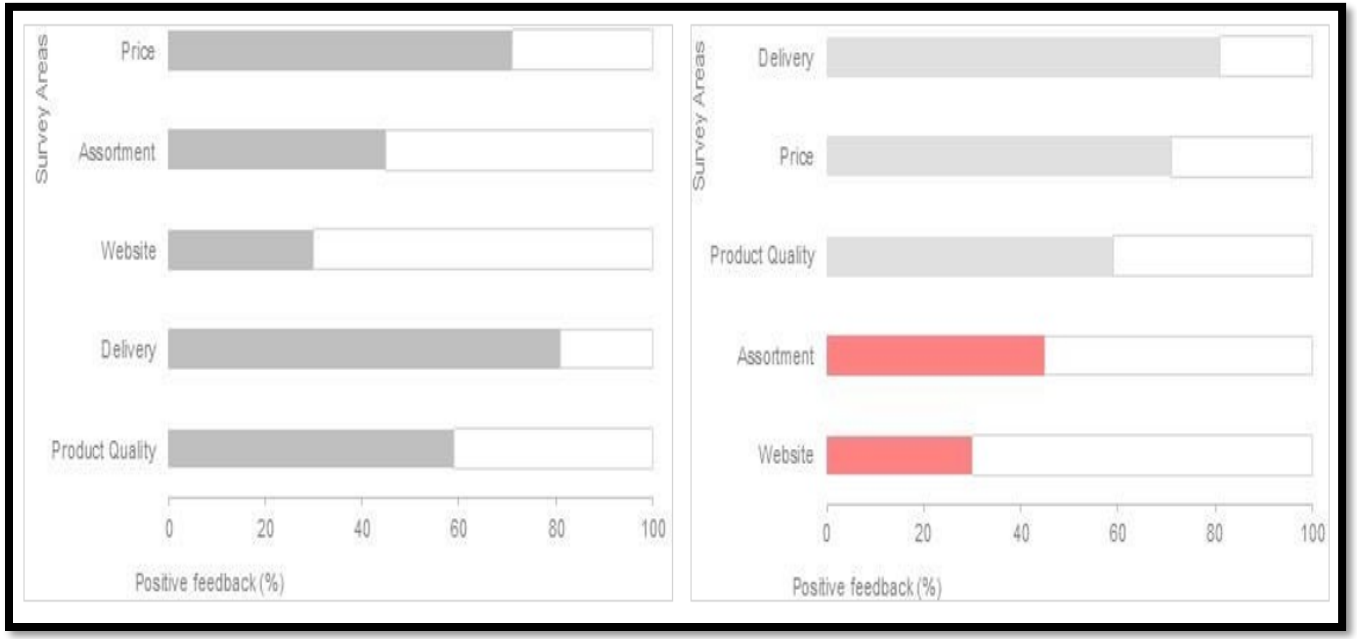


VISUALIZATION

시각화 원리

- **전주의적 속성: Pre-attentive Attribute**

  주의를 기울이지 않아도 직관적으로 정보를 처리할 수 있는 시각적 속성



시각화 원리

- **전주의적 속성: Pre-attentive Attribute**

  주의를 기울이지 않아도 직관적으로 정보를 처리할 수 있는 시각적 속성

Sales Table

Sub-Category	Consumer	Corporate	Home Office
Accessories	8,964	5,225	2,295
Appliances	4,294	2,738	1,230
Art	1,211	830	333
Binders	7,839	5,469	2,790
Bookcases	-1,966	-12	332
Chairs	1,380	1,233	1,414
Copiers	12,159	3,190	3,978
Envelopes	966	781	162
Fasteners	185	69	21
Furnishings	4,395	1,673	1,573
Labels	1,167	905	231
Machines	2,194	960	-3,773
Paper	5,617	3,883	2,619
Phones	4,469	2,565	2,077
Storage	3,828	3,491	1,326
Supplies	226	362	38
Tables	522	1,074	-114

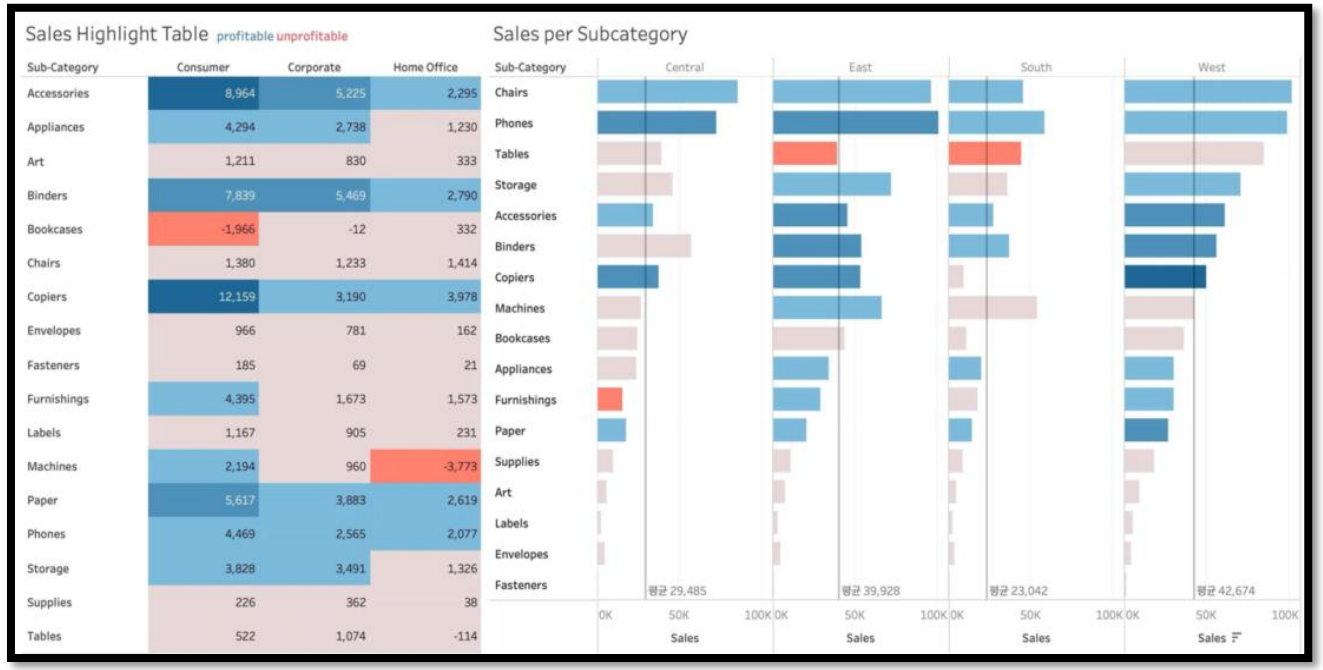
Sub-Category	Consumer	Corporate	Home Office
Accessories	8,964	5,225	2,295
Appliances	4,294	2,738	1,230
Art	1,211	830	333
Binders	7,839	5,469	2,790
Bookcases	-1,966	-12	332
Chairs	1,380	1,233	1,414
Copiers	12,159	3,190	3,978
Envelopes	966	781	162
Fasteners	185	69	21
Furnishings	4,395	1,673	1,573
Labels	1,167	905	231
Machines	2,194	960	-3,773
Paper	5,617	3,883	2,619
Phones	4,469	2,565	2,077
Storage	3,828	3,491	1,326
Supplies	226	362	38
Tables	522	1,074	-114

VISUALIZATION

시각화 원리

- **전주의적 속성: Pre-attentive Attribute**

  주의를 기울이지 않아도 직관적으로 정보를 처리할 수 있는 시각적 속성



◆ 시각화 원리

- **데이터 잉크 Data Ink**

  → 데이터 시각화 분야의 선구자인 Edward Tufle이 제창한 개념

  → 데이터 잉크 : 차트 표현 시 데이터 자체를 나타내는 부분 ➔ 시그널

  → 논데이터 잉크 : 데이터 이외의 것을 나타내는 부분 ➔ 노이즈, 잡음

  → Data-ink ratio 높을 수록 좋은 차트

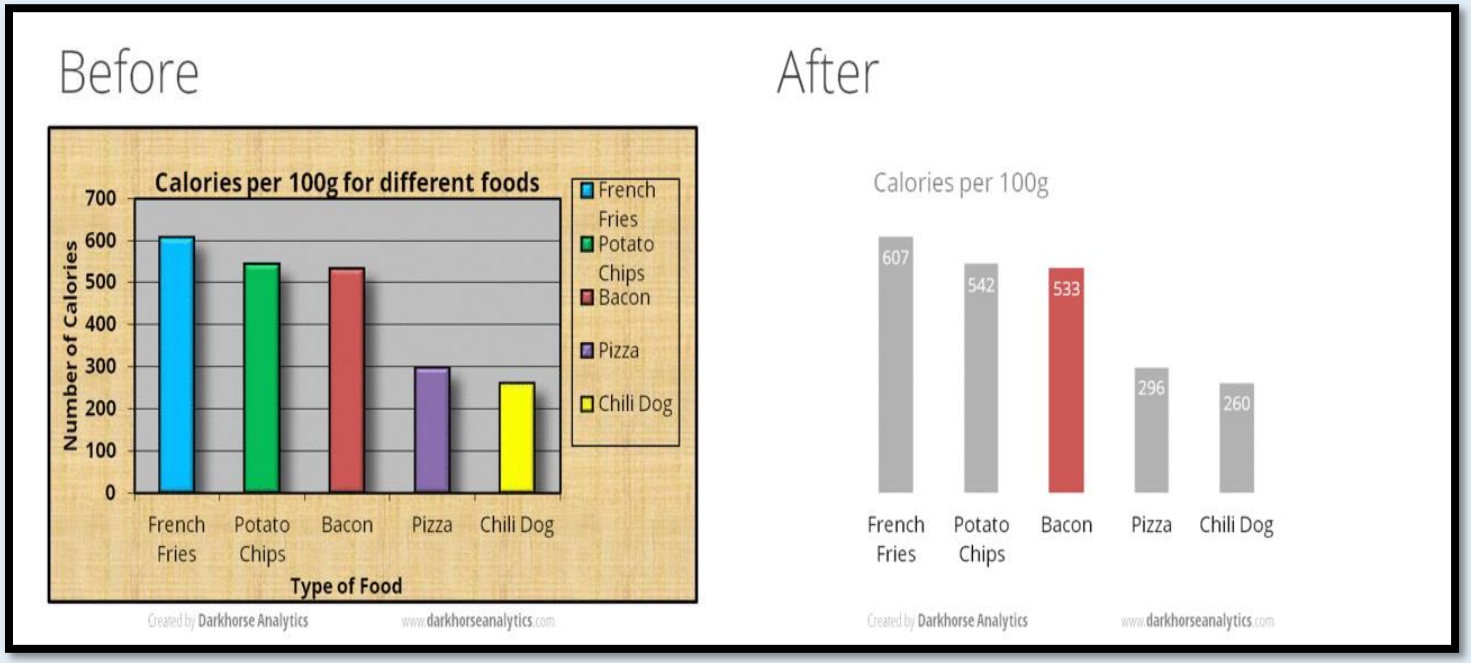
시각화 원리

- **데이터 잉크 Data Ink**
  - → 배경을 지워라.
  - → 범례를 지워라.
  - → 테투리를 지워라.
  - → 색깔을 지워라.
  - → 특수효과를 지워라.
  - → 굵은 글씨를 지워라.
  - → 라벨을 흐리게 처리해라.
  - → 보조선을 흐리게 처리하든지 아예 지워라.
  - → 라벨을 직접 표시해라.

VISUALIZATION

시각화 원리

- 데이터 잉크 Data Ink



◆ 인터렉티브 시각화 *Interactive Visualization*

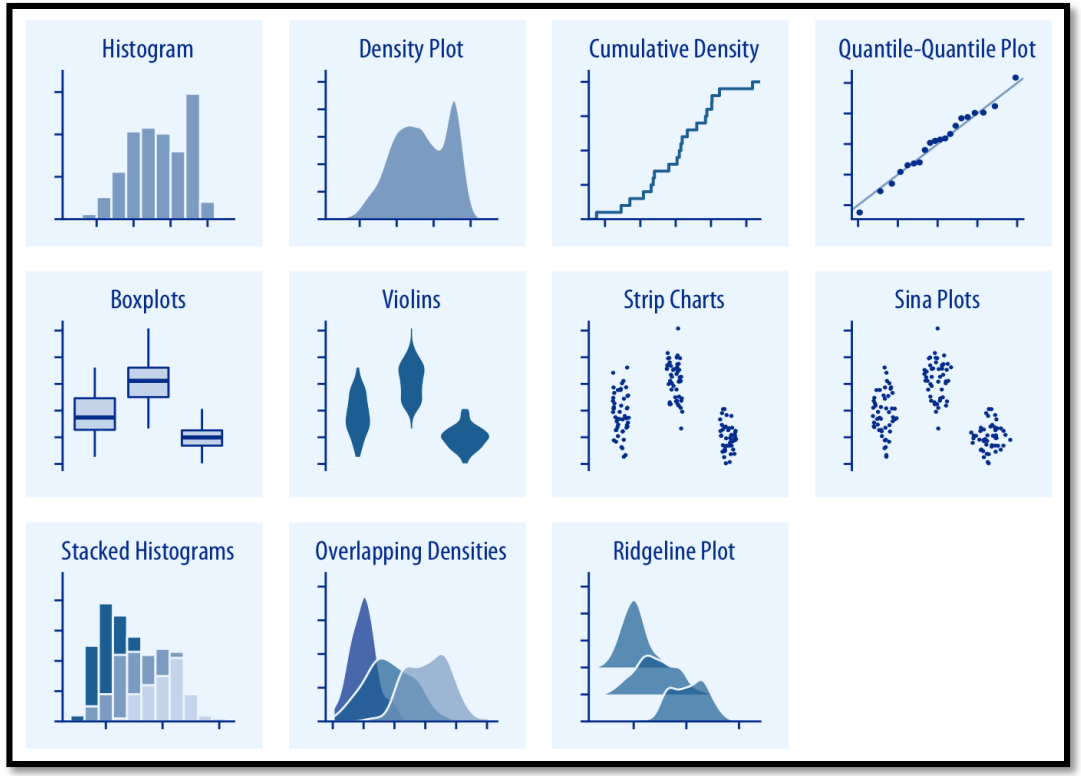
- **정보의 효과적 전달 위한 인터렉션 기반의 시각화 필요**
  - → 정보 사용자의 행동에 따른 반응과 변화를 통해 유용한 정보 제공
  - → 인포그래픽이나 모션그래픽보다 더 흥미롭고 풍부한 사용자 경험 제공

- **시각화 통한 사용자 인터렉션 방식**
  - → 하이라이트 등을 통해 강조하고 디테일을 보여주는 방식
  - → 사용자의 관심에 따라 사용자가 원하는 콘텐츠를 선택하는 방식
  - → 여러가지 방법과 다양한 차원으로 데이터를 보여주는 방식
  - → 데이터에 사용자의 관점과 의견이 반영되도록 하는 방식

VISUALIZATION

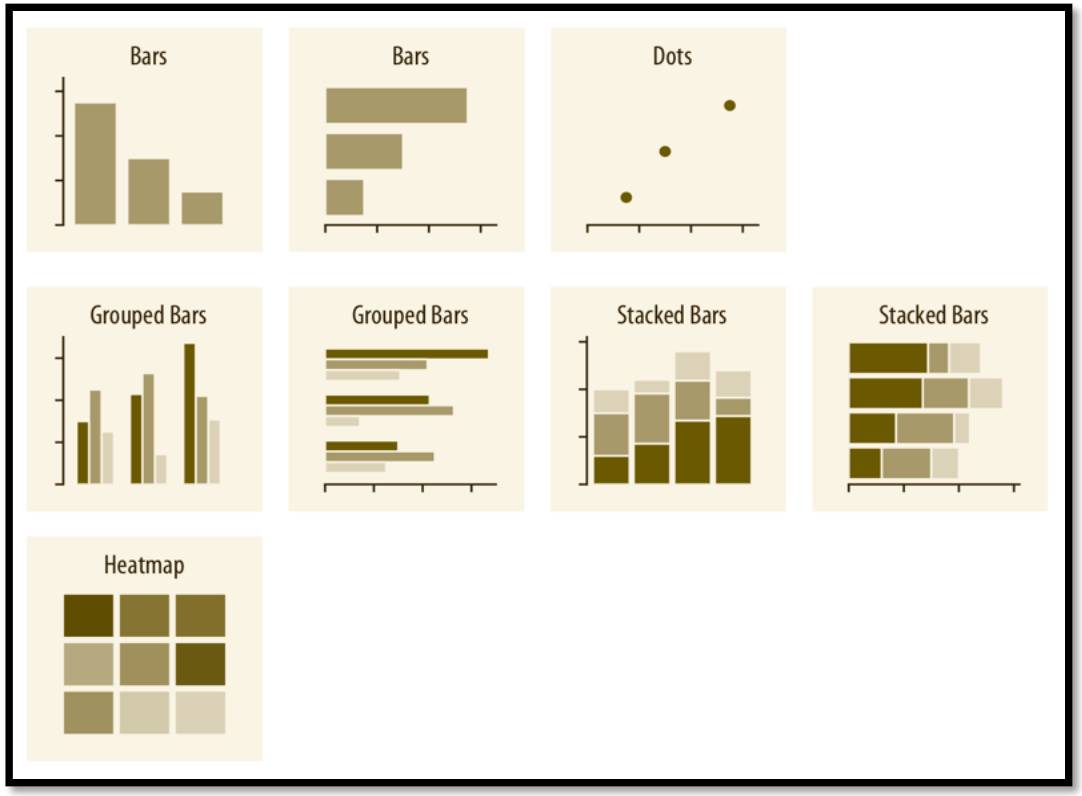
시각화 유형

- **분포(Distribution)** 값의 분포, 이상치, 편향 확인
  - → Histogram, Density Plot
  - → Cumulative Density
  - → QQ Plot , Box Plot
  - → Violin Plot , Strip Chart
  - → Sina Plot , Stacked Histogram
  - → Overlapping Density
  - → Ridgeline Plot



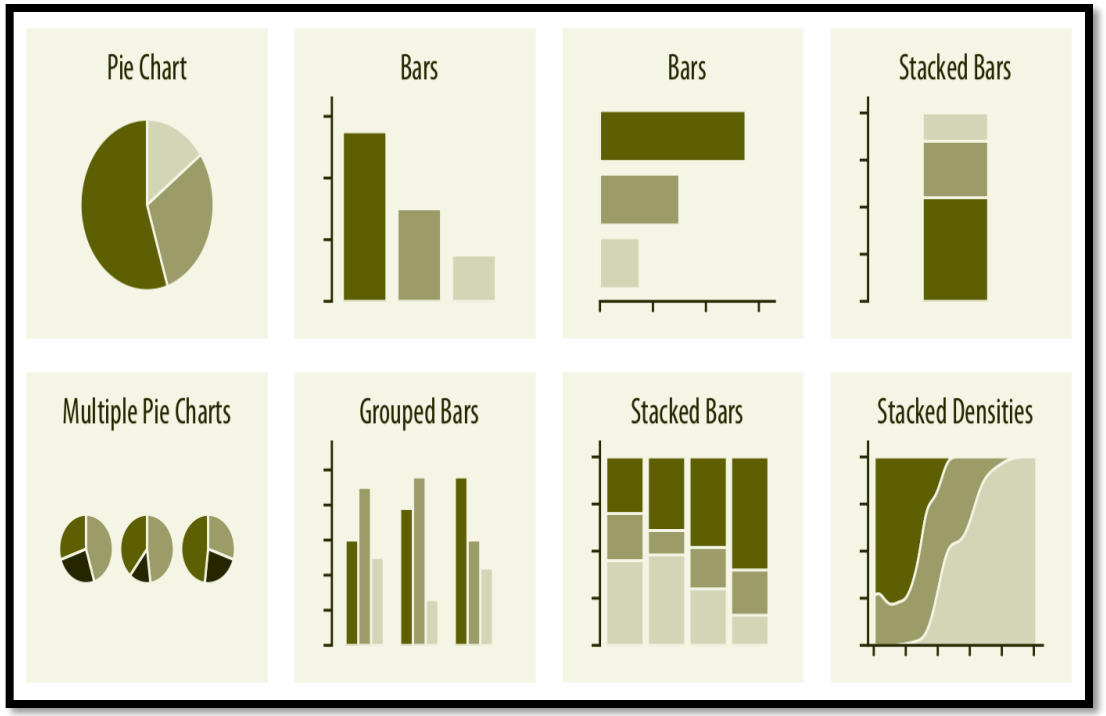
◆ 시각화 유형

- **비교(Comparison)** 그룹 간 수치 비교, 추세 비교
  - → Bar Chart
  - → Grouped Bars
  - → Stacked Bars
  - → Heatmap



◆ 시각화 유형

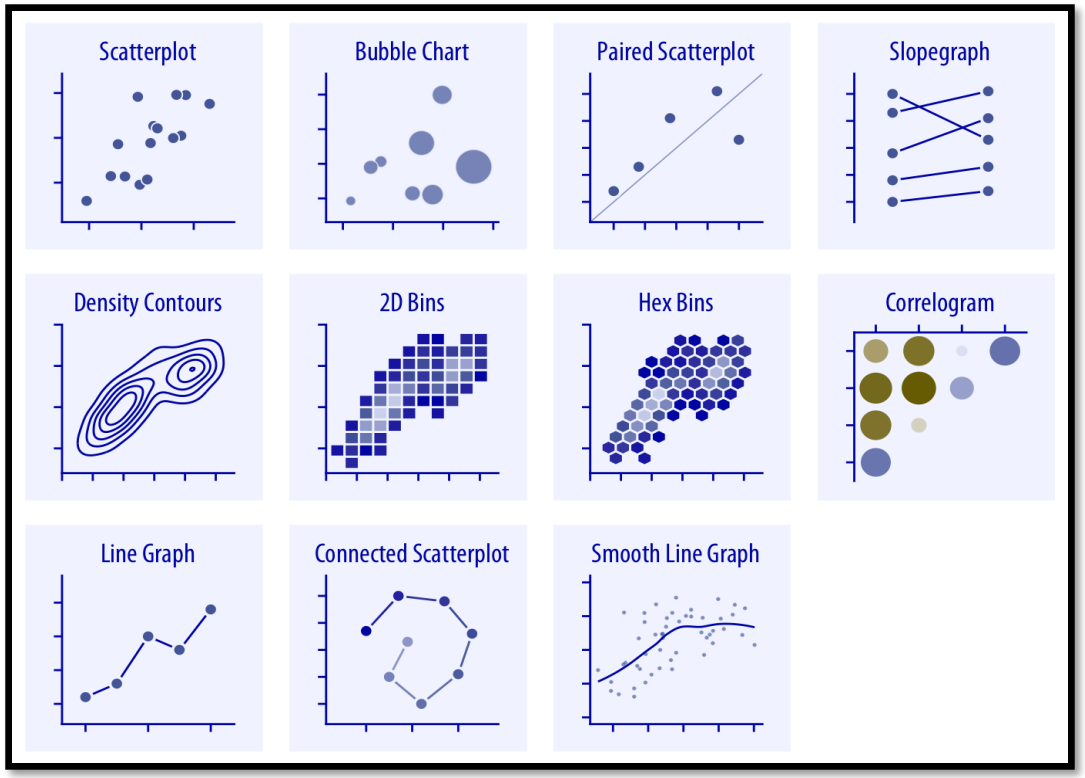
- **구성(Composition)** 전체 대비 부분 비율
  - → Pie Chart
  - → Bar Chart
  - → Stacked Bar
  - → Multiple Pies
  - → Grouped Bars
  - → Stacked Bars
  - → Stacked Densities



VISUALIZATION

시각화 유형

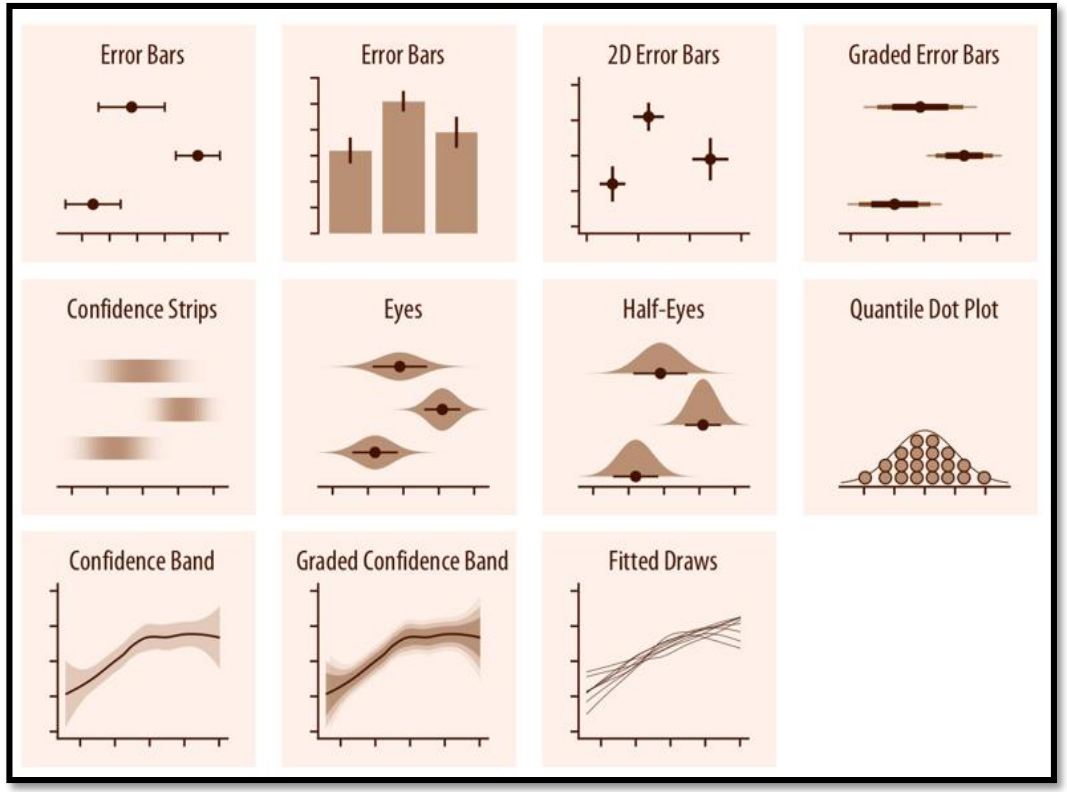
- **관계(Relationship)** 변수 간 상관관계 확인
  - → Scatter Plot, Bubble Chart
  - → Paired Scatter Plot
  - → Slope Graph
  - → Density Contours
  - → 2D Bins, Hex Bins
  - → Correlogram, Line Graph
  - → Connected Scatter Plot
  - → Smooth Line Graph



VISUALIZATION

◆ 시각화 유형

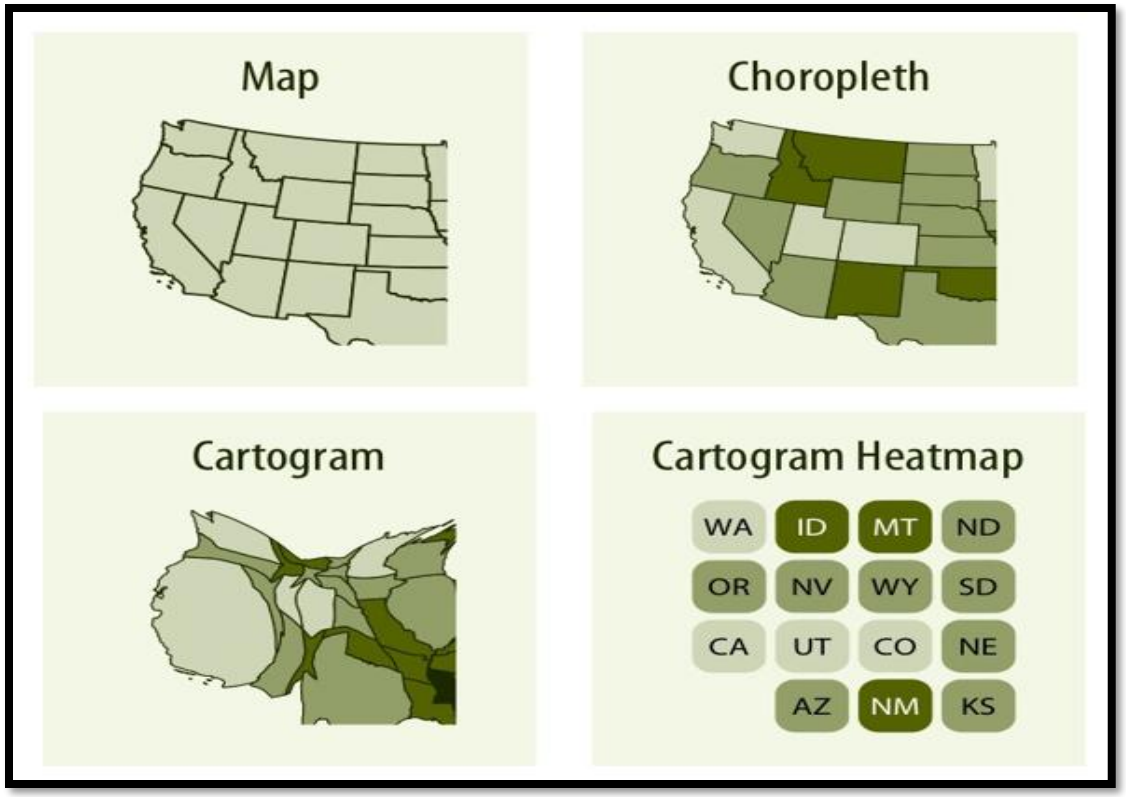
- **불확실성**
  - → Error Bars, 2D Error Bars
  - → Graded Error Bars
  - → Confidence Strips
  - → Eyes, Half-Eyes
  - → Quantile Dot Plot
  - → Confidence Band
  - → Graded Confidence Band
  - → Fitted Draws



VISUALIZATION

◆ 시각화 유형

- **지리(Geo-spatial)** 지역별 데이터, 공간 패턴
  - → Map
  - → Choropleth
  - → Cartogram
  - → Cartogram Heatmap



VISUALIZATION

◆ 시각화 유형

시각화 도구

▪ **Matplotlib**	파이썬 기본 그래프 라이브러리, 커스터마이징 자유도 높음
▪ Seaborn	통계 시각화에 강함, Matplotlib 기반, 미려한 기본 스타일
▪ Pandas.plot()	DataFrame에서 바로 시각화 가능 (내부적으로 Matplotlib 사용)
▪ Plotly / Bokeh	대화형(interactive) 그래프 제작 가능 (웹 시각화용)
▪ **Altair**	선언적 문법, 깔끔한 시각화 스타일, 복잡한 관계 데이터 표현에 유용
▪ Matplotlib + Korean Font 설정	한글 시각화를 위해 plt.rcParams["font.family"] 지정 필요

MATPLOTLIB

Matplotlib

- **분석가 중심의 시각화 제작용**
- 파이썬에서 다양한 형태의 그래프(2D, 3D)를 그릴 수 있는 시각화 패키지
- 설치 : conda install matplotlib / pip install koreanize-matplotlib
- 특징
  - → 범용성 : 선 그래프, 막대, 히스토그램, 산점도 등 거의 모든 형태 지원
  - → 세밀한 제어 : 색상, 선 스타일, 폰트, 범례, 축 눈금 등 전부 조정 가능
  - → 호환성 : Numpy, Pandas, Seaborn, Scikit-learn 등과 연동
  - → 백엔드 지원 : PNG, PDF, SVG, GUI 창(Tkinter, Qt) 등 다양한 출력 가능
  - → 기반 역할 : Seaborn, Pandas.plot() 등은 내부적으로 Matplotlib 사용

Matplotlib

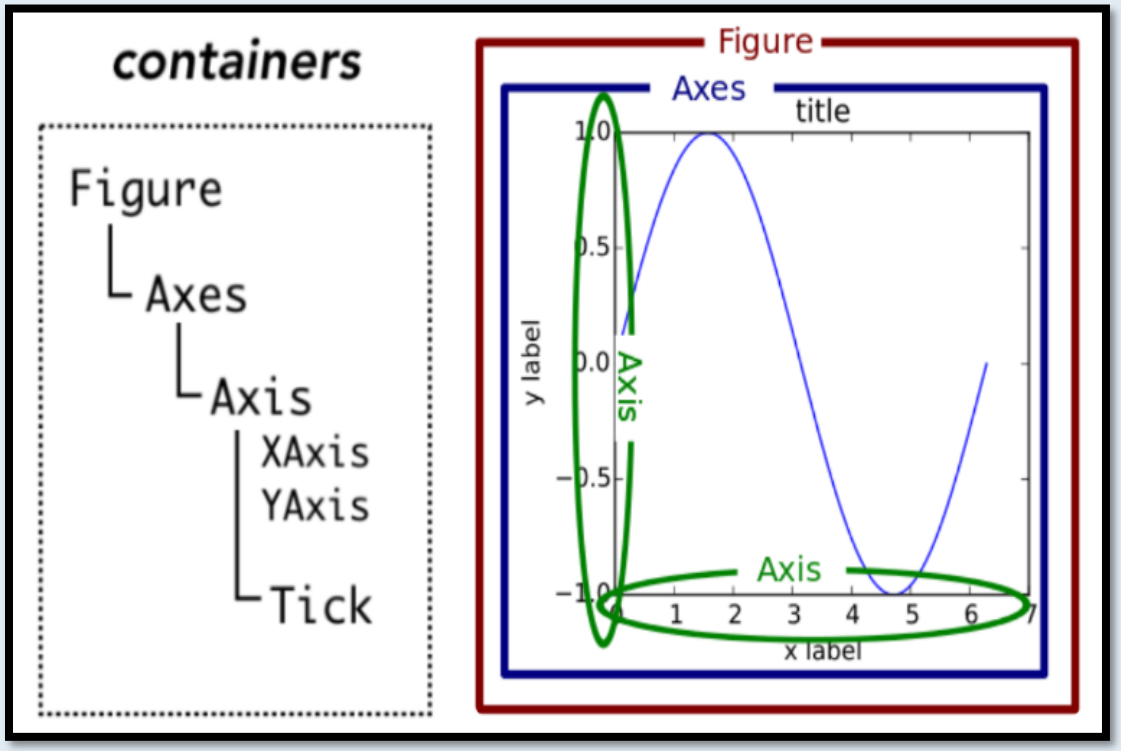
- **기본 구성 구조**

Figure (전체 도화)

└── Axes (하나의 그래프 영역)

├── Axis (x축, y축)

├── Lines / Markers / Text (그래프 요소)



MATPLOTLIB

Matplotlib

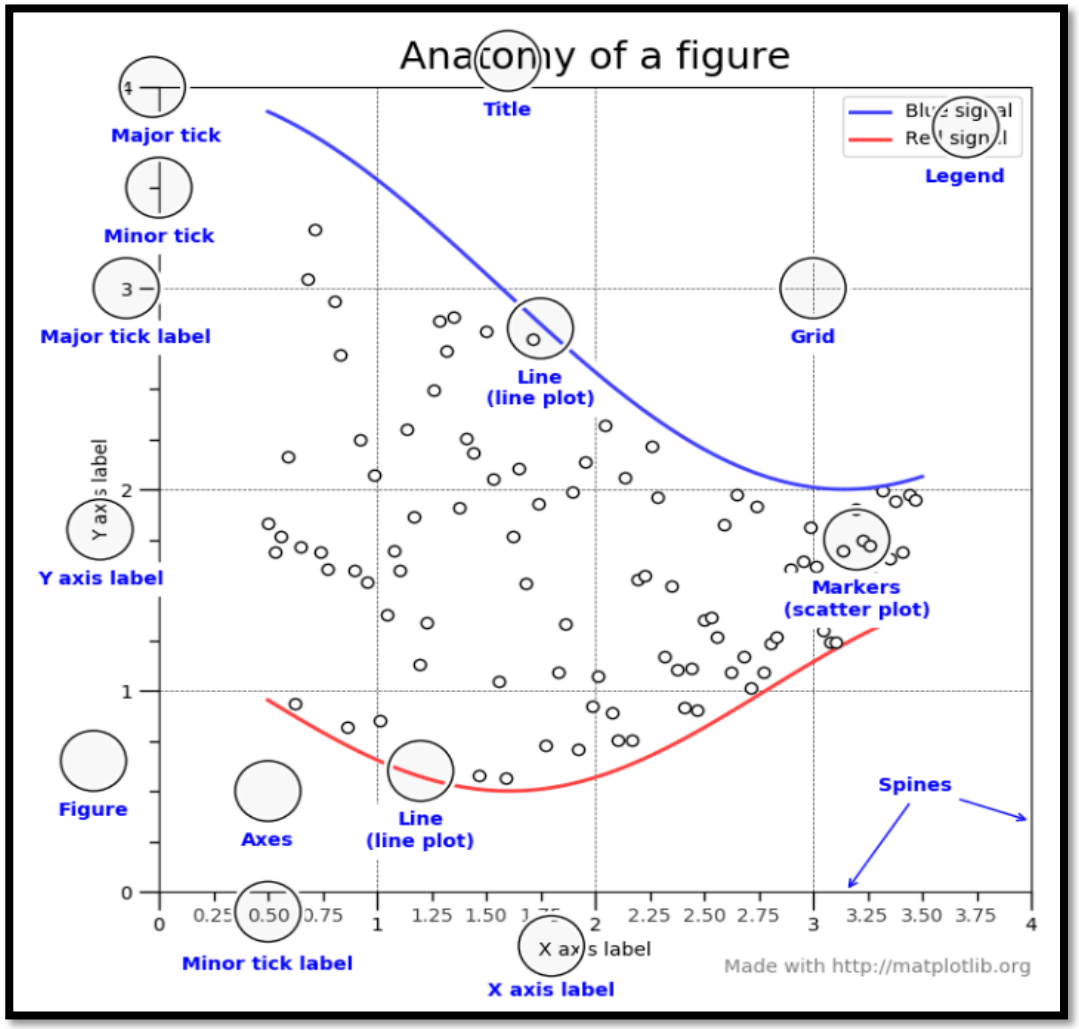
- **기본 구성 구조**

  Figure (전체 도화)

  └── Axes (하나의 그래프 영역)

  ├── Axis (x축, y축)

  ├── Lines / Markers / Text (그래프 요소)



MATPLOTLIB

Matplotlib 환경 설정

- **rc :resource configurations**

  matplotlib의 시각화에 필요한 설정 클래스

matplotlib.rcParams # [source]

An instance of `RcParams` for handling default Matplotlib values.

class **matplotlib.RcParams**(**args*, ***kwargs*) [source]

A dict-like key-value store for config parameters, including validation.

Validating functions are defined and associated with rc parameters in `matplotlib.rcsetup`.

The list of rcParams is:

- _internal.classic_mode
- agg.path.chunksize
- animation.bitrate
- animation.codec
- animation.convert_args
- animation.convert_path

Matplotlib 환경 설정 : 현재 설정값

```
# 모듈로딩
from matplotlib


# 현재 설정된 값 전체 확인
matplotlib.rcParams


# 특정 값 확인
for key in matplotlib.rcParams.keys():
    if 'axis' in key: print(key)
```

```
RcParams({'_internal.classic_mode': False,
          'agg.path.chunksize': 0,
          'animation.bitrate': -1,
          'animation.codec': 'h264',
          'animation.convert_args': ['-layers', 'OptimizePlus'],
          'animation.convert_path': 'convert',
          'animation.embed_limit': 20.0,
          'animation.ffmpeg_args': [],
          'animation.ffmpeg_path': 'ffmpeg',
          'animation.frame_format': 'png',
          'animation.html': 'none',
```

```
axes.axisbelow
axes.grid.axis
axes3d.xaxis.panecolor
axes3d.yaxis.panecolor
axes3d.zaxis.panecolor
xaxis.labellocation
yaxis.labellocation
```

Matplotlib 환경 설정 : 한글 폰트 설정

```
# matplotlib 한글 폰트 오류 문제 해결
from matplotlib import font_manager, rc

# 폰트파일 위치
FONT_PATH = "./data/malgun.ttf"
FONT_NAME = font_manager.FontProperties(fname=FONT_PATH).get_name()

# 한글 폰트 설정 – 방법1)
rc('font', family=FONT_NAME)

# 한글 폰트 설정 – 방법2)
rcParams['font.family'] = FONT_NAME
```

Matplotlib 환경 설정 : 마이너스 설정

```
# matplotlib 마이너스 깨짐 문제 설정
from matplotlib import rc, rcParams


# 마이너스 기호 설정 – 방법1)
rc('axes', unicode_minus=False)


# 마이너스 기호 설정 – 방법2)
rcParams['axes.unicode_minus'] = True
```

VISUALIZE DATA BY TYPE

VISUALIZE DATA BY TYPE

◆ 선(Line) 그래프

- **선 기울기**로 데이터 표현
- 시기별 수익 / 트렌드 변화 및 예측 분석
- **시간에 따른 데이터 변화** 분석 효과적
- **활용 ➔항목 간 트렌드 변화 비교, 데이터 변화**

[팁]

- 한 그래프에 **너무 많은 선 사용하지 말것**
- 항목 차별은 색상 적용
- **세로축의 범위 적절한 조정 필요**

VISUALIZE DATA BY TYPE

선(Line) 그래프

plot([x], y, [fmt], *, data=None, **kwargs **)**

plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs **)**

```
plot(x, y)          # plot x and y using default line style and color
plot(x, y, 'bo')    # plot x and y using blue circle markers
plot(y)             # plot y using x as index array 0..N-1
plot(y, 'r+')       # ditto, but with red plusses
```

VISUALIZE DATA BY TYPE

선(Line) 그래프

fmt = '[marker][line][color]'

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker

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

VISUALIZE DATA BY TYPE

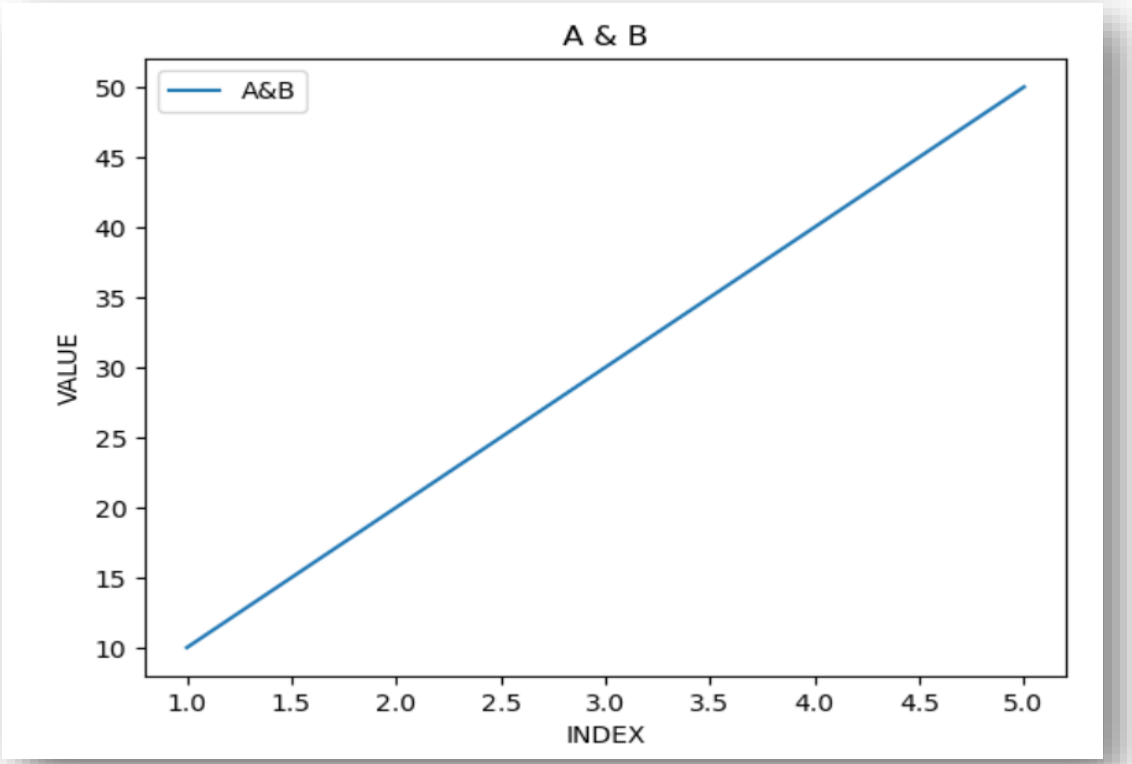
선(Line) 그래프

```
## 모듈 로딩
import pandas as pd                                  # 데이터 분석 및 전처리 모듈
import matplotlib.pyplot as plt                      # 시각화 모듈

## 데이터 준비
data = {'A' :[10,20,30,40,50], 'B':[15,25,35,45,55]}
dataDF=pd.DataFrame(data)

## - A컬럼 시각화
xData = dataDF.index + 1
yData = dataDF['A']
```

VISUALIZE DATA BY TYPE

선(Line) 그래프

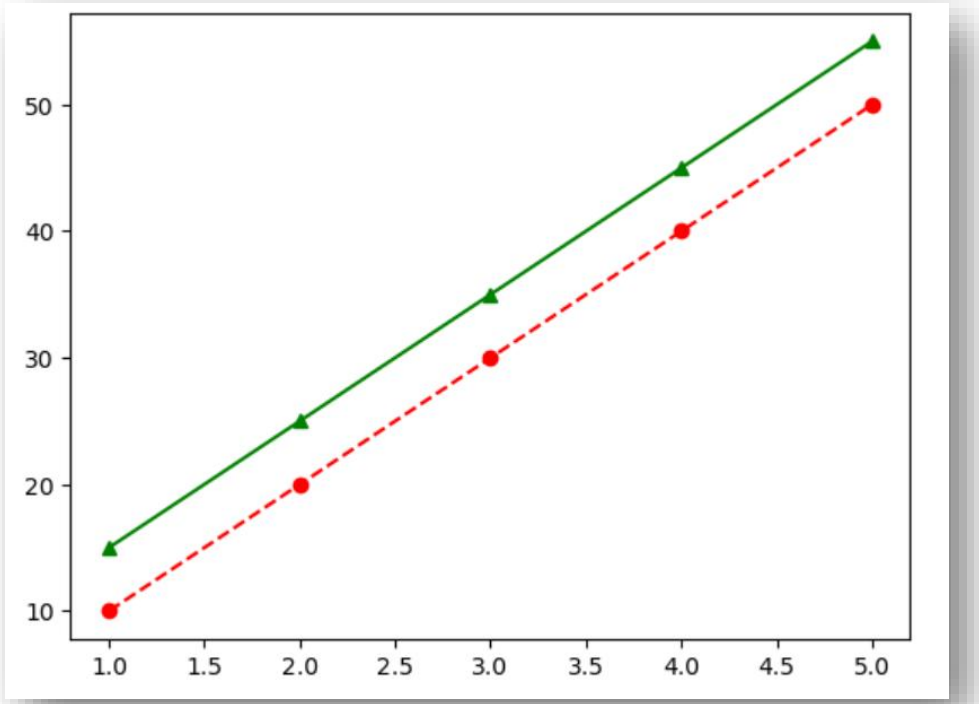
```
## - A컬럼 시각화
xData = dataDF.index + 1
yData = dataDF['A']

## - 선 그래프 시각화
plt.plot(xData, yData)

plt.title("A & B")
plt.xlabel("INDEX")
plt.ylabel("VALUE")
plt.legend(labels=['A&B'])

plt.show()
```



VISUALIZE DATA BY TYPE

선(Line) 그래프

```
## - A컬럼과 B컬럼 시각화
xData = dataDF.index + 1
yData1 = dataDF['A']
yData2 = dataDF['B']

## - 시각화
## A 컬럼과  B 컬럼 한번에 그리기
plt.plot(xData, yData1, 'ro--',
         xData, yData2, 'g^-')

plt.show()
```



VISUALIZE DATA BY TYPE

막대(Bar) 그래프

- **범주/항목 있는 데이터 값을 직사각형의 막대로 표현**하는 그래프
- 항목별 **구체적 수치 비교**
- 수직 막대 그래프, 수평 막대 그래프
- **활용 ➔트렌드 분석, 순위 비교, 달성도 확인**

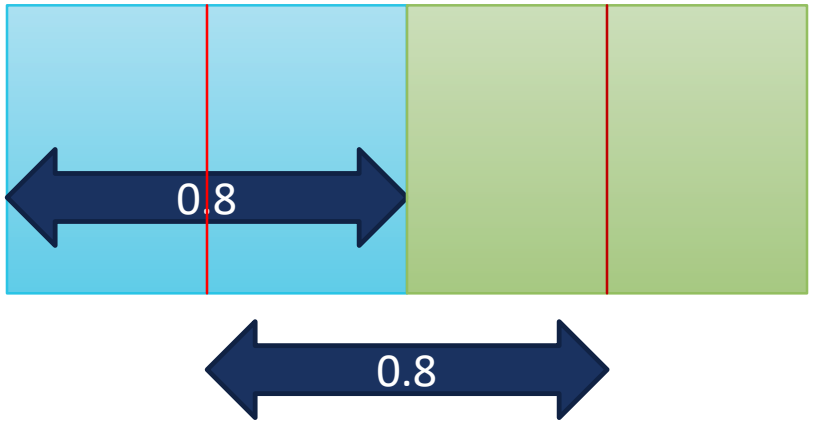
- **수직 막대 그래프 ← 순위(정렬), 트렌드 파악**
- **수평 막대 그래프 ← 항목 많은 경우, 달성도 파악**
- 막대 색상 및 채도 조절에 따른 시각화

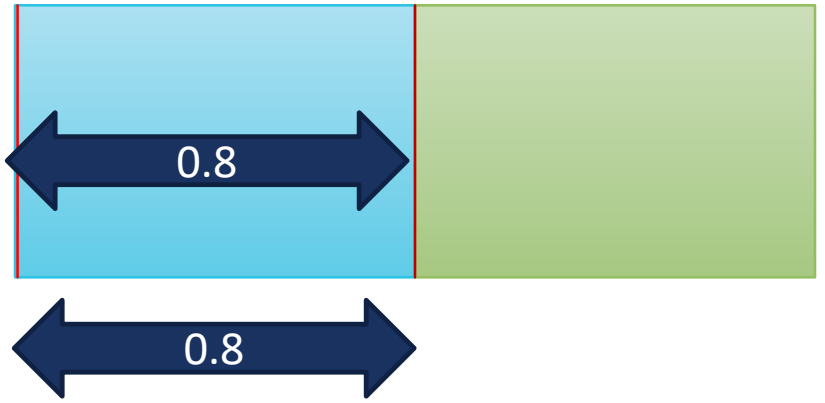
VISUALIZE DATA BY TYPE

막대(Bar) 그래프

bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs **)**

barh(y, width, height=0.8, left=None, *, align='center', data=None, **kwargs **)**

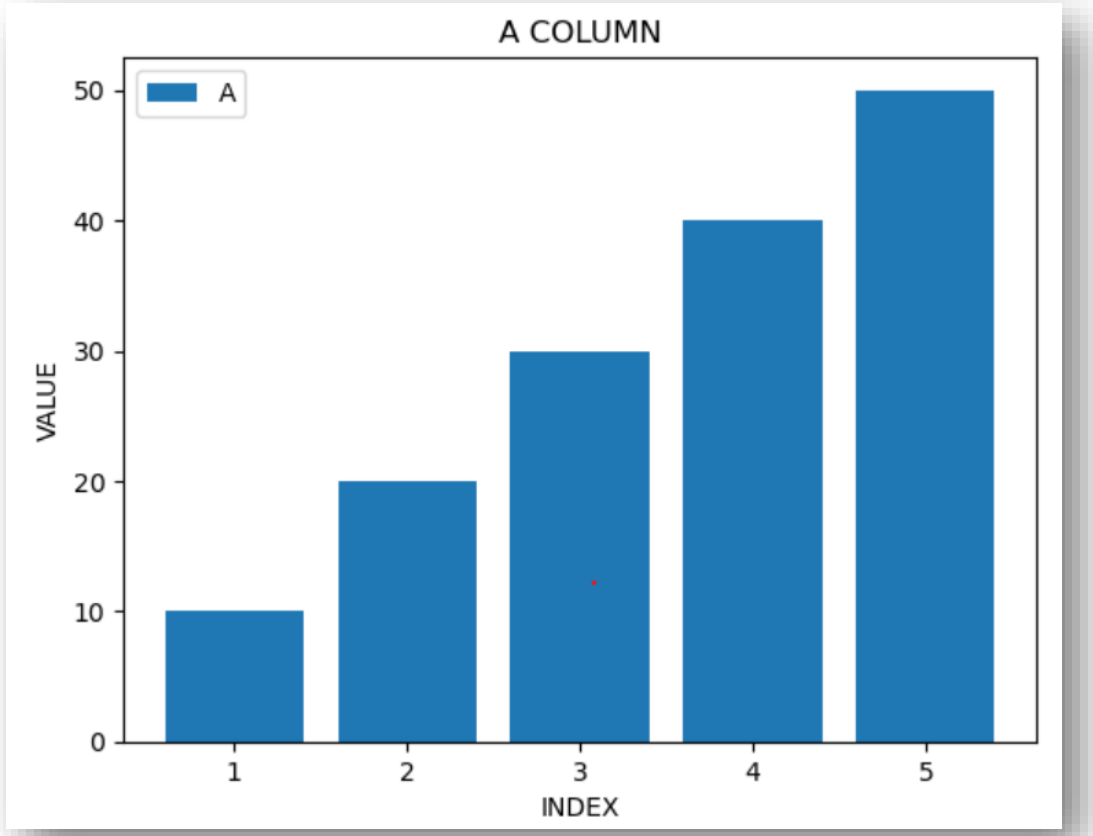




VISUALIZE DATA BY TYPE

막대(Bar) 그래프

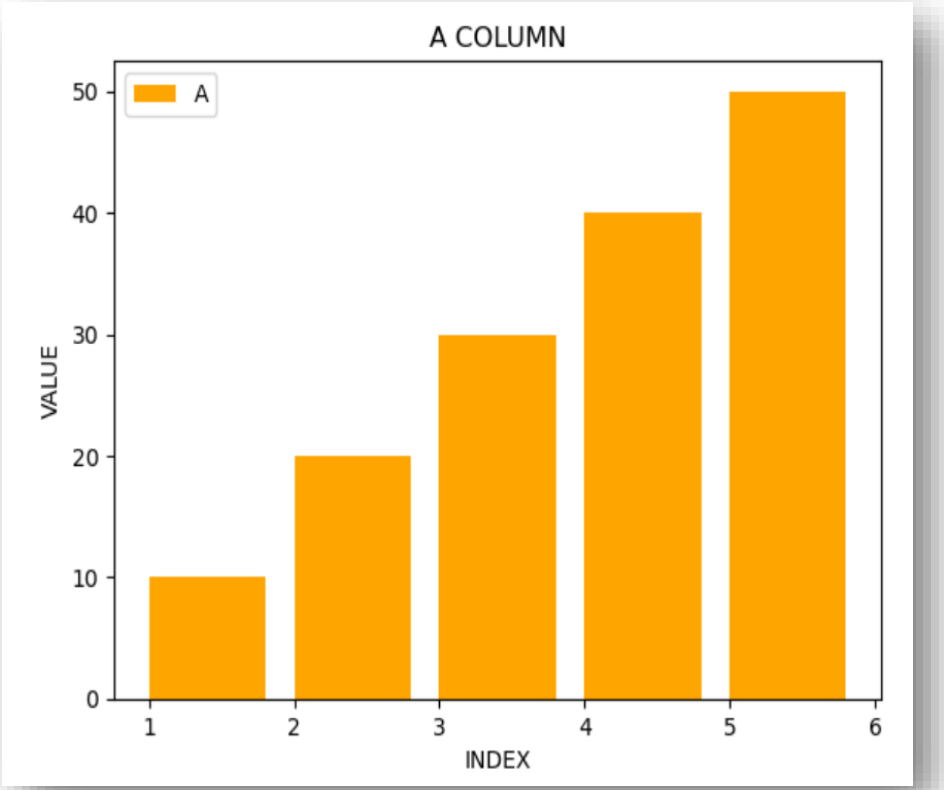
```
## - A컬럼 시각화
xData = dataDF.index + 1
yData = dataDF['A']

## - 막대 그래프 시각화
plt.bar(xData, yData)

plt.title("A & B")
plt.xlabel("INDEX")
plt.ylabel("VALUE")
plt.legend(labels=['A&B'])

## - 화면 출력
plt.show()
```



VISUALIZE DATA BY TYPE

막대(Bar) 그래프

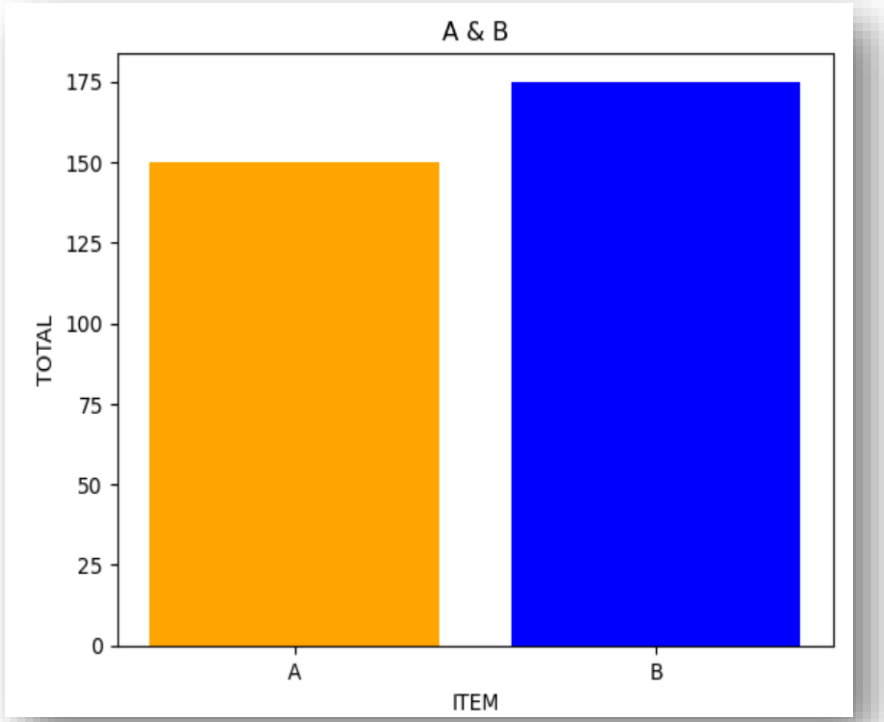
```
## - A컬럼 시각화
xData = dataDF.index + 1
yData = dataDF['A']

## - 막대 그래프 시각화
plt.bar(xData, yData, color='orange', align='edge')

plt.title("A & B")
plt.xlabel("INDEX")
plt.ylabel("VALUE")
plt.legend(labels=['A&B'])

## - 화면 출력
plt.show()
```



VISUALIZE DATA BY TYPE

막대(Bar) 그래프

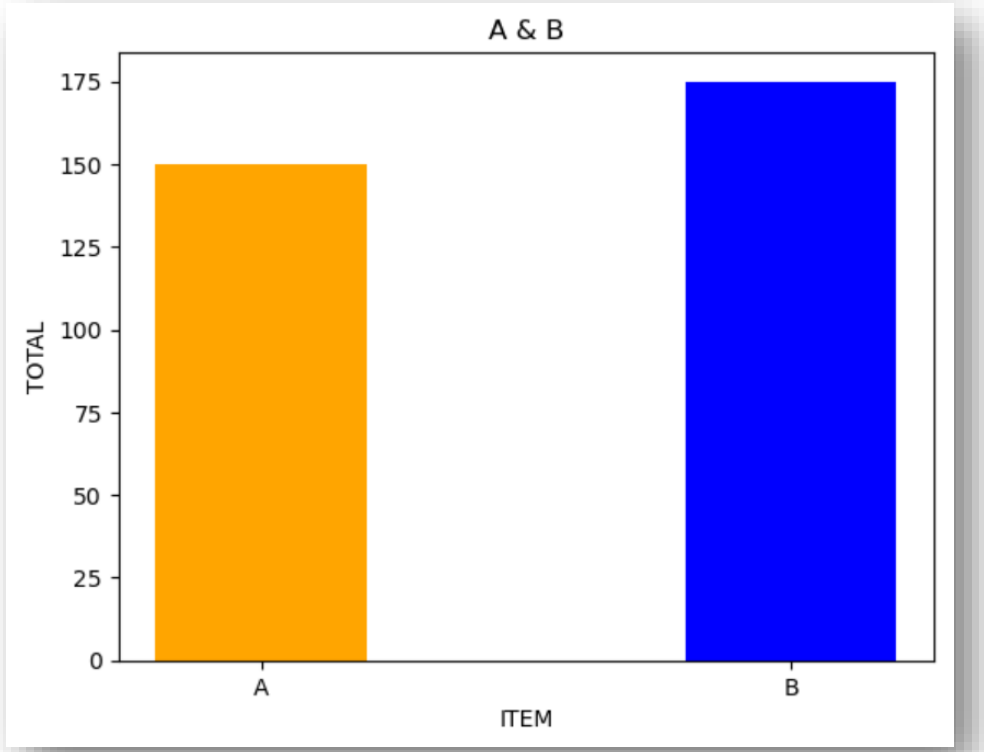
```
## - A컬럼과 B컬럼 시각화
sumSR = dataDF.sum()
xData  = sumSR.index
yData  = [sumSR['A'], sumSR['B']]
colors  = ['orange','blue']

## - 막대 그래프 시각화
plt.bar(xData, yData, color=colors, align='center')

plt.title("A & B")
plt.xlabel("ITEM")
plt.ylabel("TOTAL")

plt.show()
```



VISUALIZE DATA BY TYPE

막대(Bar) 그래프

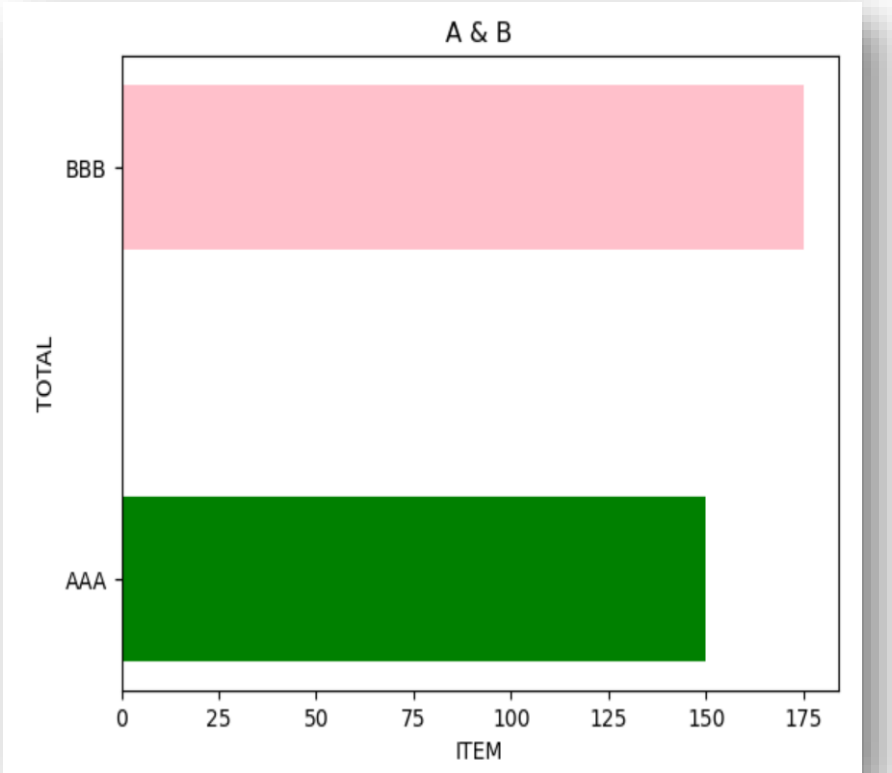
```
## - A컬럼과 B컬럼 시각화
sumSR = dataDF.sum()
xData  = sumSR.index
yData  = [sumSR['A'], sumSR['B']]
colors  = ['orange','blue']

## - 막대 그래프 시각화
plt.bar(xData, yData,
        color=colors, width=0.4)

plt.title("A & B")
plt.xlabel("ITEM")
plt.ylabel("TOTAL")

plt.show()
```



VISUALIZE DATA BY TYPE

막대(Bar) 그래프

```
## - A컬럼과 B컬럼 시각화
sumSR = dataDF.sum()
xData  = sumSR.index
yData  = [sumSR['A'], sumSR['B']]
colors  = ['green','pink']

## - 막대 그래프 시각화
plt.barh(xData, yData, color=colors, height=0.4)

plt.yticks(xData, ['AAA','BBB'])
plt.title("A & B")
plt.xlabel("ITEM")
plt.ylabel("TOTAL")

plt.show()
```



◆ 영역(Area) 그래프

- **선 그래프의 일종으로 선 아래에 영역이 채워진 형태**
- **시간에 따라 측정값의 변화를 시각화**
- **동일 데이터 내에서 세부 분류의 비중/정도 표현 가능**
- **누적(Stack)** 개념 추가된 시각화에 주로 사용 ➔ 날짜 데이터 시각화에는 선 그래프 사용

[팁!]

- 너무 많은 조작으로 나누지 말것
- 크기를 큰 순서대로 나열하여 명확하게 비교
- 핵심 정보만 담을 것

VISUALIZE DATA BY TYPE

◆ 영역(Area) 그래프

```
fill_between( x,  y1,  y2=0, where=None, interpolate=False, step=None, *,
                                    data=None, **kwargs )
```

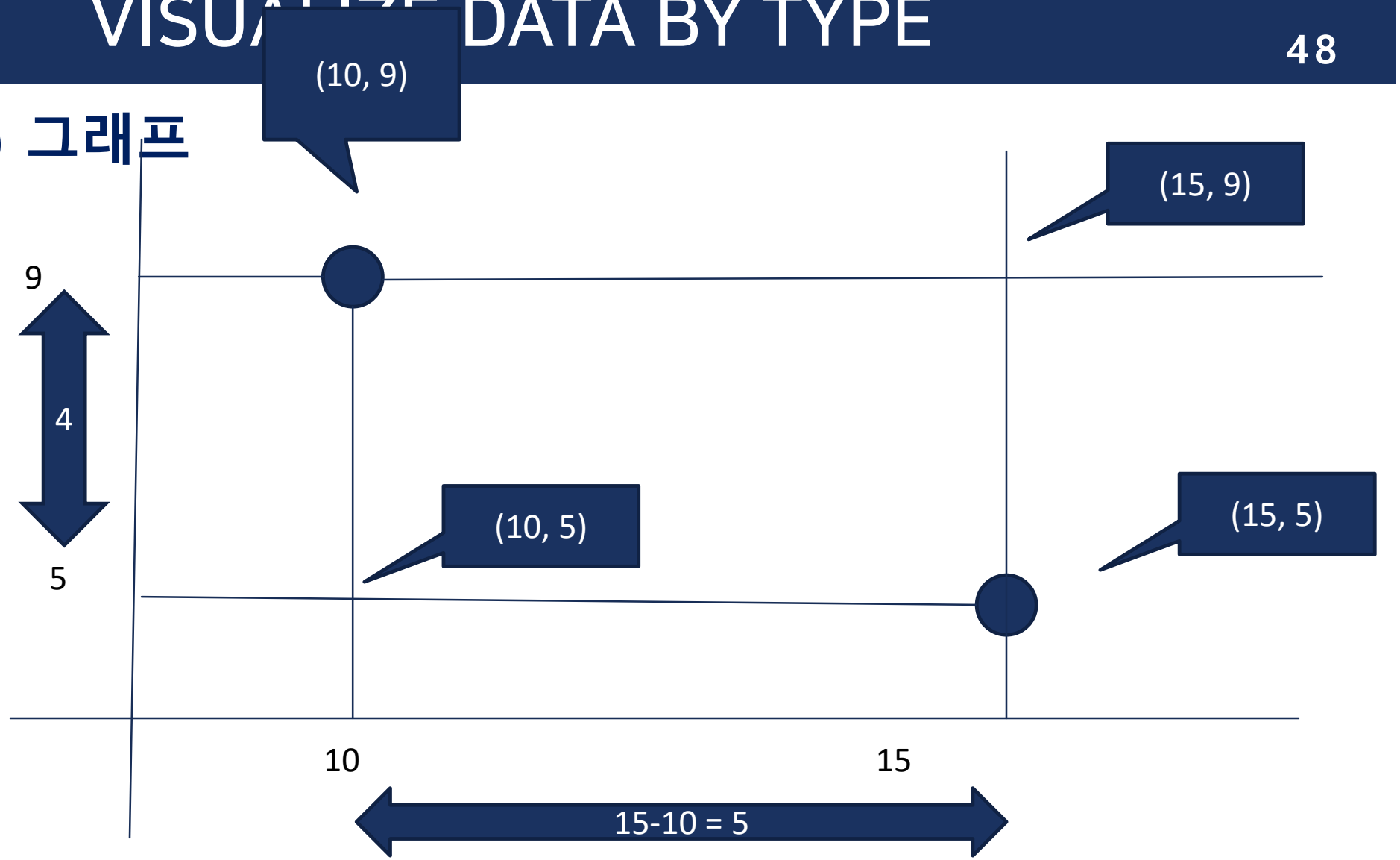
x : 영역 채우기 시작 좌표값, array 타입

y1 : 영역 채우기 끝 좌표값, array 타입

y2 : 두번째 영역 채우기 끝 좌표값, array타입

VISUALIZE DATA BY TYPE

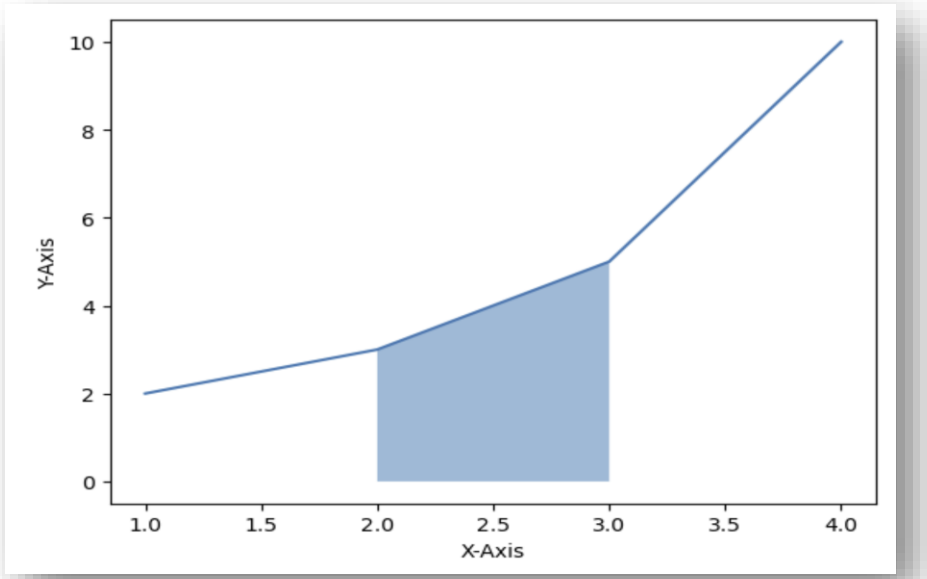
- 영역(Area) 그래프



VISUALIZE DATA BY TYPE

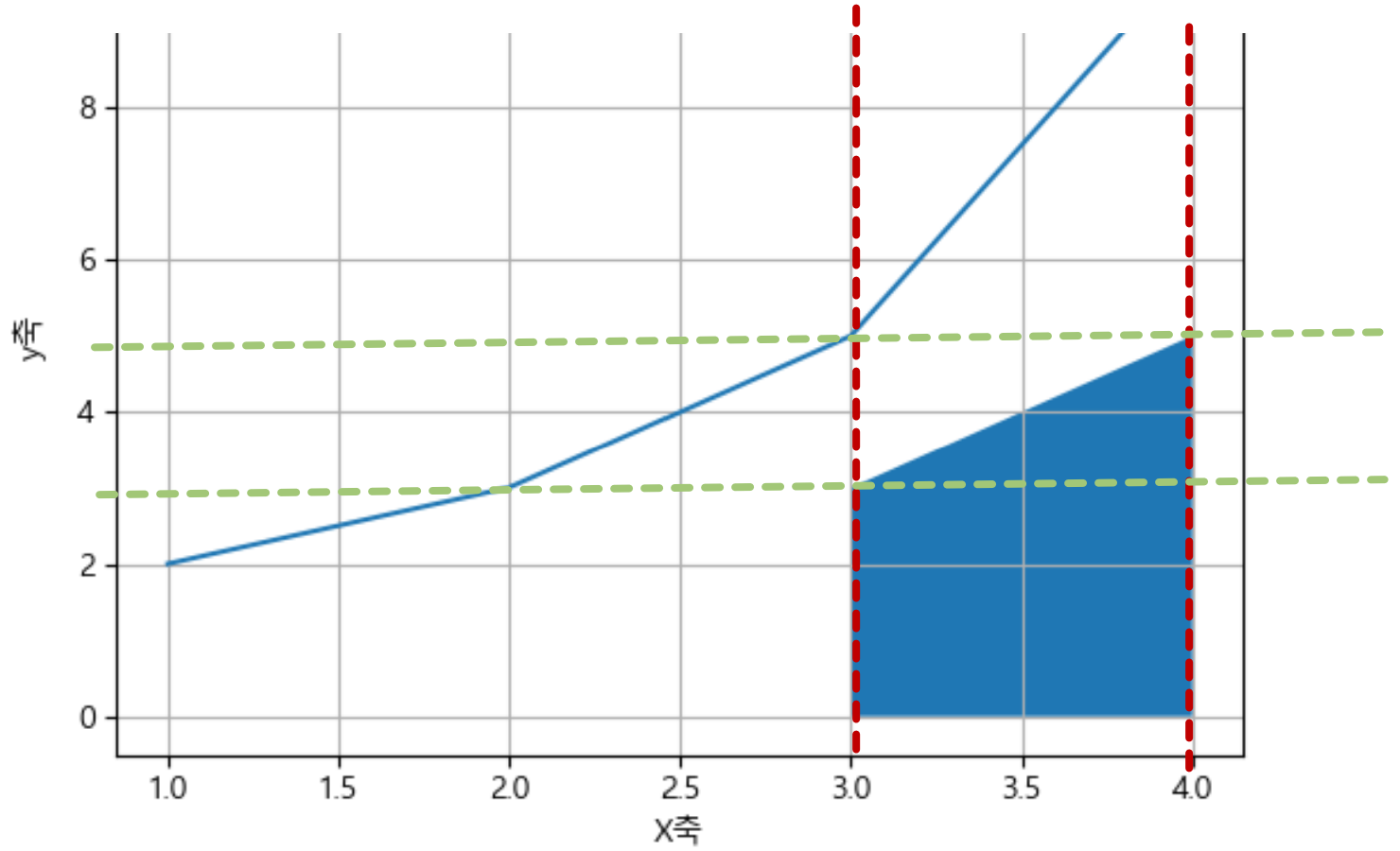
영역(Area) 그래프

```
## 데이터 준비
xData = pd.Series([1, 2, 3, 4])
yData = pd.Series([2, 3, 5, 10])

## 시각화
plt.plot(xData, yData)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')

## 영역 채우기
plt.fill_between(xData[1:3], yData[1:3], alpha=0.5)
## 그래프 그리기
plt.show()
```



VISUALIZE DATA BY TYPE

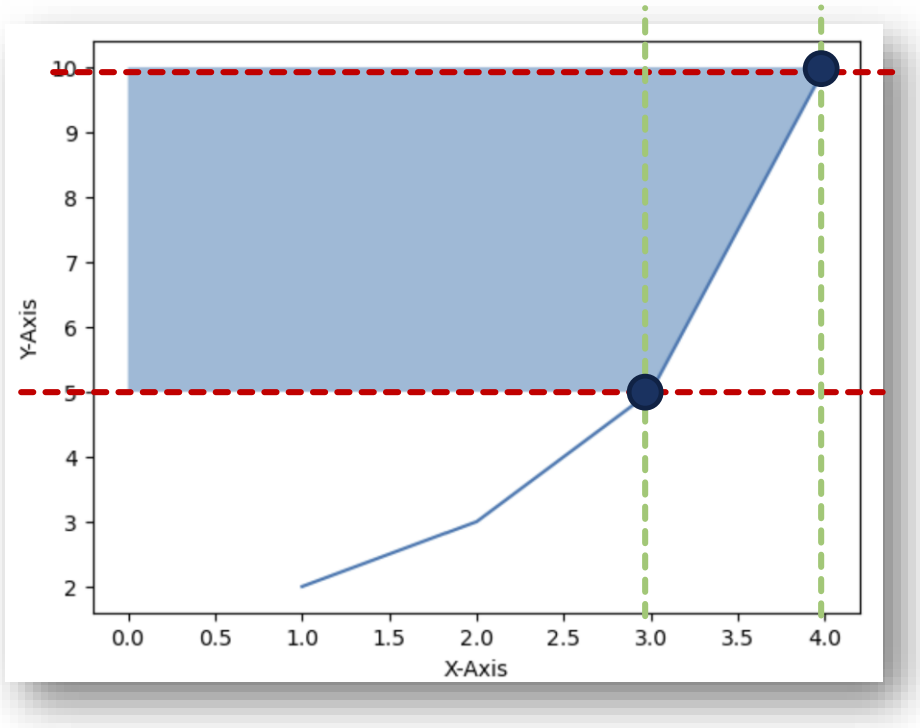
- 영역(Area) 그래프



VISUALIZE DATA BY TYPE

영역(Area) 그래프

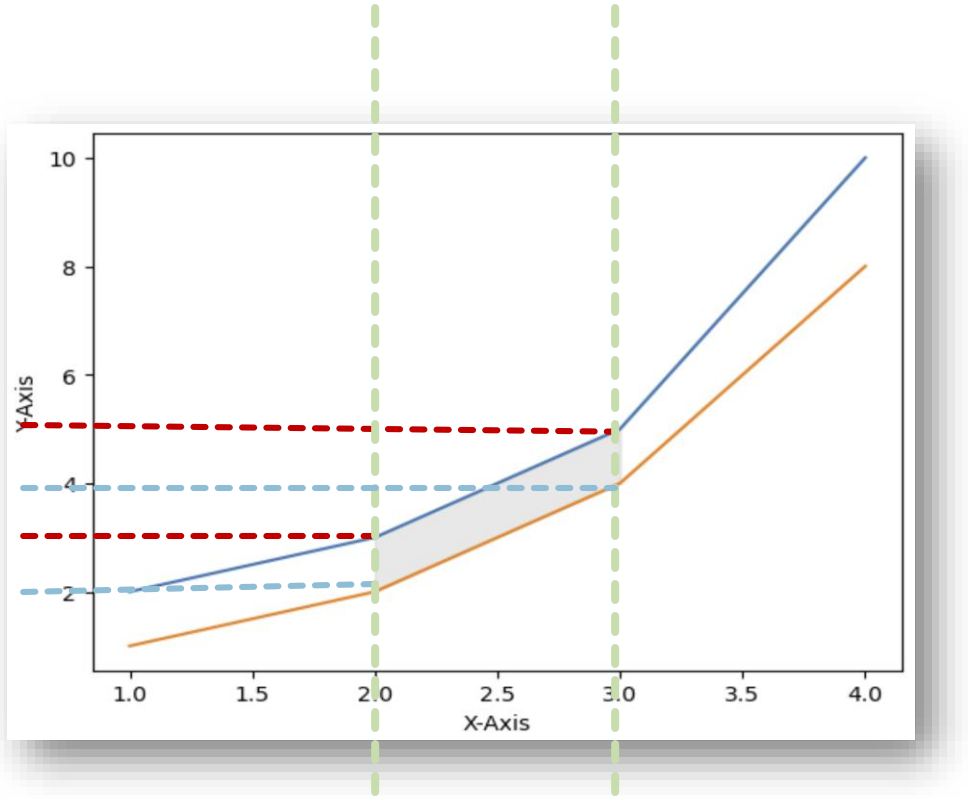
```
## 데이터 준비
xData = pd.Series([1, 2, 3, 4])
yData = pd.Series([2, 3, 5, 10])

## 시각화
plt.plot(xData, yData)

plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')

## 영역 채우기
plt.fill_betweenx( yData[2:4], xData[2:4], alpha=0.5)
## 그래프 그리기
plt.show()
```



VISUALIZE DATA BY TYPE

영역(Area) 그래프

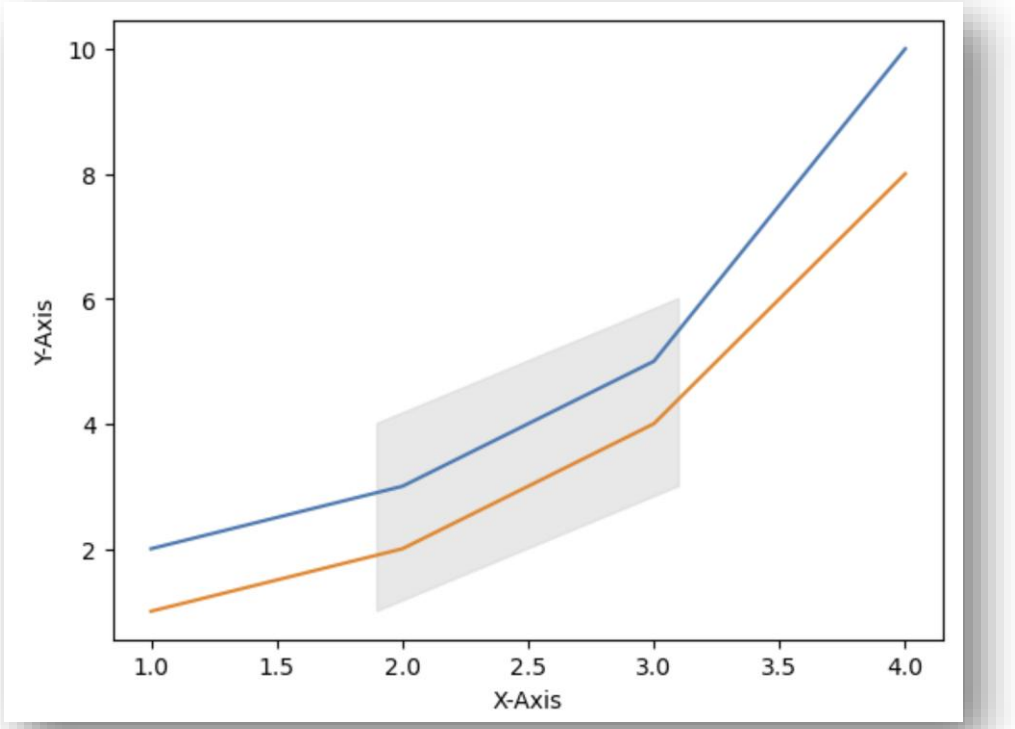
```
# 데이터 생성
xData = pd.Series([1, 2, 3, 4])
yData1 = pd.Series([2, 3, 5, 10])
yData2 = pd.Series([1, 2, 4, 8])

# 시각화
plt.plot(xData, yData1)
plt.plot(xData, yData2)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')

## 영역 채우기
plt.fill_between(xData[1:3], yData1[1:3], yData2[1:3],
                 color='lightgray', alpha=0.5)
## 그래프 그리기
plt.show()
```



VISUALIZE DATA BY TYPE

영역(Area) 그래프

```
# 데이터 생성
xData = pd.Series([1, 2, 3, 4])
yData1 = pd.Series([2, 3, 5, 10])
yData2 = pd.Series([1, 2, 4, 8])

# 시각화
plt.plot(xData, yData1)
plt.plot(xData, yData2)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')

## 영역 채우기
plt.fill( [1.9, 1.9, 3.1, 3.1], [1.0, 4.0, 6.0, 3.0],
          color='lightgray', alpha=0.5 )
## 그래프 그리기
plt.show()
```



VISUALIZE DATA BY TYPE

◆ 산점도(Scatter) 그래프

- **두 변수/데이터 값을 직교 좌표계의 평면에 점으로 표현하는 그래프**
- 용도 : 두 변수/데이터의 **패턴, 상관 관계, 이상치**
- corr()로 상관계수 수치화 확인

[주의]

- 데이터가 너무 많으면 복잡해서 해석 어려움
- 산점도 통해 패턴과 상관관계 예상 가능 ➔ 추가 검증 필요
- 다변량 데이터 관계성 시각화 불가 ➔ 다른 기법 필요

◆ 산점도(Scatter) 그래프

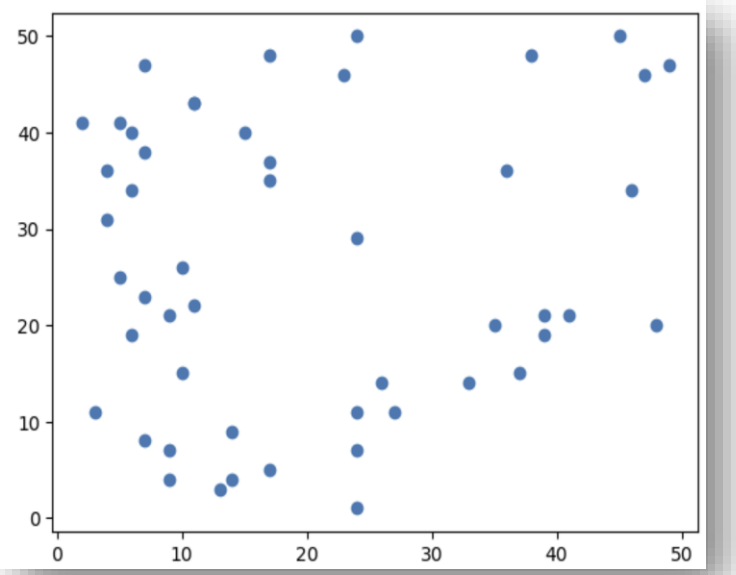
[해석]

- 상관관계 : 점들이 직선형태이면 상관관계 높음(▲)
- 패턴 : 비선형 관계나 군집 등 데이터의 패턴 파악
- 이상치 : 다른 점들과 멀리 떨어진 점들을 이상치로 간주 가능

VISUALIZE DATA BY TYPE

산점도(Scatter) 그래프

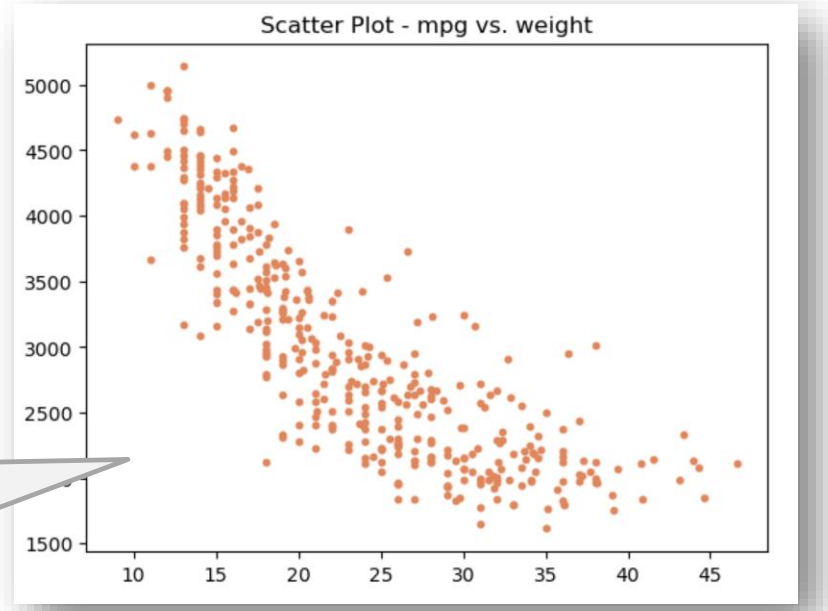
```
## 모듈 로딩
import matplotlib.pyplot as plt
import random
import pandas as pd

## 데이터 준비
xData=pd.Series([random.randint(1,50) for _ in range(50) ])
yData=pd.Series([random.randint(1,50) for _ in range(50) ])


## 시각화
plt.scatter( xData, yData )
plt.show( )
```



VISUALIZE DATA BY TYPE

산점도(Scatter) 그래프

```
# 연비(mpg)와 차중(weight) 열에 대한 산점도 그리기
xData = df['mpg']
yData = df['weight']

# 시각화
plt.scatter(xData, yData,  c= ' coral ' ,  s=10)
plt.title('Scatter Plot - mpg vs. weight')
plt.show( )
```

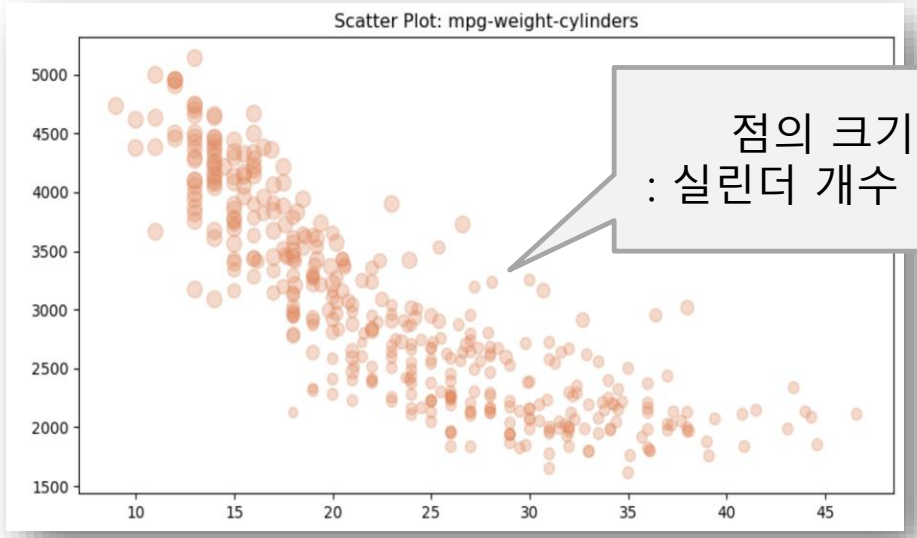


점들의 직선 형태의 분포
mpg와 weight 반비례 관계

VISUALIZE DATA BY TYPE

산점도(Scatter) 그래프

```
# cylinders 개수의 상대적 비율을 계산하여 시리즈 생성
cylinders_size = (df['cylinders'] / df['cylinders'].max()) * 100
xData = df['mpg']
yData = df['weight']


# 3개의 변수로 산점도 그리기
plt.figure(figsize=(10,5))
plt.scatter(xData, yData,     c='coral',
                s=cylinders_size, alpha=0.3)
plt.title('Scatter Plot: mpg-weight-cylinders')
plt.show( )
```



VISUALIZE DATA BY TYPE

산점도(Scatter) 그래프

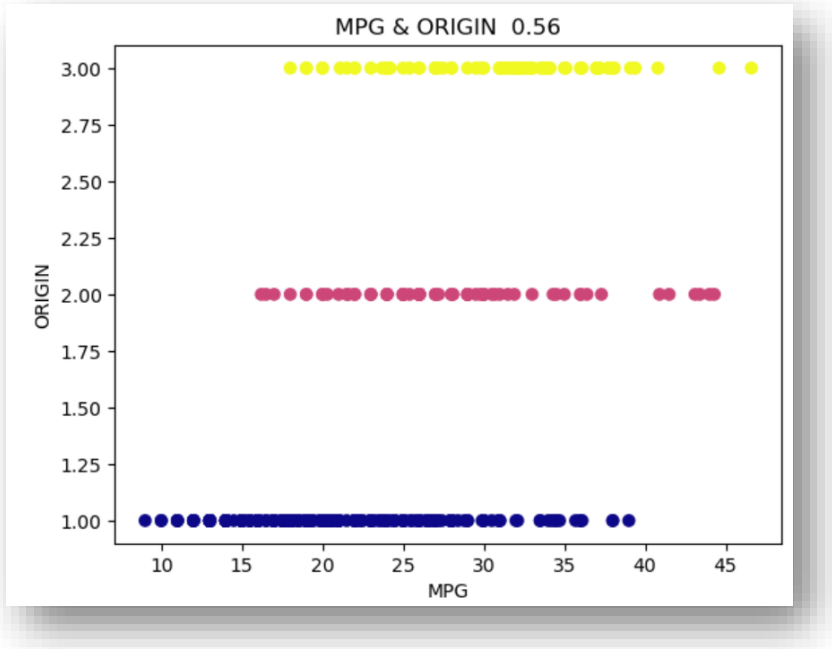
```
## - 상관계수값 추출
corrSR=dataDF.corr(numeric_only=True)['mpg']
corrSR
```

```
mpg             1.000000
cylinders      -0.775396
displacement   -0.804203
weight         -0.831741
acceleration    0.420289
model year      0.579267
origin          0.563450
Name: mpg, dtype: float64
```

VISUALIZE DATA BY TYPE

산점도(Scatter) 그래프

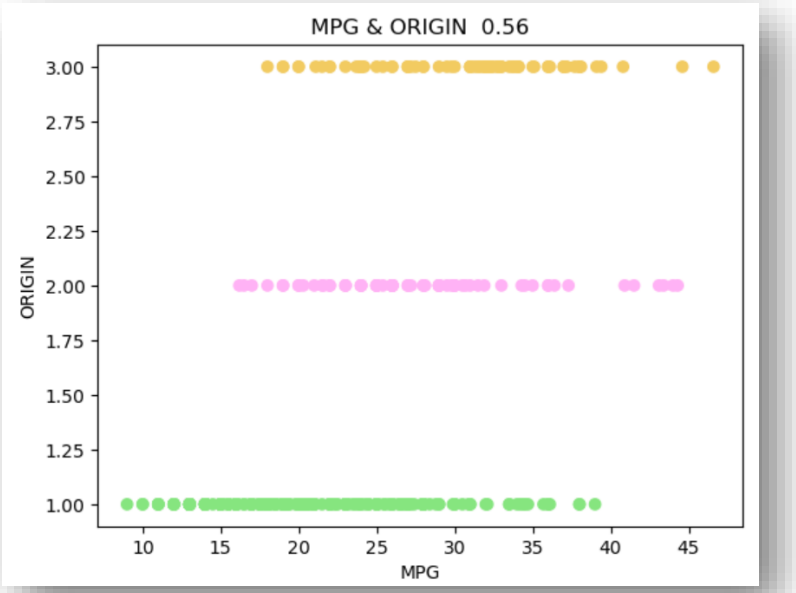
```
## - mpg와 origin 컬럼의 관계성
xData=dataDF['mpg']
yData=dataDF['origin']

## 관계성 시각화
plt.scatter(xData, yData, c=yData, cmap='plasma')
plt.title(f"MPG & ORIGIN  {corrSR['origin']:.2f}")
plt.xlabel('MPG')
plt.ylabel('ORIGIN')
plt.show( )
```



VISUALIZE DATA BY TYPE

산점도(Scatter) 그래프

```
## - mpg와 origin 컬럼의 관계성
cData=dataDF['origin'].replace({1:'#86E57F', 2:'#FFB2F5', 3:'#F2CB61'})
xData=dataDF['mpg']
yData=dataDF['origin']

## 관계성 시각화
plt.scatter(xData, yData, c=cData)
plt.title(f"MPG & ORIGIN  {corrSR['origin']:.2f}")
plt.xlabel('MPG')
plt.ylabel('ORIGIN')
plt.show( )
```



VISUALIZE DATA BY TYPE

파이(Pie) 그래프

- **전체 데이터에 대한 데이터의 비율** 표현
- 정당별 의원수, 지지율, 특정 지역 인구 비율, 서비스 만족도 등 **세부 항목 간 비율 분석**
- **부채꼴의 중심각을 구성 비율에 비례**하도록 표현
- **활용 ➔항목 비율 / 항목 간 상대적 크기 비교**

파이(Pie) 그래프

```
pie( x, *, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6,
     shadow=False, labeldistance=1.1, startangle=0, radius=1, counterclock=True,
     wedgeprops=None, textprops=None, center=(0, 0), frame=False, rotatelabels=False,
     normalize=True, hatch=None, data=None )
```

파이(Pie) 그래프

매개변수		
x	입력	1D 배열 데이터
explode	None	부채꼴이 파이 그래프의 중심에서 벗어나는 정도를 설정
labels	None	
colors	None	
autopct	None	부채꼴 안에 표시될 숫자의 형식을 지정
pctdistance	0.6	
labeldistance	1.1	
startangle	0	부채꼴이 그려지는 시작 각도를 설정, 양의 방향 x축 설정
radius	1	
counterclock	True	데이터 배치 방향 설정, True - 반시계 방향 기본값, False – 시계 방향

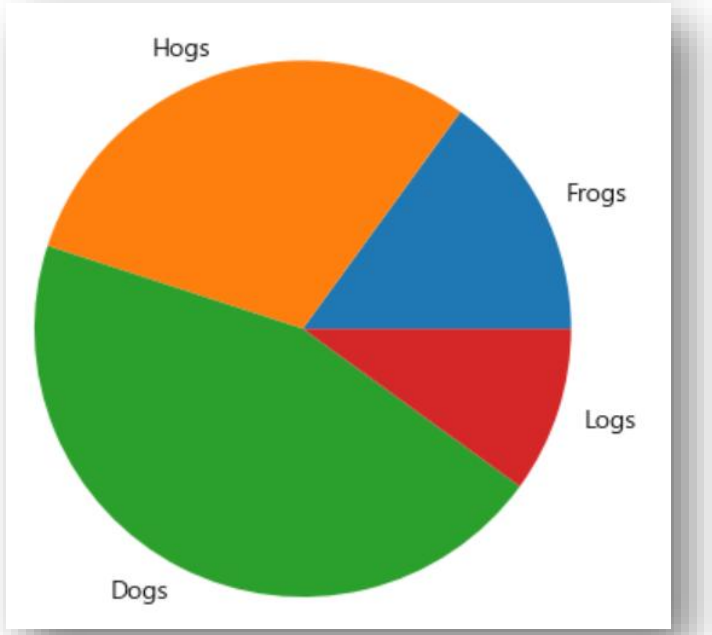
파이(Pie) 그래프

매개변수		
wedgeprops	None	부채꼴 영역의 matplotlib.patches.Wedge 클래스 속성으로 스타일 설정. 'width', 'edgecolor', 'linewidth' 키 이용 => 부채꼴 영역 너비(반지름에 대한 비율), 테두리 색상, 테두리 선 너비 설정 [예] wedgeprops={'width': 0.7, 'edgecolor': 'w', 'linewidth': 5}
textprops	None	텍스트 스파일 설정
center	(0,0)	중심 좌표 값
frame	False	파이 그래프에 사각형 프레임 그려주기 여부 설정
rotatelabels	False	라벨 표시
normalize	True	x 값의 합계를 1로 설정

VISUALIZE DATA BY TYPE

파이(Pie) 그래프

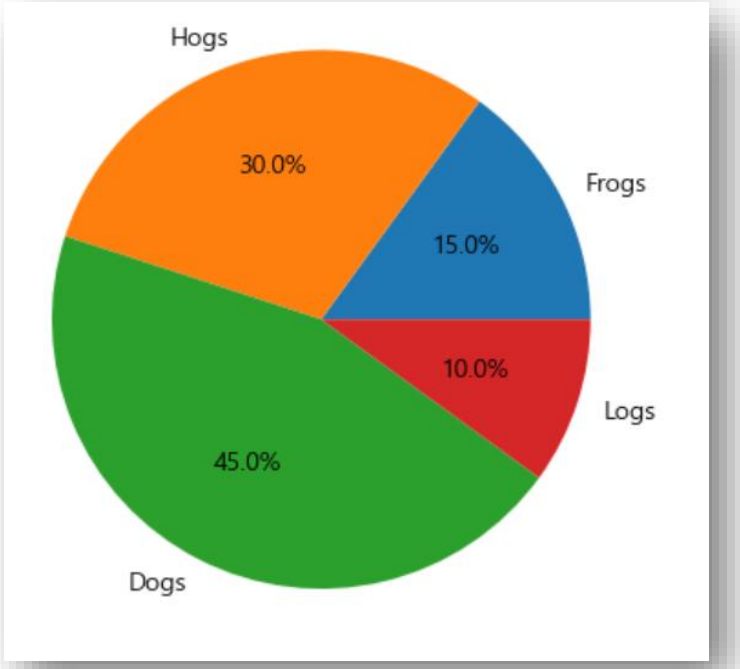
```
## 데이터 준비
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]

## 시각화
plt.pie(sizes, labels=labels)
plt.show()
```



VISUALIZE DATA BY TYPE

파이(Pie) 그래프

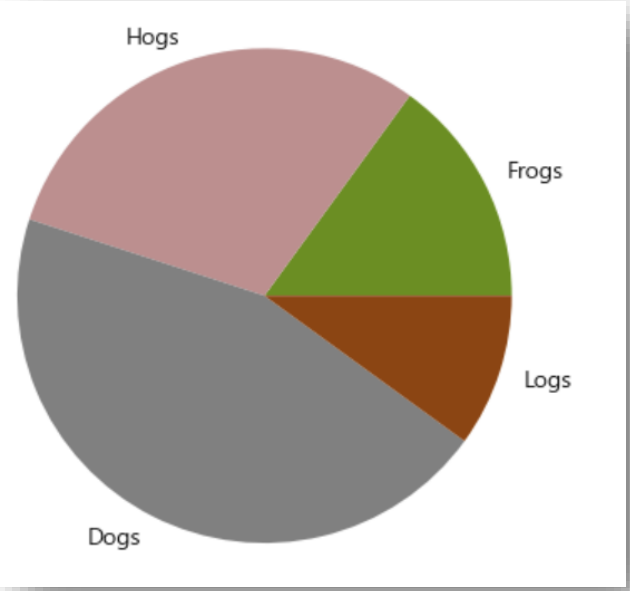
```
## 데이터 준비
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]

## 시각화
## 조각별 비율 출력 => autopct 매개변수 : 00.00%
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.show()
```



VISUALIZE DATA BY TYPE

파이(Pie) 그래프

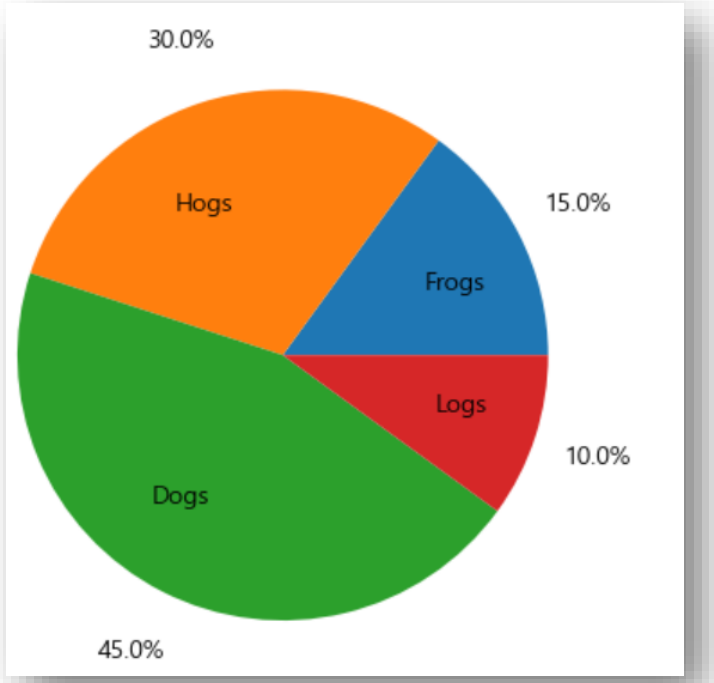
```
## 데이터 준비
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]

## 시각화
## 커스텀 색상 설정 =>
    colors 매개변수 : [ 조각별 원하는 색상 리스트 ]
plt.pie(sizes, labels=labels,
        colors=['olivedrab', 'rosybrown', 'gray', 'saddlebrown'])
plt.show()
```



VISUALIZE DATA BY TYPE

파이(Pie) 그래프

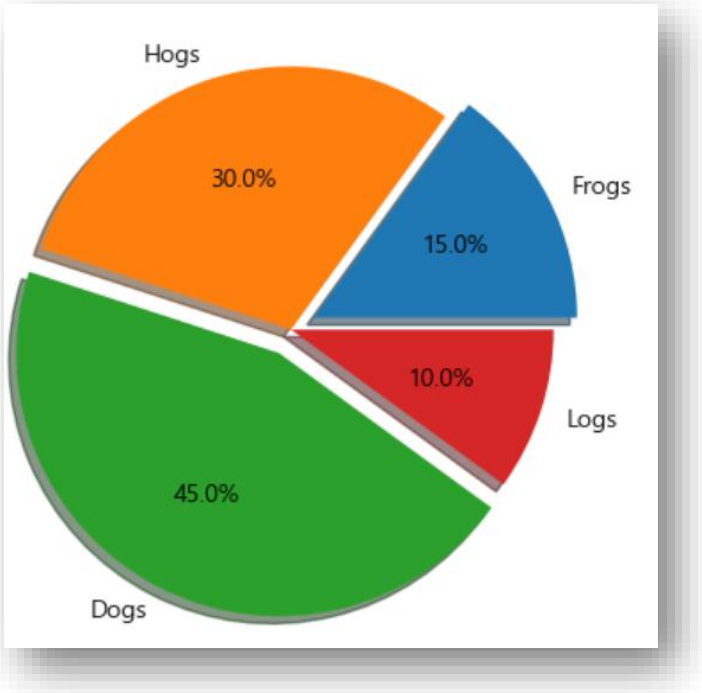
```
## 시각화
## 라벨과 퍼센트 출력 위치 설정
## => pctdistance 매개변수   : 값 출력 위치 설정 [기]0.6
## => labeldistance 매개변수 : 라벨 출력 위치 설정 [기]1.1
plt.pie( sizes,
         labels=labels,
         autopct='%1.1f%%',
         pctdistance=1.25,
         labeldistance=.6 )

plt.show()
```



VISUALIZE DATA BY TYPE

파이(Pie) 그래프

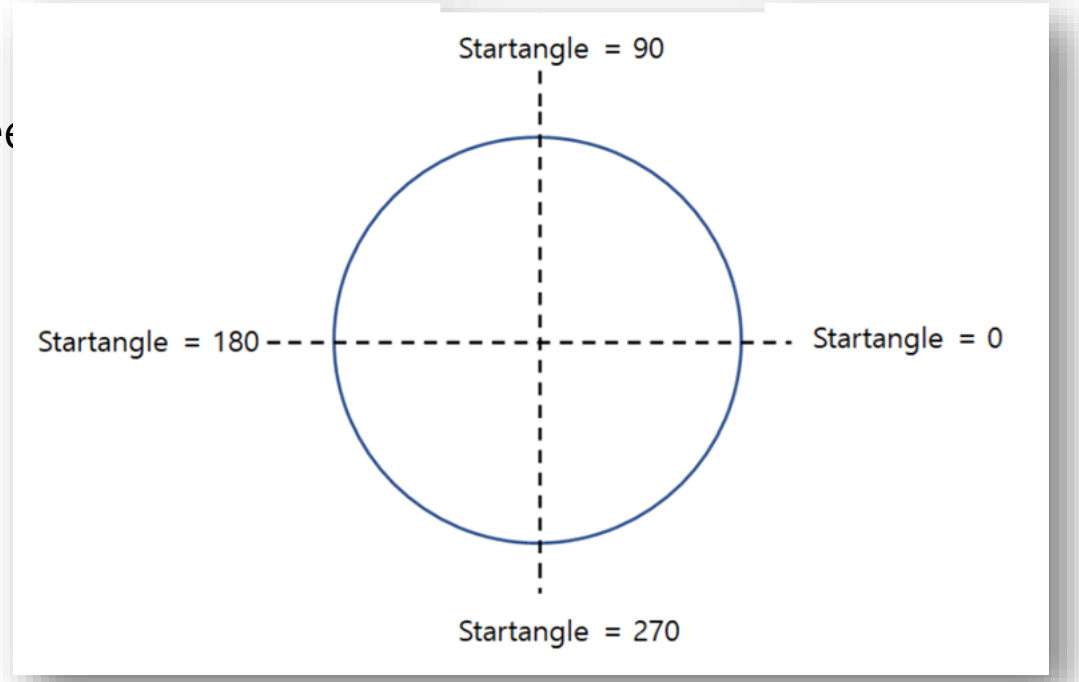
```
## 시각화
## explode 매개변수 : 중심점에서의 떨어지는 값
## shadow 매개변수 : 그림자 설정
## labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
explode = (0.1, 0,0.1,0)

plt.pie( sizes,
         explode=explode,
         labels=labels,
         autopct='%1.1f%%',
         shadow=True)
plt.show()
```



파이(Pie) 그래프

매개변수

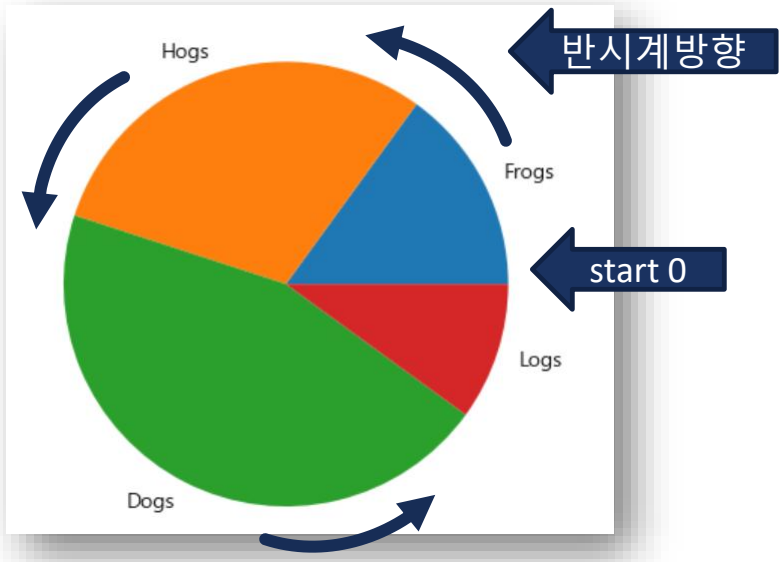
startangle : 파이 그래프 시작점 각도(degree

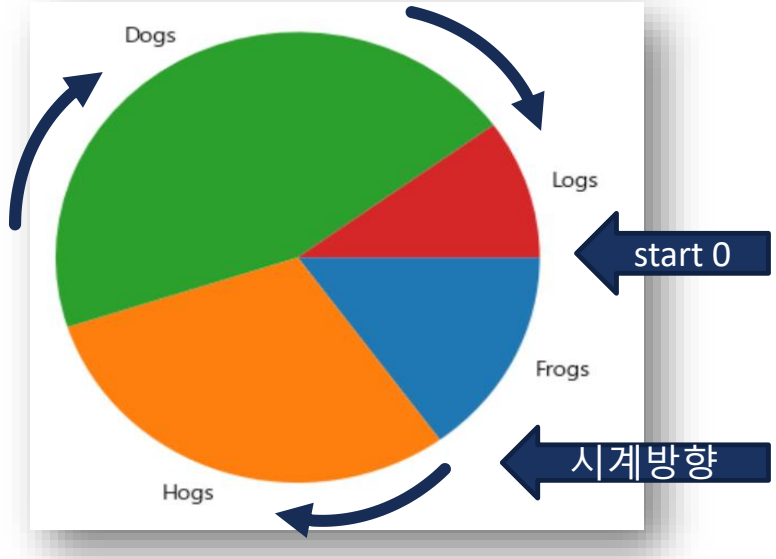


파이(Pie) 그래프

매개변수

counterclock : 배치 방향, counterclockwise CCW 반시계 / colockwise CW 시계 방향

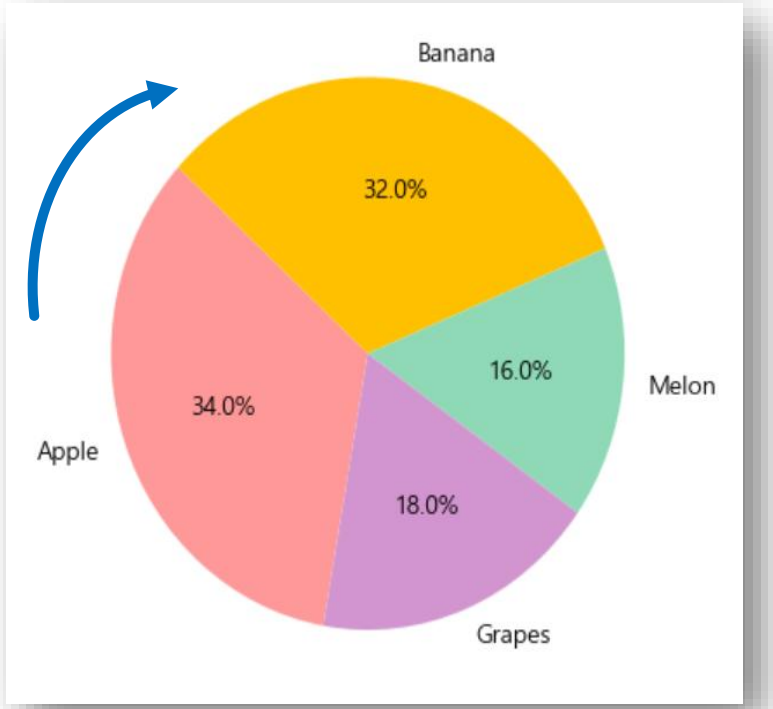




VISUALIZE DATA BY TYPE

파이(Pie) 그래프

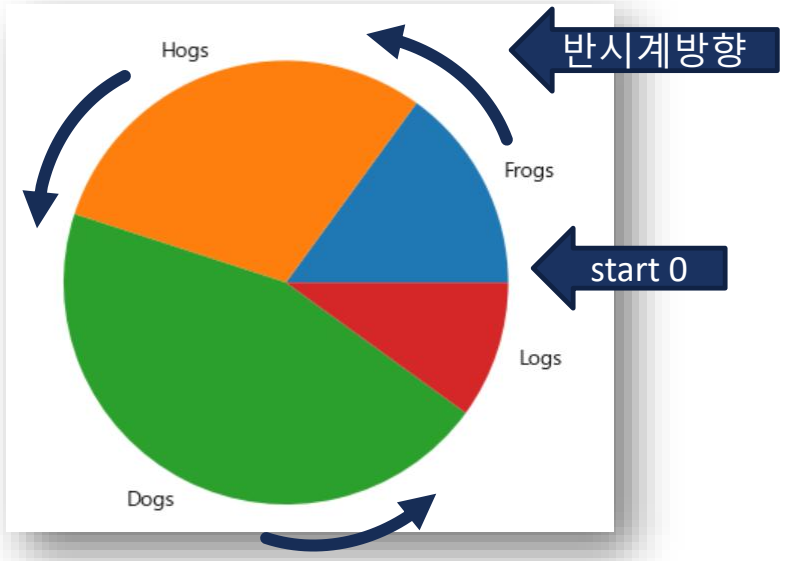
```
## 시각화
## startangle     매개변수 : 부채꼴 그려지는 시작 각도
## counterclock 매개변수 :  False 시계 방향
ratio    = [34, 32, 16, 18]
labels   = ['Apple', 'Banana', 'Melon', 'Grapes']
colors   = ['#ff9999', '#ffc000', '#8fd9b6', '#d395d0']

plt.pie( ratio, labels=labels, autopct='%.1f%%',
         startangle=260, counterclock=False,
         colors=colors)
plt.show()
```

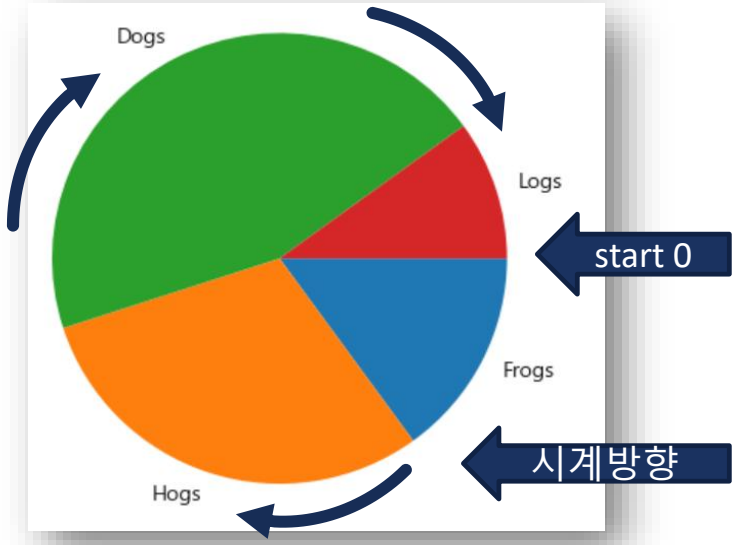


파이(Pie) 그래프

매개변수

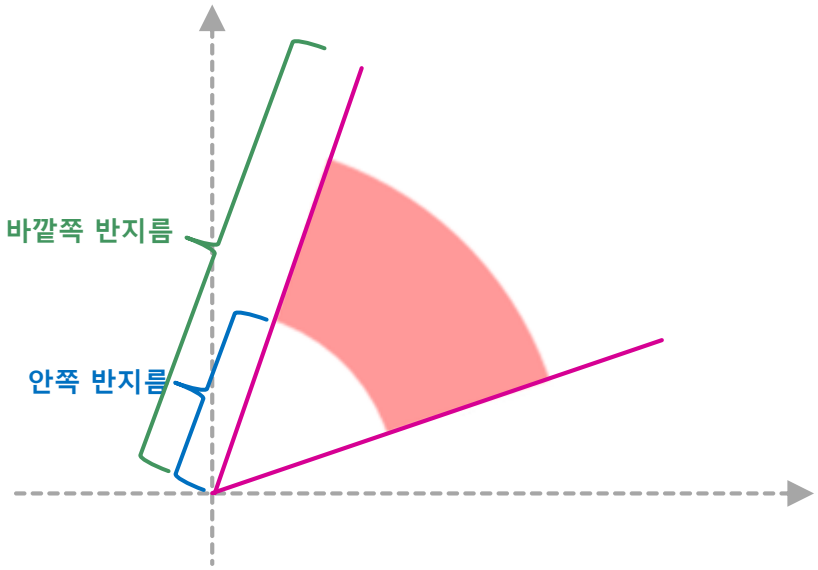
counterclock : 배치 방향, counterclockwise CCW 반시계 / colockwise CW 시계 방향





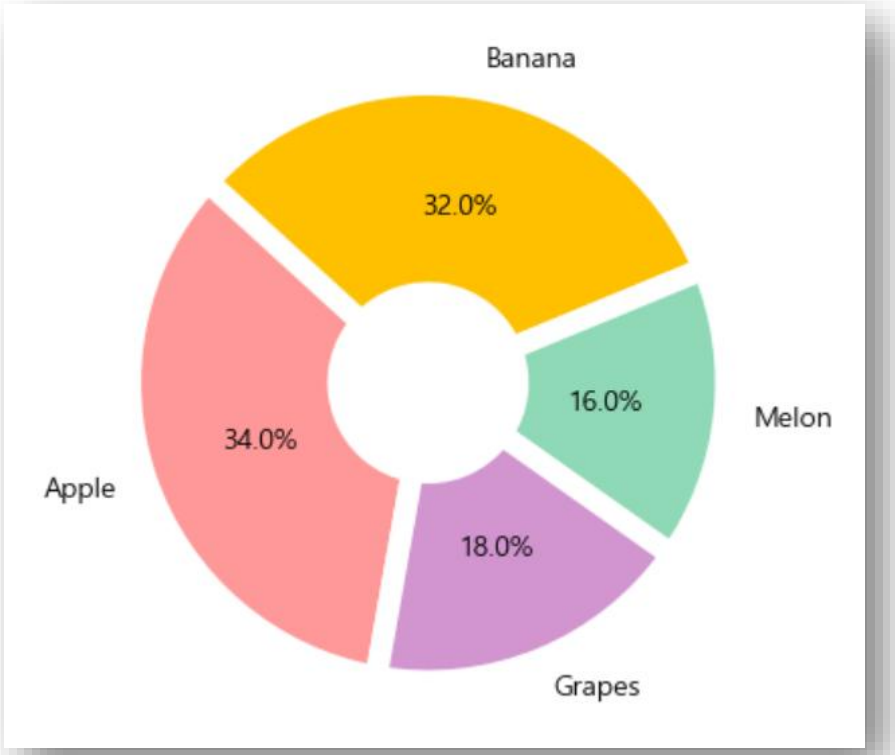
파이(Pie) 그래프

```
## 시각화
## wedgeprops 매개변수 :  쐐기 생성
##  - width  :  쐐기 너비, 바깥쪽 너비
               바깥쪽 반지름 – 안쪽 반지름
wedgeprops={'width': 0.7,
            'edgecolor': 'w',
            'linewidth': 8}
```



파이(Pie) 그래프

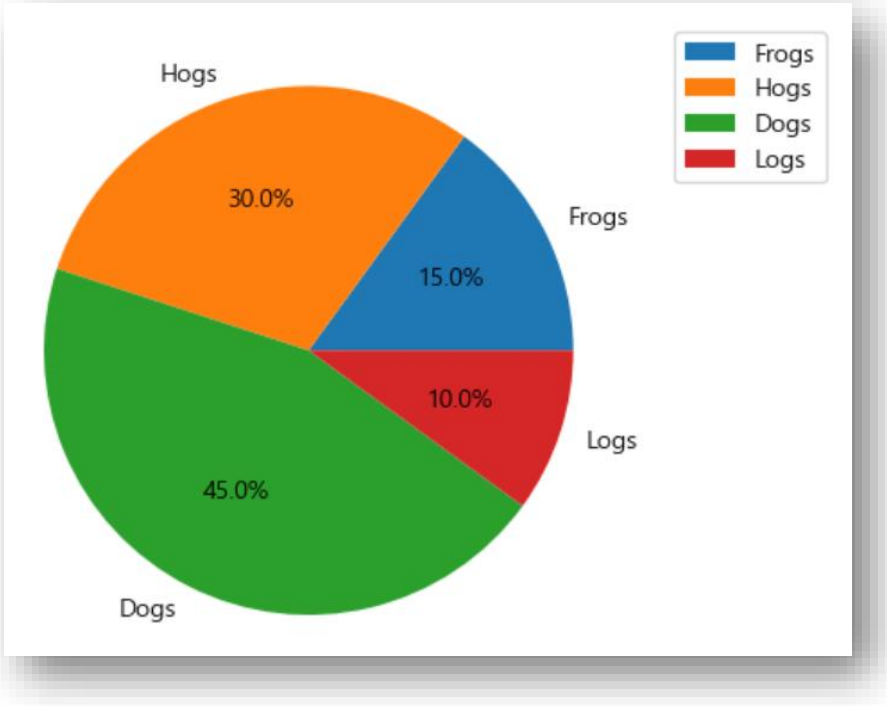
```
## 시각화
## wedgeprops 매개변수 : 쐐기 생성
##  - width  :  쐐기 너비, r – width ~  r까지
wedgeprops={'width': 0.7,
            'edgecolor': 'w',
            'linewidth': 8}

plt.pie( ratio, labels=labels, autopct='%.1f%%',
         startangle=260, counterclock=False,
         colors=colors,  wedgeprops=wedgeprops)
plt.show()
```



VISUALIZE DATA BY TYPE

파이(Pie) 그래프

```
## 범례
## bbox_to_anchor 매개변수 : 범례 박스 위치 설정
## x, y >1.0 이상 이면 프레임 밖에 위치
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.legend(loc='upper right',
           bbox_to_anchor=(1.3, 1))
plt.show()
```



VISUALIZE DATA BY TYPE

에러 바(Errorbar) 그래프

- 설문조사나 실험을 통해 **얻어진 값의 오차를 표현**
- **데이터의 편차를 표시하기 위한 그래프 형태**
- 평균값이나 중앙값과 같은 대표값 주위에 수직 또는 수평 선으로 표시
- 선의 길이가 데이터의 분산 정도를 나타냄
- **데이터/값의 신뢰도나 변동성, 불확실성을 시각적으로 직관적으로 표현**

- 짧은 에러바 : 높은 데이터 신뢰도
- 긴 에러바 : 높은 데이터 불확실성

VISUALIZE DATA BY TYPE

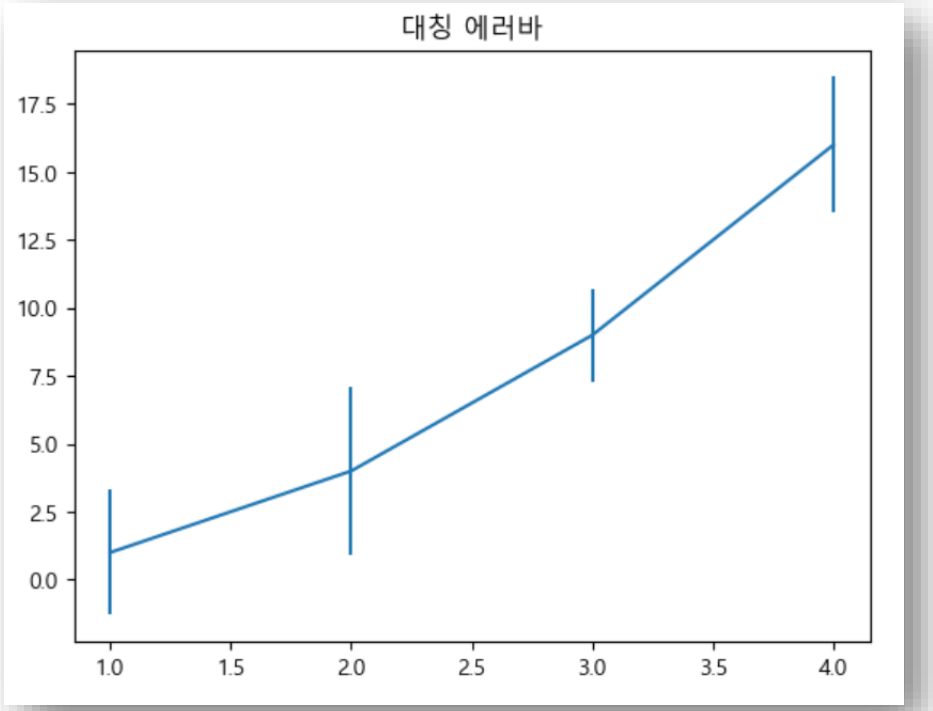
에러 바(Errorbar) 그래프

- **주식 시장 분석** : 월별 주가 변동성 표현
- **과학 실험** : 측정값의 정확도 평가
- **비즈니스 데이터** : 시장 조사 결과의 신뢰 구간 표시

- **겹치는 에러바** : 데이터 포인트 간 차이가 유의미하지 않을 수 있음
- **겹치지 않는 에러바** : 데이터 포인트 간 차이가 유의미할 가능성 높음

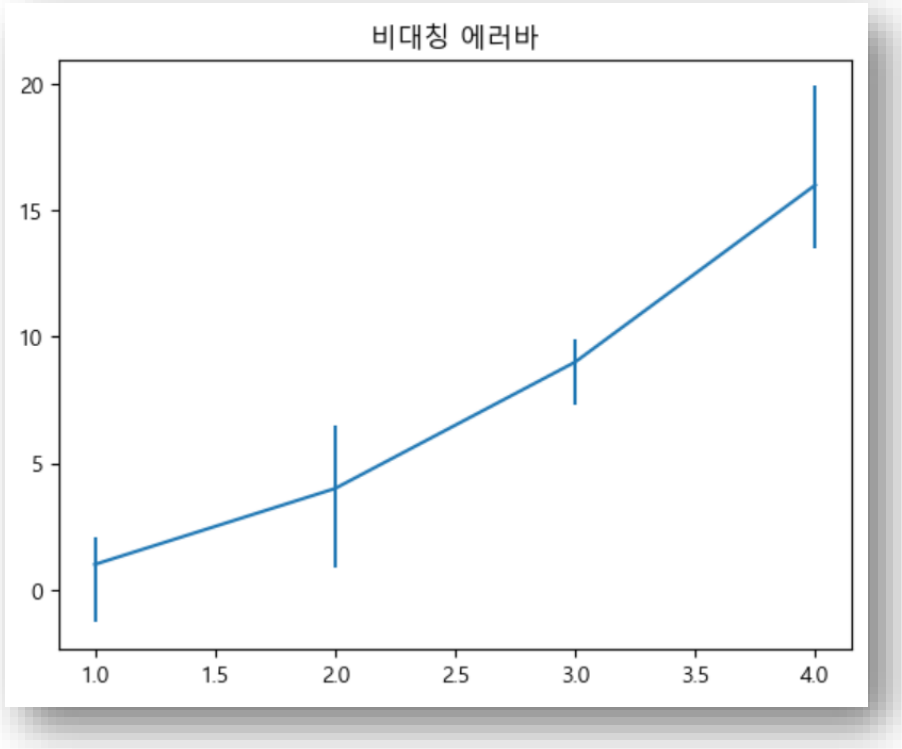
에러 바(Errorbar) 그래프

```
## 데이터 준비
x = [1, 2, 3, 4]
y = [1, 4, 9, 16]

## 데이터 편차
yerr = [2.3, 3.1, 1.7, 2.5]

## 시각화
plt.errorbar(x, y, yerr=yerr)
plt.title('대칭 에러바')
plt.show()
```



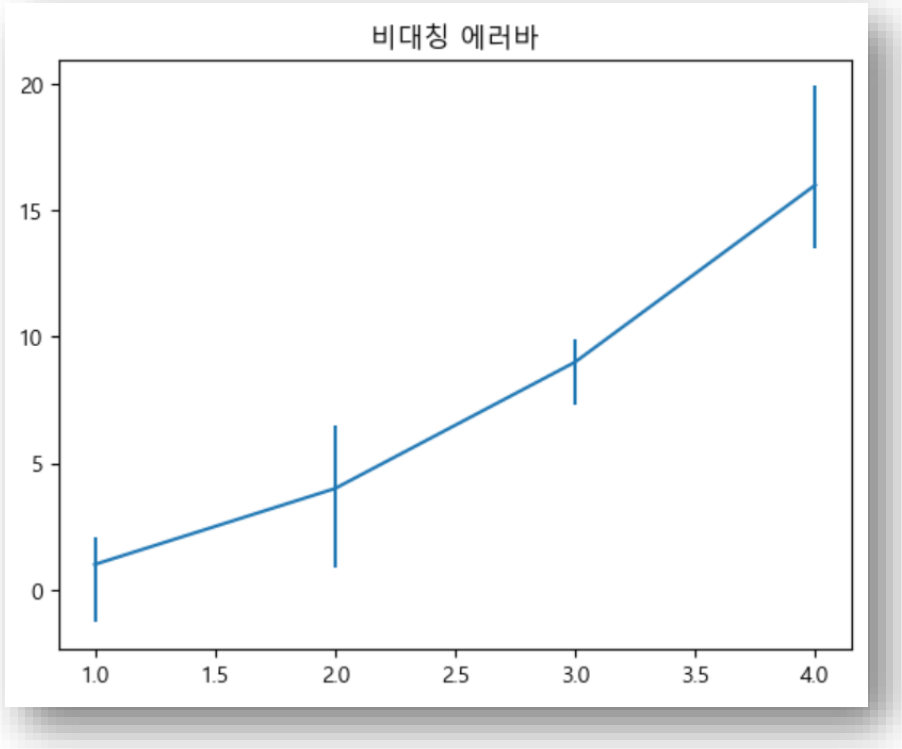
에러 바(Errorbar) 그래프

```
## 데이터 준비
x = [1, 2, 3, 4]
y = [1, 4, 9, 16]

## 데이터 편차
##        아래 방향 편차      위 방향 편차
yerr = [(2.3, 3.1, 1.7, 2.5), (1.1, 2.5, 0.9, 3.9)]

## 시각화
plt.errorbar(x, y, yerr=yerr)
plt.title('비대칭 에러바')
plt.show()
```



에러 바(Errorbar) 그래프

```
## 데이터 준비
x = [1, 2, 3, 4]
y = [1, 4, 9, 16]

## 데이터 편차
##        아래 방향 편차      위 방향 편차
yerr = [(2.3, 3.1, 1.7, 2.5), (1.1, 2.5, 0.9, 3.9)]

## 시각화
plt.errorbar(x, y, yerr=yerr)
plt.title('비대칭 에러바')
plt.show()
```



VISUALIZE DATA BY TYPE

히스토그램(Historgram) 그래프

- **수치형 데이터의 분포를 시각적으로 표현해주는 그래프**
- 도수분포표(Frequency Table)를 시각적으로 표현한 막대 그래프
- 가로축은 계급(보통 변수 구간), 세로축은 도수(해당 계급/구간에 속하는 데이터 수의 비율)
- **수집 데이터의 전체 분포 확인 가능 ➔ 대칭/비대칭/이상치/패턴**

◆ 히스토그램(Historgram) 그래프

- **도수분포표(Frequency Table)**

  → 특정 구간에 속하는 자료의 개수를 나타내는 표

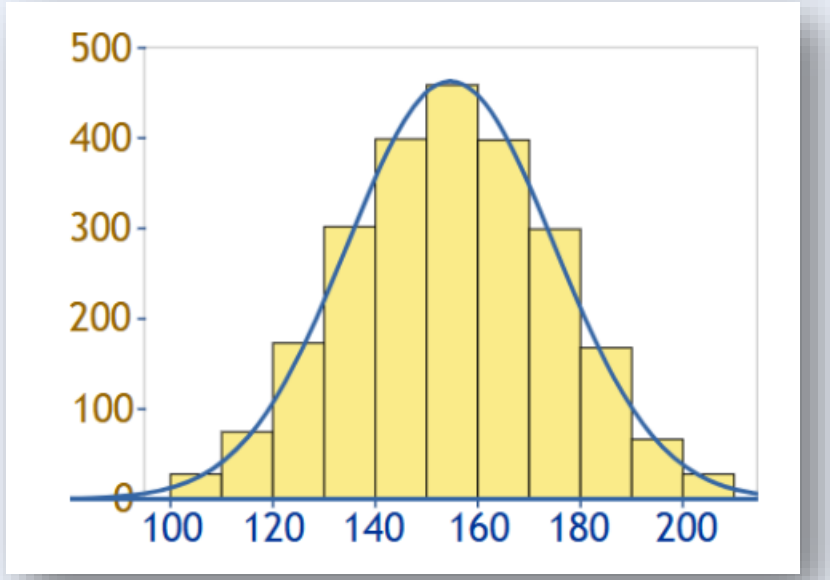
  - 전체 데이터 중 최소값, 최대값 찾기
  - 몇 개 구간으로 나눌지 결정
  - 구간폭 설정

성적 구간(계급)	학생 수(도수)
0점 ~ 20점	1
20점 ~ 40점	3
40점 ~ 60점	11
60점 ~ 80점	20
80점 ~ 100점	5
합계	40

히스토그램(Historgram) 그래프

- **정규 분포형 히스토그램 (Normal Distribution Histogram)**

  → 가운데 부분이 높고 양쪽으로 대칭되는 종 모양의 분포

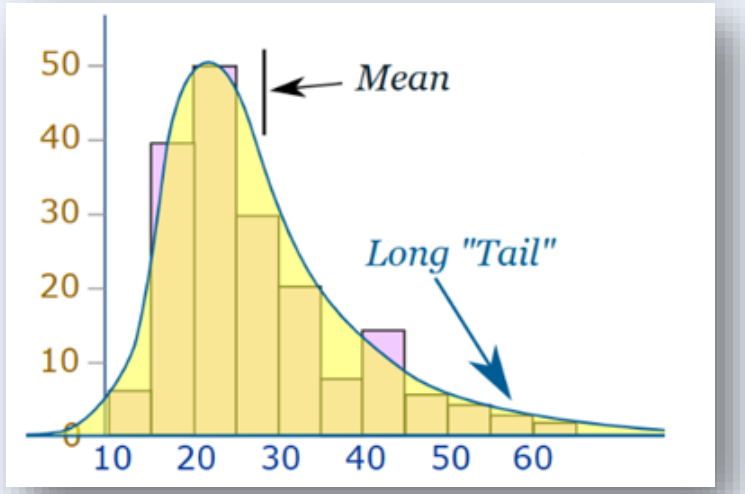


히스토그램(Historgram) 그래프

- **치우침 분포형 히스토그램 (Skewed Distribution Histogram)**

  → 데이터가 한쪽으로 치우쳐 있는 경우, 좌측 또는 우측으로 긴 꼬리를 가진 비대칭형

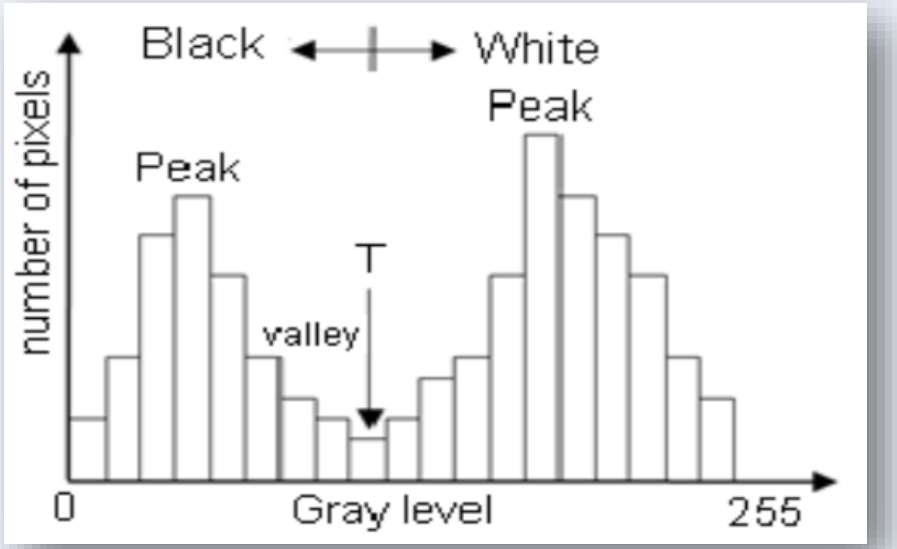
  예) 소득 분포에서 저소득자가 많은 경우 좌측으로 치우친 분포



히스토그램(Historgram) 그래프

- **이중봉형 히스토그램 (Bimodal Histogram)**

  → 두 개의 봉우리를 가진 분포로, 두 개의 뚜렷한 그룹이 있는 데이터에서 발생

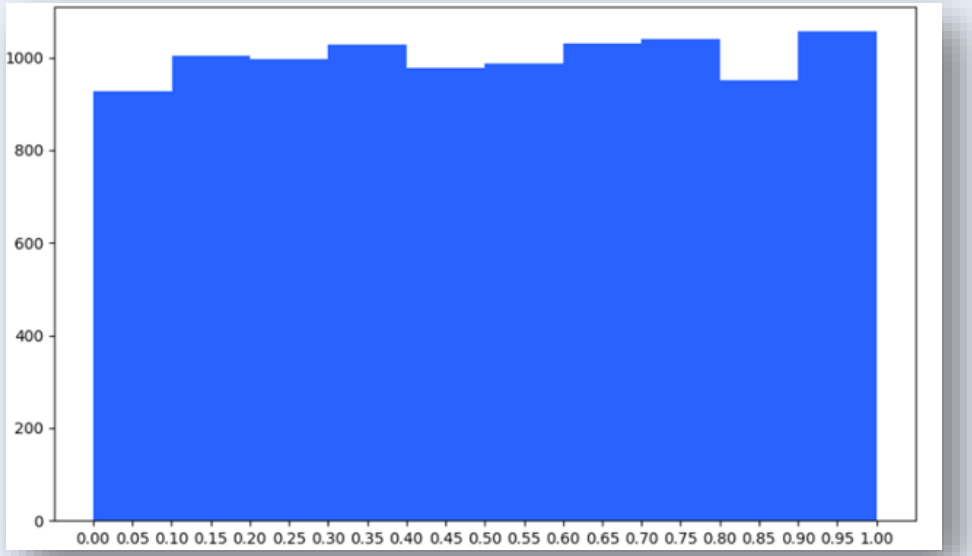


VISUALIZE DATA BY TYPE

◆ 히스토그램(Historgram) 그래프

▪ 균등 분포형 히스토그램 (Uniform Distribution Histogram)

→ 각 구간에 속하는 데이터의 빈도수가 거의 일정할 때 나타나는 분포

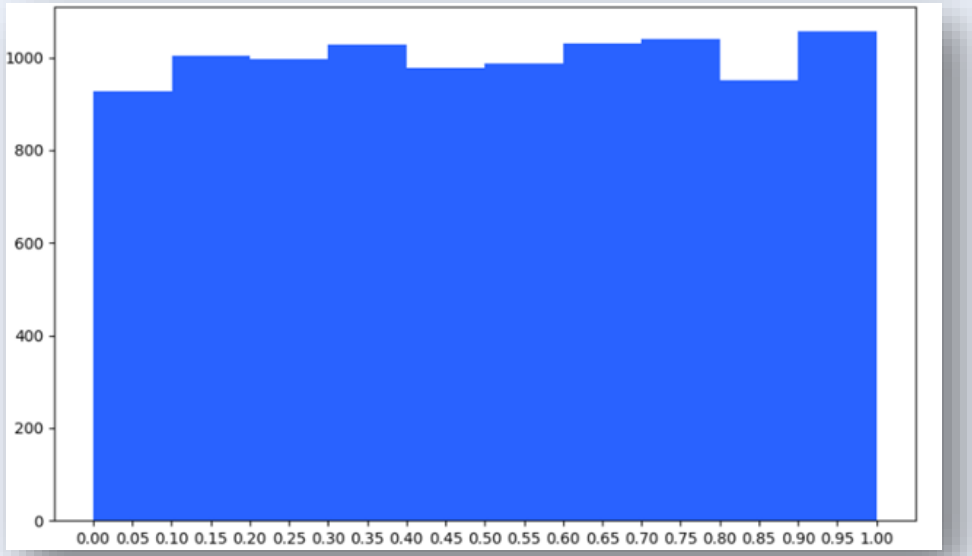


VISUALIZE DATA BY TYPE

히스토그램(Historgram) 그래프

- **균등 분포형 히스토그램 (Uniform Distribution Histogram)**

  → 각 구간에 속하는 데이터의 빈도수가 거의 일정할 때 나타나는 분포



◆ 히스토그램(Historgram) 그래프

- **왜도(skewness)**
  - → **데이터의 좌우 쏠림을 수치로 나타내는 척도**
  - → 분석 결과의 정확도가 떨어지거나 오류가 발생할 수 있음
  - → **분포가 대칭적이면 왜도는 0에 가까움**
  - → 보　완 : **EDA 진행하면서 LOG변환 등을 통해 정규분포에 가깝게 변환할 것인지 판단**
  - → 문제점 : 데이터가 양쪽 극단에 치우쳐져 있거나 이상점(outlier)이 포함되어 있는 경우 대칭분포에 비해 큰 값으로 나타날 수 있음. 분포 모양을 왜곡시킬 수 있음
    첨도와 함께 고려해서 분포 평가

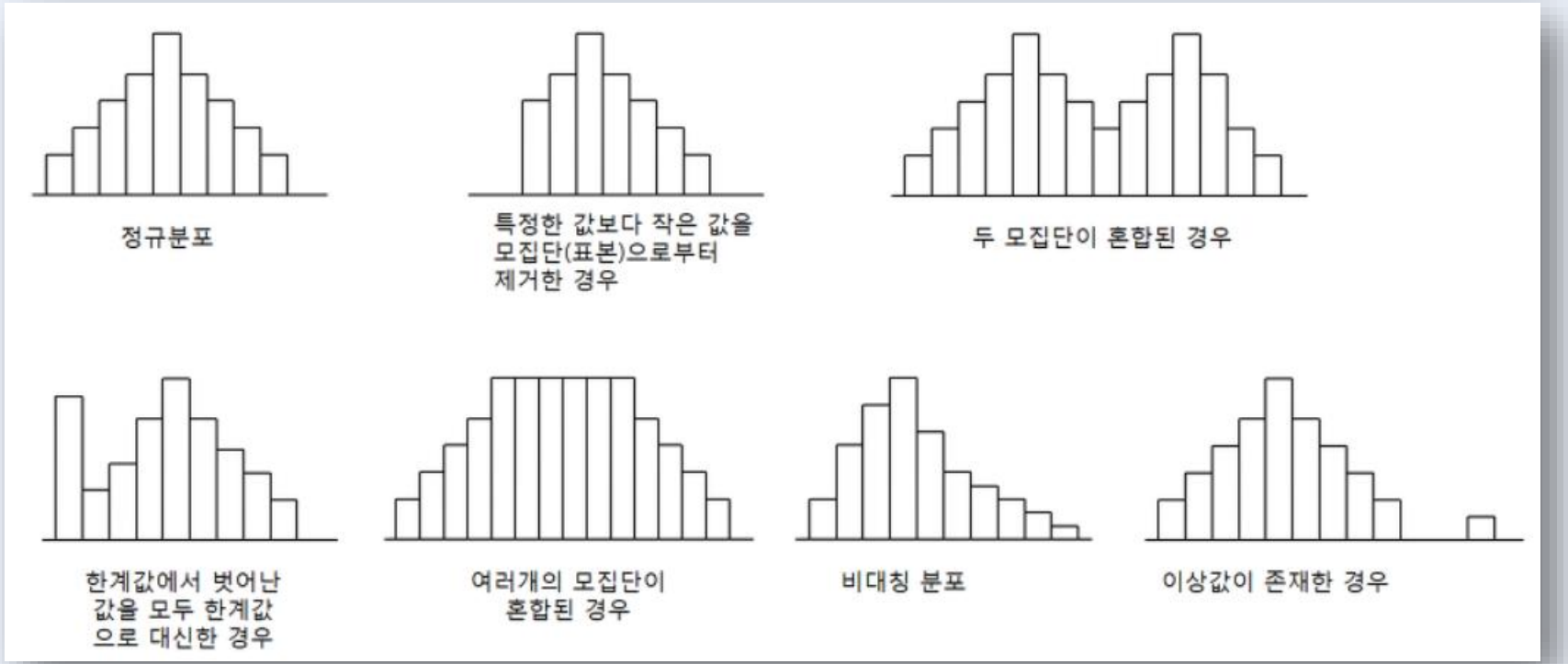
히스토그램(Historgram) 그래프

- **첨도(kurtosis)**
  - → **분포가 평균을 중심으로 얼마나 뾰족하거나 평평한지**를 나타내는 지표
  - → **극단값(이상치)의 존재 가능성이나 꼬리의 두터움을 판단**할 때 사용
  - → 첨도 == 3 : 정규분포 경우. 이론적
  - → 첨도 > 3 (양의 첨도) : 꼬리가 두껍고 중심이 뾰족함. 이상치가 많을 수 있음
  - → 첨도 < 3 (음의 첨도) : 꼬리가 얇고 중심이 평평함. 이상치가 적을 수 있음

VISUALIZE DATA BY TYPE

히스토그램(Historgram) 그래프

- 데이터 분포



VISUALIZE DATA BY TYPE

히스토그램(Historgram) 그래프

```
hist( x, bins=None, *, range=None, density=False, weights=None, cumulative=False,
      bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None,
      log=False, color=None, label=None, stacked=False, data=None, **kwargs )
```

x	데이터
bins	막대 수 / 구간 수
density	True : y축을 비율로 표시, False : y축을 도수로 표시
color	막대 색상
edgecolor	막대 테두리 색상
histtype	bar, barstacked, step, stepfilled 유형

VISUALIZE DATA BY TYPE

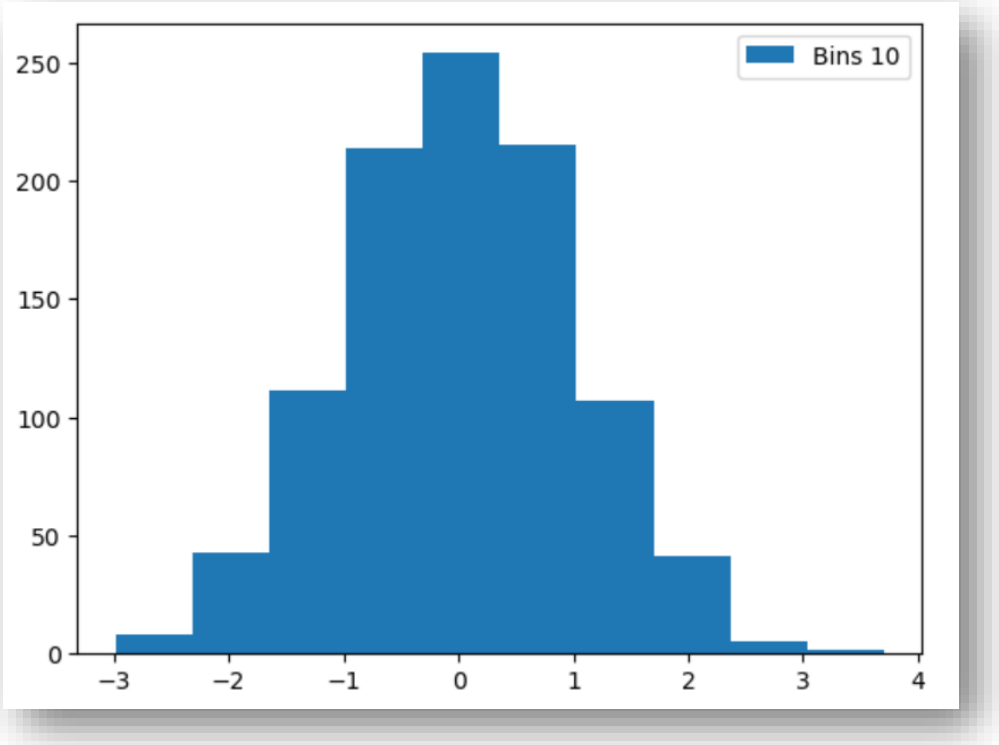
히스토그램(Historgram) 그래프

```
## 모듈 로딩
import numpy as np
import matplotlib.pyplot as plt

## 데이터 준비
np.random.seed(3)
data = np.random.randn(1000)
print(f'평균:{data.mean():.1f}, 표준편차:{data.std():.1f}')
```

VISUALIZE DATA BY TYPE

히스토그램(Historgram) 그래프

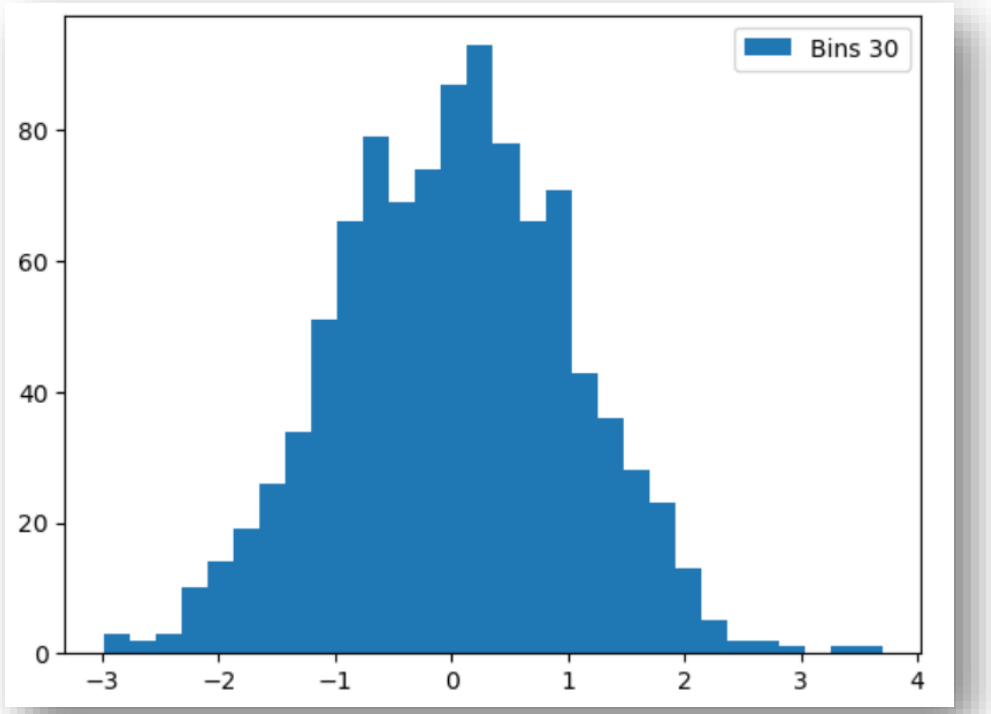
```
## 시각화
## 구간 설정 - 기본값
plt.hist(data, label='Bins 10')
plt.legend()
plt.show()
```



VISUALIZE DATA BY TYPE

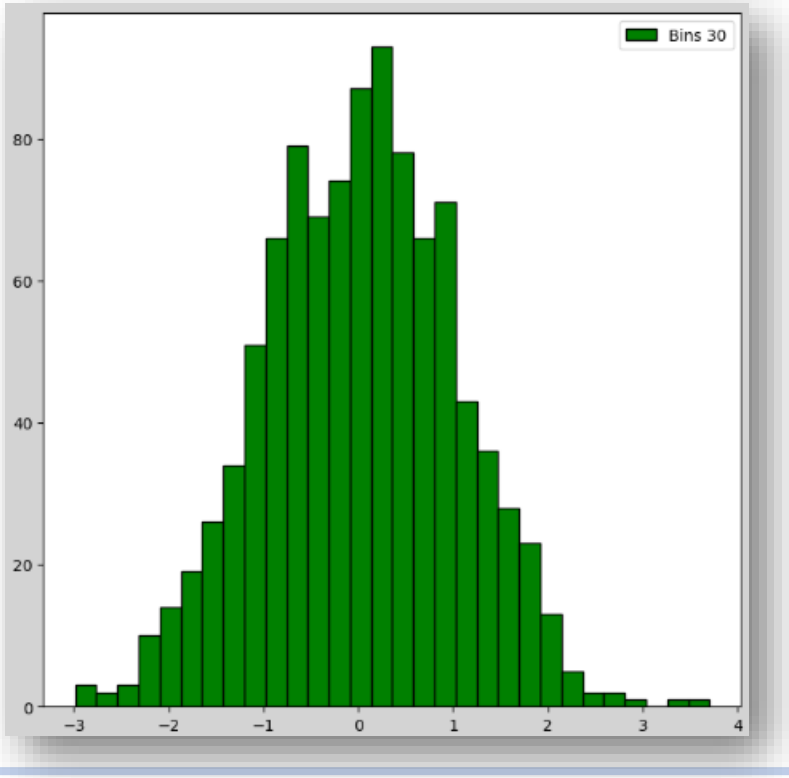
히스토그램(Historgram) 그래프

```
## 시각화
## bins 매개변수 : 막대 수 설정
plt.hist(data, bins=30, label='Bins 30')
plt.legend()
plt.show()
```



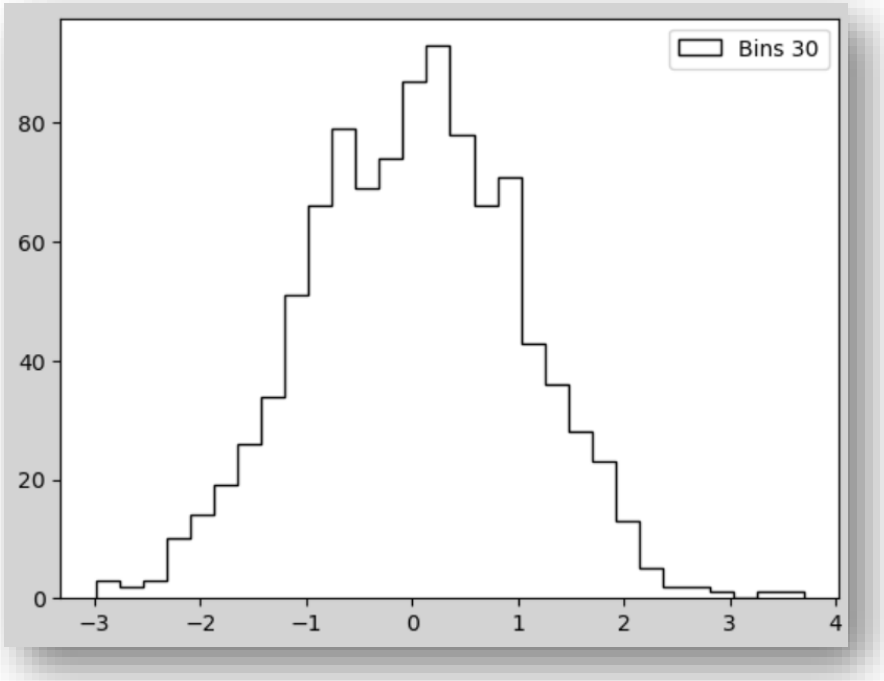
히스토그램(Historgram) 그래프

```
## 시각화
##  color / edgecolor 매개변수 : 색상설정
fig = plt.figure(figsize=(8, 8))
fig.set_facecolor('lightgray')
plt.hist(data, bins=30, label='Bins 30',
         color='green', edgecolor='black')
plt.legend()
plt.show()
```



히스토그램(Historgram) 그래프

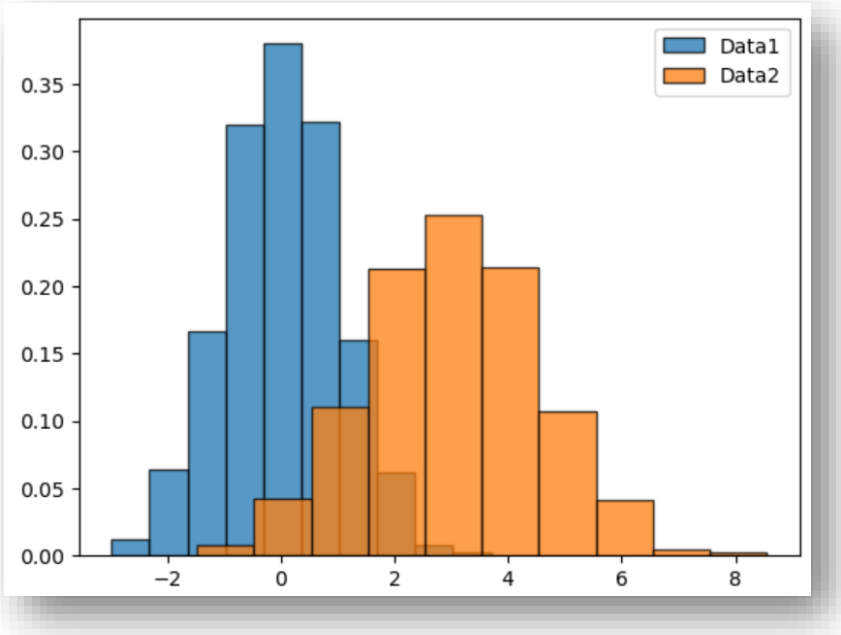
```
## 시각화
## histtype 매개변수 : 타입 설정
fig = plt.figure()
fig.set_facecolor('lightgray')

plt.hist(data, bins=30, label='Bins 30',
         color='green', edgecolor='black',
         histtype='step')

plt.legend()
plt.show()
```



VISUALIZE DATA BY TYPE

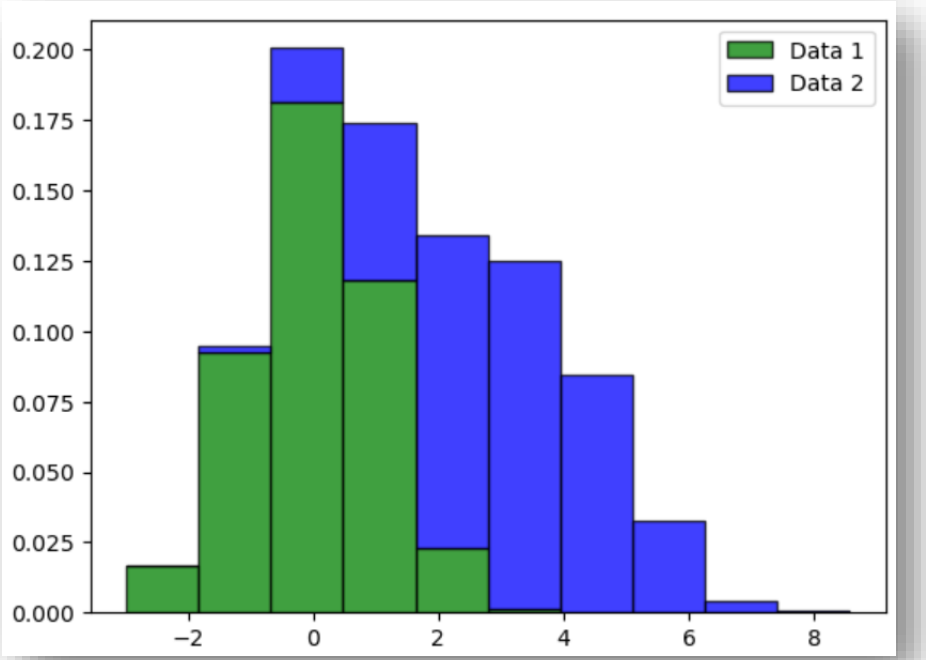
히스토그램(Historgram) 그래프

```
## 데이터 준비
np.random.seed(3)
data2 = (np.random.randn(1000) + 2) * 1.5
print(f'평균:{data2.mean():.1f}, 표준편차:{data2.std():.1f}')

## 시각화
## density 매개변수 : y축을 비율로 표현
plt.hist(data, label='Data1', density=True,
         alpha=0.75, edgecolor='k')
plt.hist(data2,label='Data2', density=True,
         alpha=0.75, edgecolor='k')
plt.legend()
plt.show()
```



히스토그램(Historgram) 그래프

```
## 시각화
## density 매개변수 : y축을 비율로 표현
plt.hist([data, data2], color=['green', 'blue'],
         histtype='barstacked',
         label=['Data 1', 'Data 2'],
         density=True, alpha=0.75, edgecolor='k')
plt.legend()
plt.show()
```



◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

- 데이터의 분포 시각화 및 이상치 체크에 사용되는 그래프
- 데이터의 사분위 수(Quartile)를 이용한 데이터 분포 시각화
- 5개 수치 : 최소값, 제 1사분위 수(Q1), 제 2사분위 수(Q2), 제 3사분위 수(Q3), 최대값

◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

▪ **사분위 수(Quartile)**

→ 데이터를 오름차순 정렬하여 4등분한 점

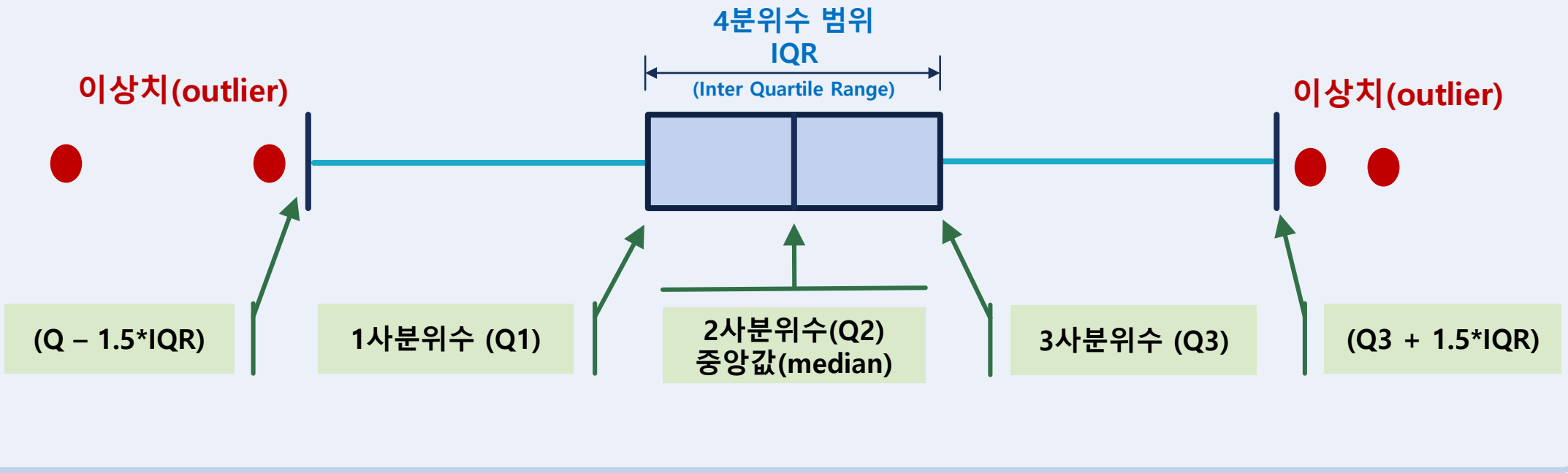
→ 3개의 사분위 수 존재

- 제 1사분위 수(Q1) : 중앙값 기준으로 하위 50% 중 중앙값, 전체 데이터 중 하위 25% 해당 하는 값
- 제 2사분위 수(Q2) : 전체 데이터의 중앙값
- 제 3사분위 수(Q3) : 중앙값 기준으로 상위 50% 중 중앙값, 전체 데이터 중 상위 25% 해당 하는 값

→ 사분위 범위(Inter Quartile Range: IQR) : 제 3사분위 수 – 제 1사분위 수

◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

▪ 사분위 수(Quartile)



◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

- **신뢰구간(Confidence Interval CI)**

  구간 추정(Interval Estimation)에서 모수가 포함될 것이라고 기대되는 구간 제시 후 추청 방식

  제시된 구간을 신뢰구간

- **신뢰수준(Confidence Interval CI)**

  모수를 갖는 모집단으로부터 같은 크기의 랜덤표본을 여러 개 추출하였을 때,

  각각 표본으로부터 얻은 신뢰구간 중 모수를 포함한 정도

  예) 신뢰구간에서 신뢰수준 95%

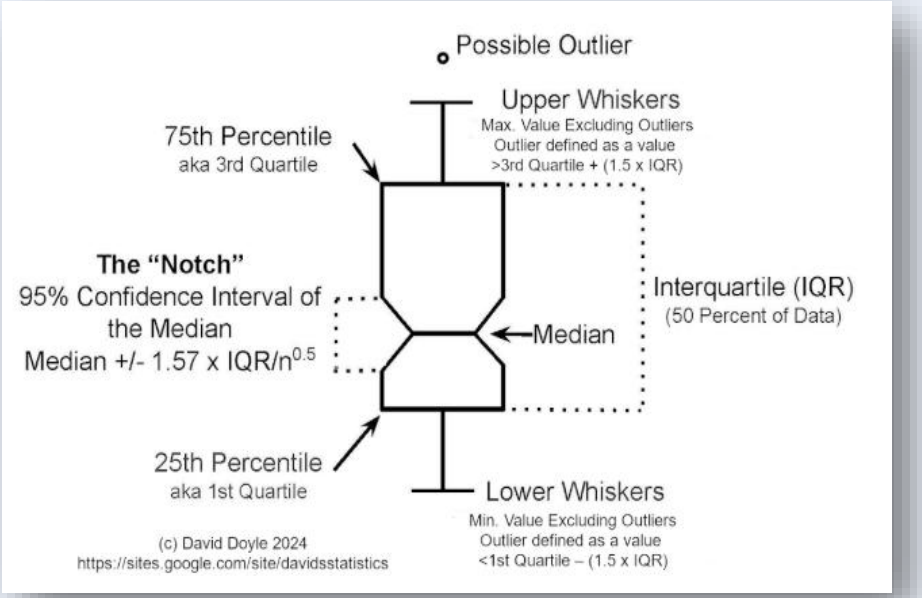
VISUALIZE DATA BY TYPE

◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

- **Norch**

중앙값의 95% 신뢰구간 표현

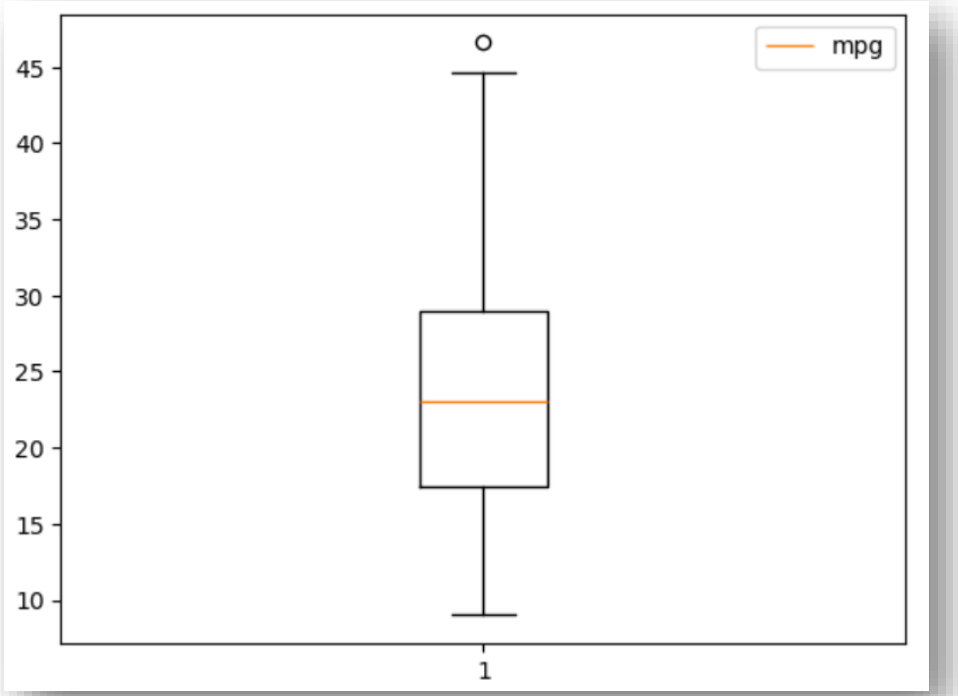
해당 구간안에 중앙값이 100개 중 95개 존재 함 의미



VISUALIZE DATA BY TYPE

상자수염(Box Whisker Plot OR Box Plot) 그래프

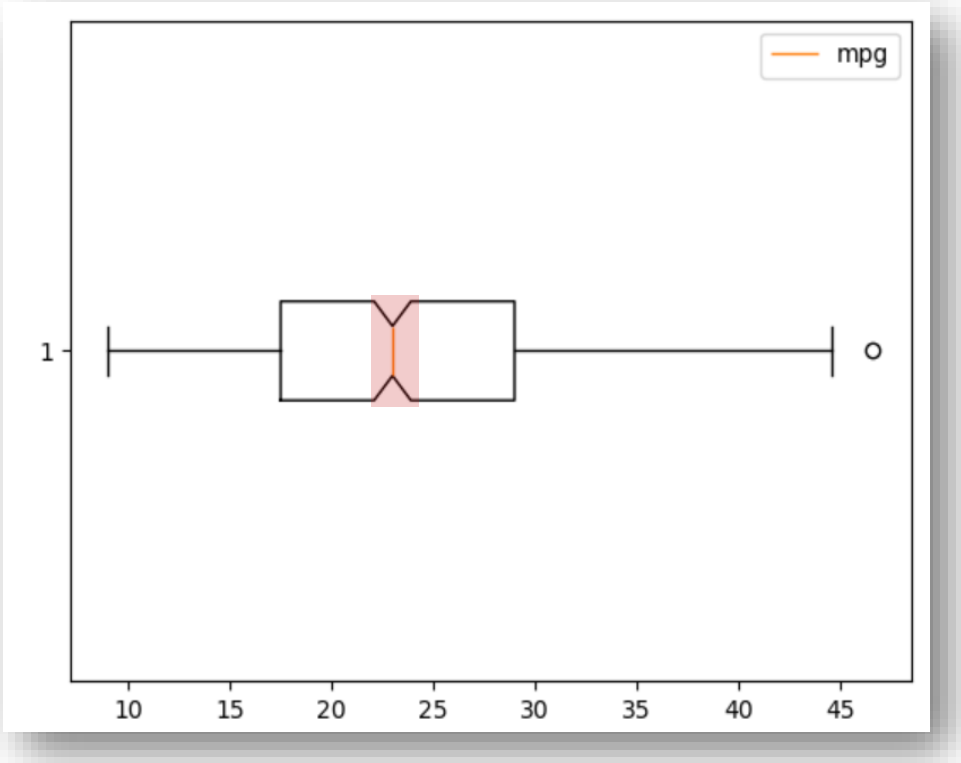
```
## 데이터 준비
D_FILE = '../DATA/auto_mpg.csv'

autoDF=pd.read_csv(D_FILE)
autoDF.head()

##  시각화
plt.boxplot(autoDF['mpg'], label='mpg')
plt.legend()
plt.show()
```



VISUALIZE DATA BY TYPE

◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

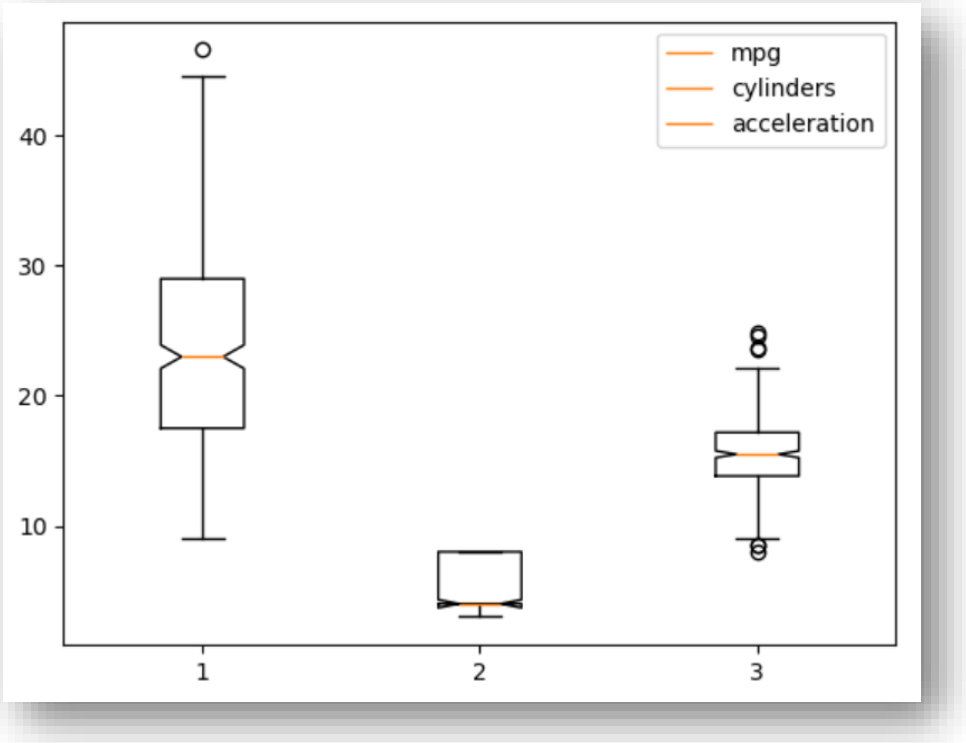
```
##  시각화
##  notch 매개변수 :  중앙값의 95% 신뢰구간 표현
##  vert   매개변수 :  수직/세로 표현 여부
plt.boxplot(autoDF['mpg'], label='mpg',
                notch=True, vert=False)
plt.legend()
plt.show()
```



VISUALIZE DATA BY TYPE

상자수염(Box Whisker Plot OR Box Plot) 그래프

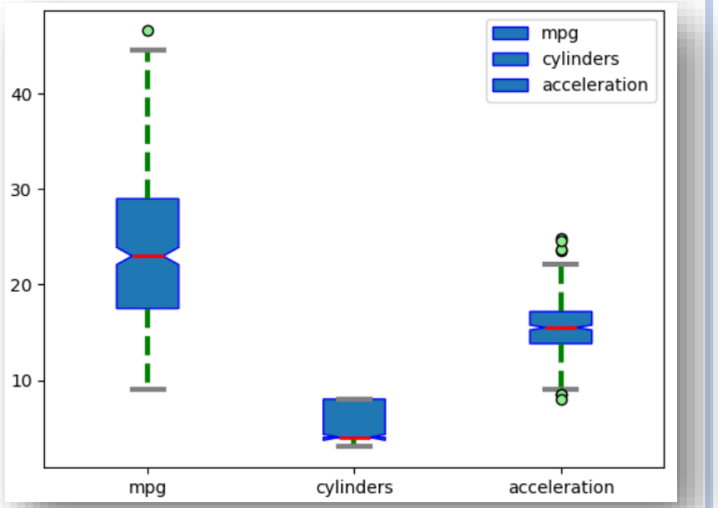
```
##  시각화
data_name=['mpg','cylinders', 'acceleration']

plt.boxplot(    autoDF[data_name],
                label=data_name,
                notch=True)
plt.legend()
plt.show()
```



VISUALIZE DATA BY TYPE

◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

```
##  시각화
plt.boxplot(autoDF[data_name],
            label=data_name,
            notch=True,
            ## 박스 안쪽 여백 색상 설정
            patch_artist=True,
             ## 이상치 설정
            flierprops=dict(markerfacecolor='lightgreen'),
            ## Q1과 최소값, Q3와 최대값 잇는 선 설정
            whiskerprops=dict(color='green', linewidth=3, linestyle='--'),
            ## 최상단, 최하단 선과 직사각형을 잇는 선 테두리 컬러
            capprops=dict(color='gray', linewidth=3),
            ## 중앙값 설정
            medianprops=dict(color='red',linewidth=2),
            ## IQR 설정
            boxprops=dict(color='blue')
)
```



◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

```
##  plot 객체 활용 정보 추출
print(f'boxplot 객체 포함 정보₩n{boxplot.keys()}')
for key in boxplot.keys():
    print(f'[{key}]====₩n{boxplot[key]}₩n')
```

```
boxplot 객체 포함 정보
dict_keys(['whiskers', 'caps', 'boxes', 'medians', 'fliers', 'means'])
[whiskers]====
[<matplotlib.lines.Line2D object at 0x00000276AC6D9040>, <matplotlib.line
0x00000276AC6D9E20>, <matplotlib.lines.Line2D object at 0x00000276AC6C38E
object at 0x00000276AC6E5040>, <matplotlib.lines.Line2D object at 0x00000
<matplotlib.lines.Line2D object at 0x00000276AC65D7F0>]

[caps]====
[<matplotlib.lines.Line2D object at 0x00000276AC6D9E50>, <matplotlib.line
0x00000276AC6C3FD0>, <matplotlib.lines.Line2D object at 0x00000276AC6E522
object at 0x00000276AC6E53D0>, <matplotlib.lines.Line2D object at 0x00000
<matplotlib.lines.Line2D object at 0x00000276AC6D1910>]
```

◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

```
# whiskers => Q1, Q3, max, min 값
print(f'boxplot["whiskers"] : {len(boxplot["whiskers"])}개')

for w in boxplot["whiskers"]:
    print(w.get_ydata())
```

```
boxplot["whiskers"] : 6개
[17.5  9. ]
[29.  44.6]
[4. 3.]
[8. 8.]
[13.825  9.   ]
[17.175 22.2  ]
boxplot["medians"] : 3개
[23. 23.]
[4. 4.]
[15.5 15.5]
```

◆ 상자수염(Box Whisker Plot OR Box Plot) 그래프

```
# 이상치에 대한 정보 추출
print(f'boxplot["fliers"] : {len(boxplot["fliers"])}개')


for m in boxplot["fliers"]:
    print(m.get_ydata())
```

```
boxplot["fliers"] : 3개
[46.6]
[]
[ 8.5  8.5  8.  23.5 24.8 23.7 24.6]
```