

# 데이터 시각화 (Matplotlib)

inky4832@daum.net

# 1장. 시각화 개요

**Comparison and Ranking**

**Part to whole**

**Trend**

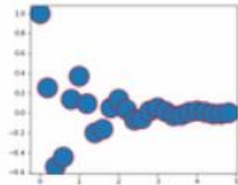
**Correlation**

**Distribution**

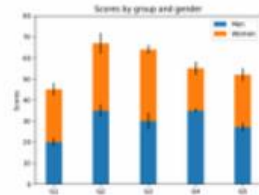
## 2. Chart 종류

python

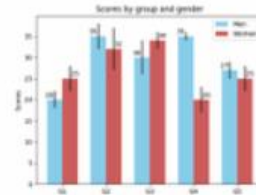
<https://matplotlib.org/stable/gallery/index.html>



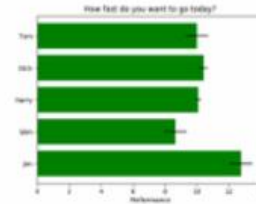
Arctest



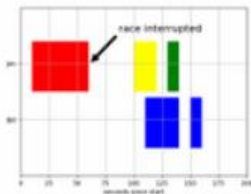
Stacked Bar Graph



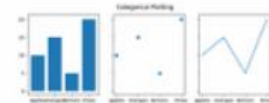
Barchart



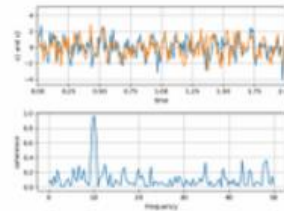
Horizontal bar chart



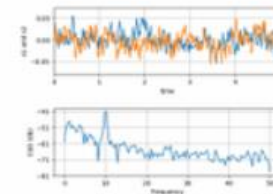
Broken Barh



Plotting categorical variables



Plotting the coherence of two signals

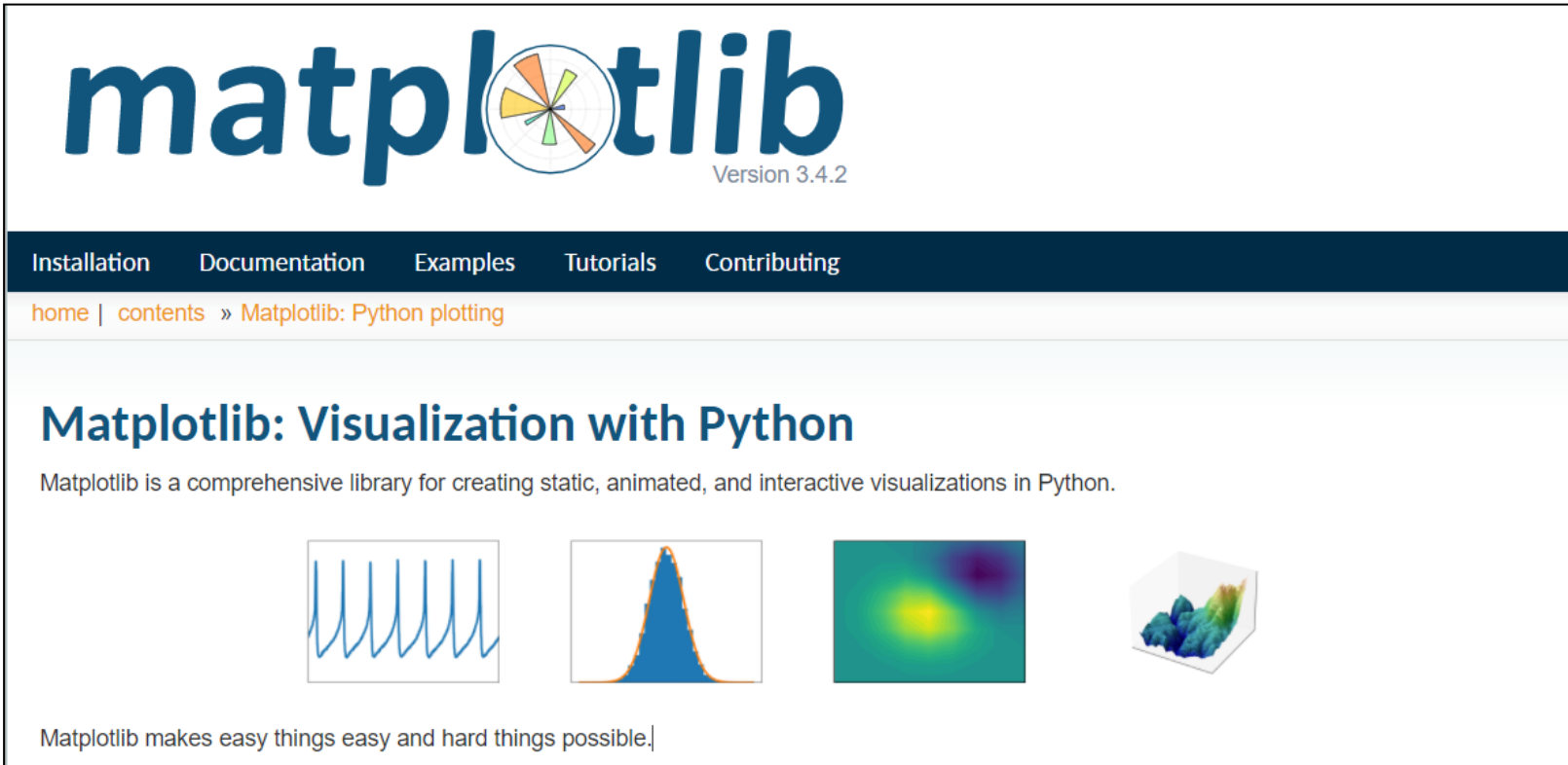


CSD Demo

### 3. matplotlib 라이브러리

*python*

<https://matplotlib.org/>



The screenshot shows the Matplotlib website homepage. At the top is the Matplotlib logo, which consists of the word "matplotlib" in a blue sans-serif font, with a circular icon containing several colored segments (yellow, orange, green, blue) in the middle of the word. To the right of the logo is the text "Version 3.4.2". Below the logo is a dark blue navigation bar with white text links: "Installation", "Documentation", "Examples", "Tutorials", and "Contributing". Below the navigation bar is a light orange breadcrumb trail: "home | contents » Matplotlib: Python plotting". The main heading is "Matplotlib: Visualization with Python" in a large, bold, blue font. Below the heading is a paragraph: "Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python." Below the text are four small square images: a line plot with multiple sharp peaks, a 2D histogram or density plot showing a bell-shaped distribution, a 2D heatmap with a yellow and green gradient, and a 3D surface plot with a blue and green gradient. At the bottom of the page is the tagline: "Matplotlib makes easy things easy and hard things possible."


**matplotlib** Version 3.4.2

Installation Documentation Examples Tutorials Contributing

home | contents » Matplotlib: Python plotting

## Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.



Matplotlib makes easy things easy and hard things possible.

<https://matplotlib.org/stable/users/installing/index.html#installing-an-official-release>

### Installation

#### Installing an official release

Matplotlib releases are available as wheel packages for macOS, Windows

```
python -m pip install -U pip  
python -m pip install -U matplotlib
```

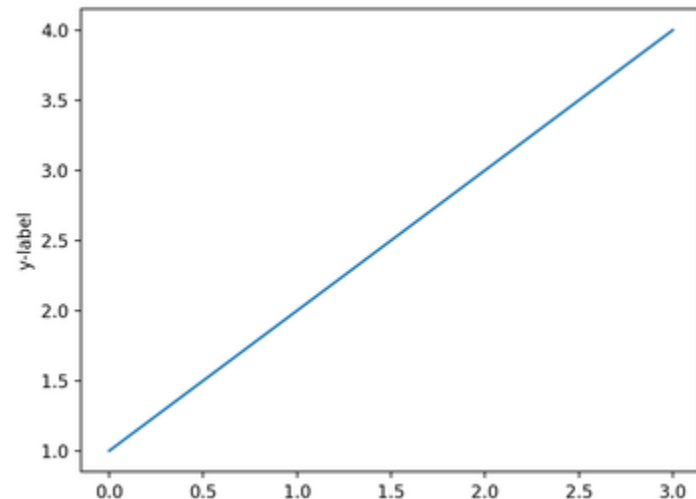
## 5. matplotlib.pyplot 모듈

*python*

<https://matplotlib.org/stable/tutorials/introductory/pyplot.html#intro-to-pyplot>

matplotlib.pyplot 모듈은 내장된 함수를 사용해서 그래프를 만들고 다양한 시각화가 가능하다.

```
import matplotlib.pyplot as plt  
plt.plot([1, 2, 3, 4])  
plt.ylabel('some numbers')  
plt.show()
```



## 6. figure 및 axes (æksiz) 개요

python

<https://matplotlib.org/stable/tutorials/introductory/usage.html#parts-of-a-figure>

```
In [1]: import matplotlib.pyplot as plt
```

```
In [2]: # 시각화하기 위한 도화지 생성
fig = plt.figure(figsize=(8,6)) # figsize=( w, h ) 단위는 inch

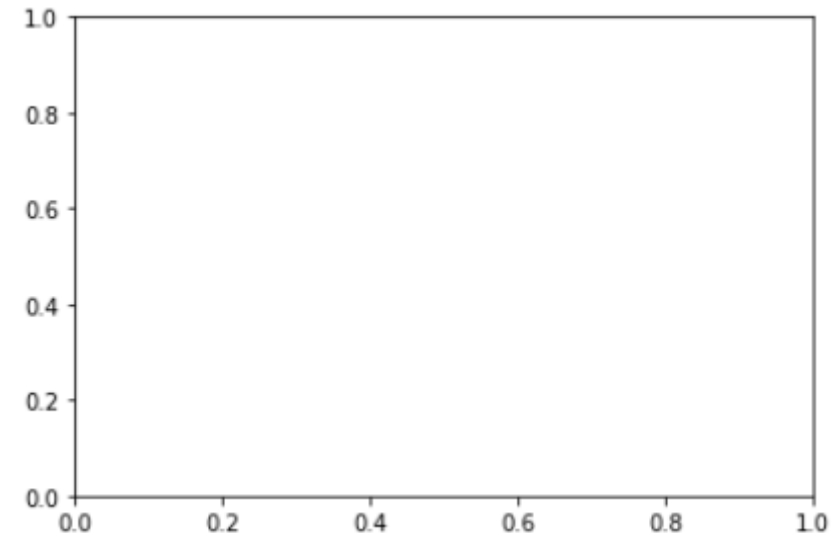
<Figure size 576x432 with 0 Axes>
```

```
In [3]: # 도화지에 그리기 위한 붓 생성
ax = plt.axes()
plt.show()
```



figure와 axes를 한꺼번에 생성

```
f, ax = plt.subplots()
```



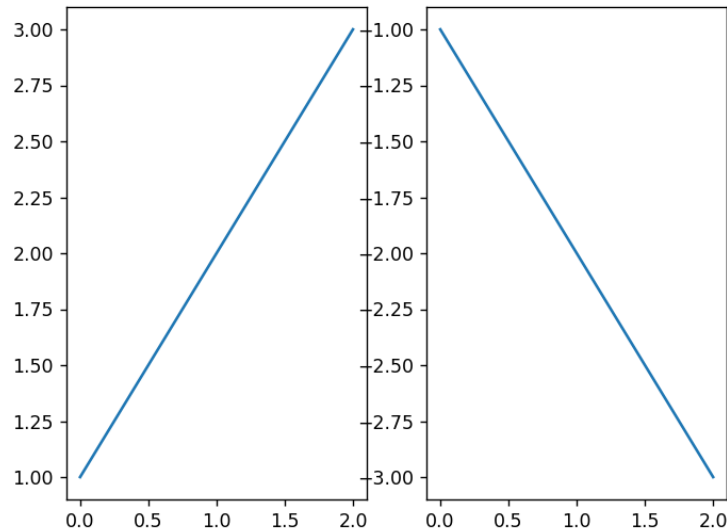


## 6. figure 및 axes (æksiz) 개요

python

- 하나의 figure안에 여러 axes 생성 가능
- 생성된 axes는 배열로 반환

```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(nrows=1, ncols=2) # 1행 2열
print(axes[0])
axes[0].plot([1,2,3])
axes[1].plot([-1,-2,-3])
plt.show()
```

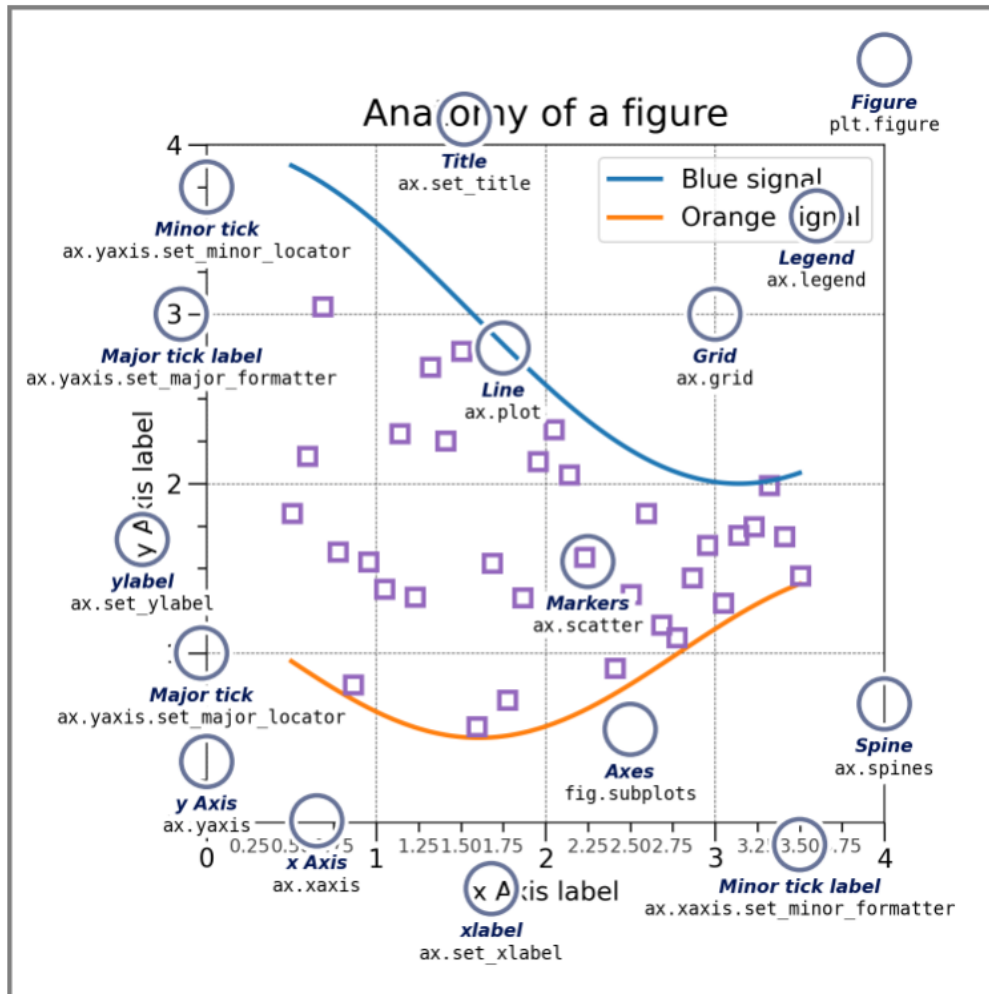


# 7. figure 구성

python

## Parts of a Figure #

Here are the components of a Matplotlib Figure.



## 2장. 선그래프

[https://matplotlib.org/2.0.2/api/pyplot\\_api.html?highlight=plot#matplotlib.pyplot.plot](https://matplotlib.org/2.0.2/api/pyplot_api.html?highlight=plot#matplotlib.pyplot.plot)

```
matplotlib.pyplot.plot(*args, **kwargs)
```

Plot lines and/or markers to the **Axes**. *args* is a variable length argument, allowing for multiple *x*, *y* pairs with an optional format string. For example, each of the following is legal:

```
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
```

If *x* and/or *y* is 2-dimensional, then the corresponding columns will be plotted.

If used with labeled data, make sure that the color spec is not included as an element in data, as otherwise the last case `plot("v", "r", data={"v":..., "r":...})` can be interpreted as the first case which would do `plot(v, r)` using the default line style and color.

If not used with labeled data (i.e., without a data argument), an arbitrary number of *x*, *y*, *fmt* groups can be specified, as in:

```
a.plot(x1, y1, 'g^', x2, y2, 'g-')
```

Return value is a list of lines that were added.

By default, each line is assigned a different style specified by a 'style cycle'. To change this behavior, you can edit the `axes.prop_cycle` rcParam.

# 1. 선그래프 - y값 입력

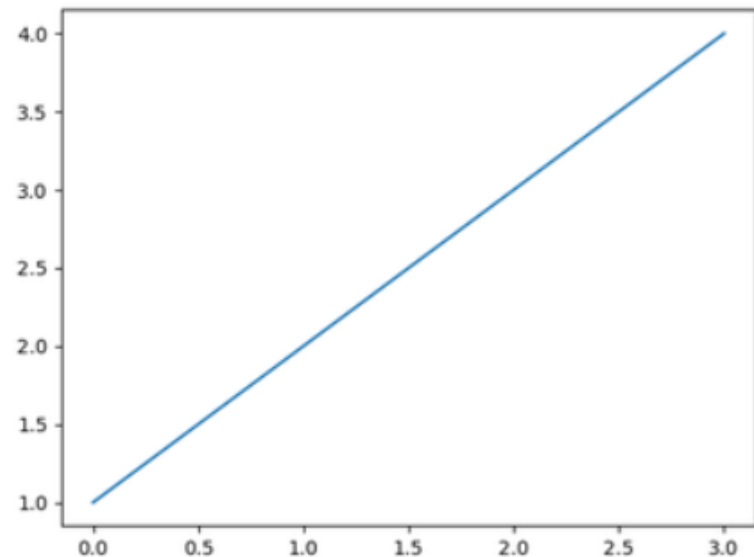
python

plot() 함수는 선(Line) 또는 마커(marker) 그래프를 그릴 때 사용되는 함수이다.

## 예제

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4])
plt.show()
```



y 값 입력하기.

[1,2,3,4]와 같이 하나의 리스트로 값들을 입력하면 y값으로 인식한다.  
X값은 자동으로 [0, 1, 2, 3]이 생성되어 y값과 대응된다.  
=> (0, 1),(1, 2),(2, 3),(3, 4)를 잇는 그래프가 그려진다.

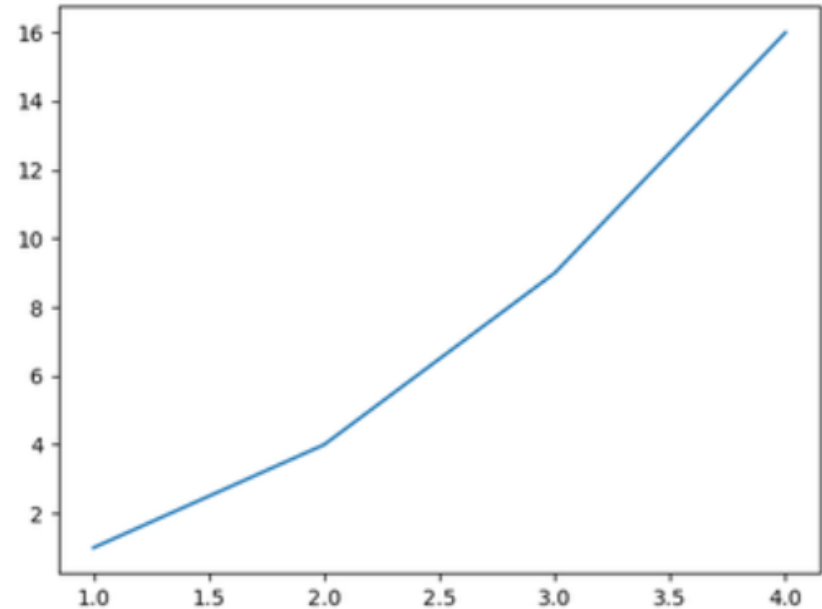
## 2. 선그래프 - x, y 값 입력

python

### 예제

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```



x, y 값 입력하기.

두 개의 리스트를 입력하면 순서대로 x, y값들로 인식한다.  
따라서 (1, 1), (2, 4), (3, 9), (4, 16) 를 잇는 그래프가 그려진다.

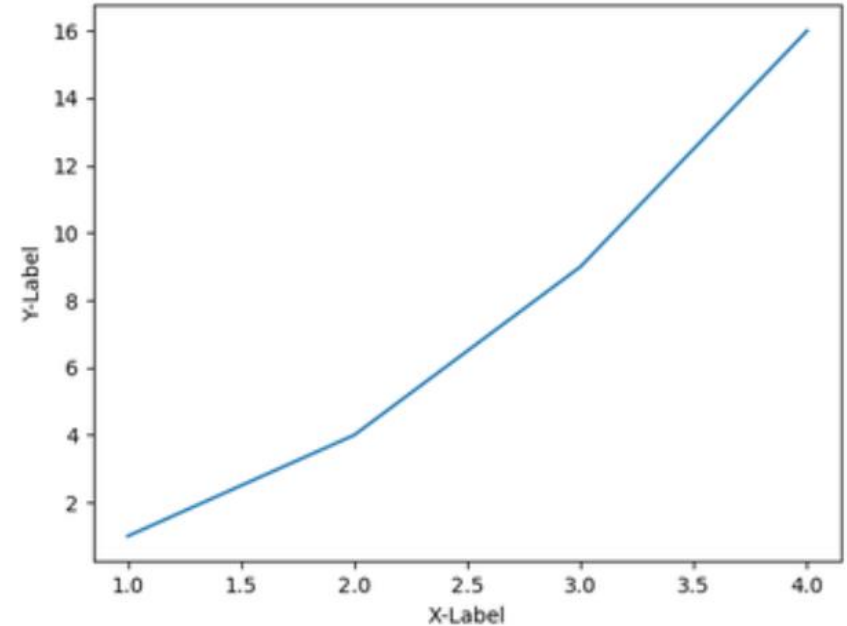
### 3. 선그래프 - 축 레이블 설정

python

#### 예제

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.xlabel('X-Label')
plt.ylabel('Y-Label')
plt.show()
```



x, y 축 레이블 설정하기.

matplotlib.pyplot 모듈의 xlabel(), ylabel() 함수를 사용하면 그래프의 x, y축에 대한 레이블을 표시할 수 있다.

## 4. 선그래프 - 축 범위 지정

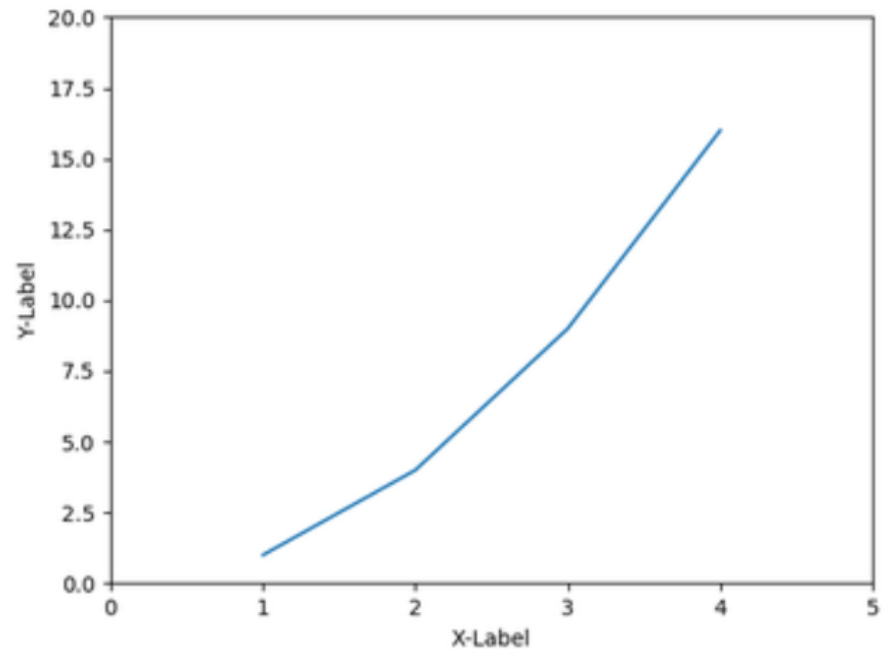
python

[https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.axes.html#matplotlib.pyplot.axes](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.axes.html#matplotlib.pyplot.axes)

### 예제

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.xlabel('X-Label')
plt.ylabel('Y-Label')
plt.axis([0, 5, 0, 20])
plt.show()
```



축 범위 지정하기.

axis() 함수에 [xmin, xmax, ymin, ymax] 형태로 x, y축의 범위를 지정한다.  
입력 리스트는 반드시 네 개의 값이 있어야 된다.  
만약 입력값이 없으면 데이터에 맞게 자동으로 범위를 지정한다.



## 5. 선그래프 - 눈금표시

python

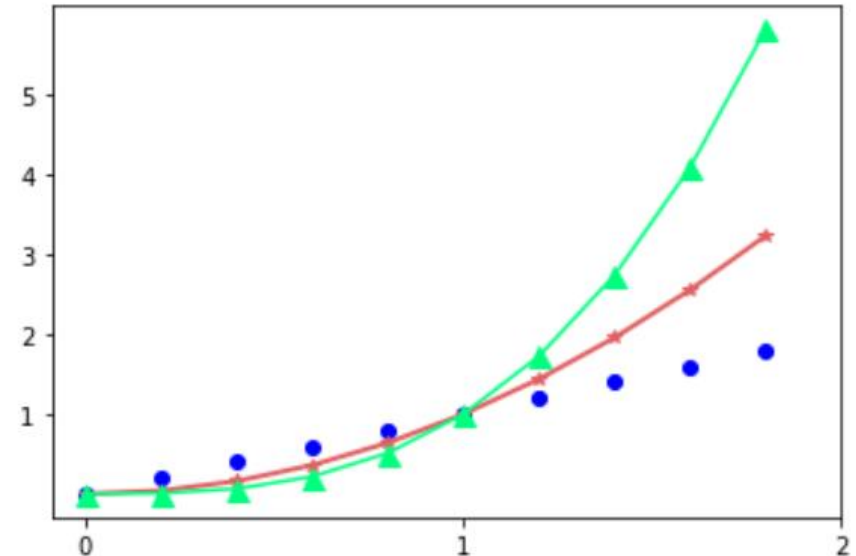
### 예제

```
import matplotlib.pyplot as plt
import numpy as np

a = np.arange(0, 2, 0.2)

plt.plot(a, a, 'bo')
plt.plot(a, a**2, color='#e35f62', marker='*', linewidth=2)
plt.plot(a, a**3, color='springgreen', marker='^', markersize=9)
plt.xticks([0, 1, 2])
plt.yticks(np.arange(1, 6))

plt.show()
```



틱(Tick)은 그래프의 축에 간격을 구분하기 위해 표시하는 눈금을 의미한다. `xticks()`, `yticks()` 함수를 사용하며 라벨(label)지정도 가능하다.

## 5. 선그래프 - 색상 지정

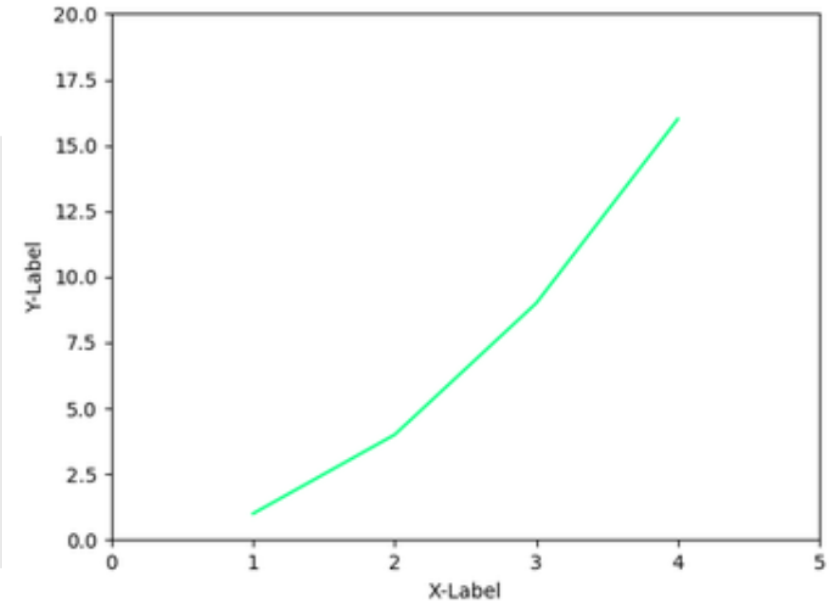
python

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

### 예제

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16], color='springgreen')
plt.xlabel('X-Label')
plt.ylabel('Y-Label')
plt.axis([0, 5, 0, 20])
plt.show()
```



색상 지정하기.

```
plt.plot([1, 4, 9, 16], 'r')
```

```
plt.plot([1, 4, 9, 16], 'b')
```

```
plt.plot([1, 4, 9, 16], 'g')
```

```
plt.plot([1, 4, 9, 16], color='springgreen')
```

```
plt.plot([1, 4, 9, 16], color='violet')
```

```
plt.plot([1, 4, 9, 16], color='dodgerblue')
```

## 5. 선그래프 - 색상 지정

python

### 색상 종류

#### 기본 색상



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

#### CSS 색상



## 5. 선그래프 - 색상 지정

python

white	floralwhite	turquoise	darkslateblue
snow	darkgoldenrod	lightseagreen	mediumslateblue
rosybrown	goldenrod	mediumturquoise	mediumpurple
lightcoral	cornsilk	azure	rebeccapurple
indianred	gold	lightcyan	blueviolet
brown	lemonchiffon	paleturquoise	indigo
firebrick	khaki	darkslategray	darkorchid
maroon	palegoldenrod	darkslategrey	darkviolet
darkred	darkkhaki	teal	mediumorchid
red	ivory	darkcyan	thistle
mistyrose	beige	aqua	plum
salmon	lightyellow	cyan	violet
tomato	lightgoldenrodyellow	darkturquoise	purple
darksalmon	olive	cadetblue	darkmagenta
coral	yellow	powderblue	fuchsia
orangered	olivedrab	lightblue	magenta
lightsalmon	yellowgreen	deepskyblue	orchid
sienna	darkolivegreen	skyblue	mediumvioletred
seashell	greenyellow	lightskyblue	deeppink
chocolate	chartreuse	steelblue	hotpink
saddlebrown	lawngreen	aliceblue	lavenderblush
sandybrown	honeydew	dodgerblue	palevioletred
peachpuff	darkseagreen	lightslategray	crimson
peru	palegreen	lightslategrey	pink
linen	lightgreen	slategray	lightpink

## 5. 선그래프 - 색상 지정

python

 orangered	 olivedrab	 lightblue	 magenta
 lightsalmon	 yellowgreen	 deepskyblue	 orchid
 sienna	 darkolivegreen	 skyblue	 mediumvioletred
 seashell	 greenyellow	 lightskyblue	 deeppink
 chocolate	 chartreuse	 steelblue	 hotpink
 saddlebrown	 lawngreen	 aliceblue	 lavenderblush
 sandybrown	 honeydew	 dodgerblue	 palevioletred
 peachpuff	 darkseagreen	 lightslategray	 crimson
 peru	 palegreen	 lightslategray	 pink
 linen	 lightgreen	 slategray	 lightpink

### 16진수 색상

93DAFF	98DFFF	9DE4FF	A2E9FF	A7EEFF
ACF3FF	B0F7FF	B4FBFF	B9FFFF	C0FFFF
87CEFA	91D8FA	A5D8FA	AFDDFA	B9E2FA
C3E7FA	CDECFA	D7F1FA	E1F6FA	EBFBFF
00BFFF	0AC9FF	14D3FF	1EDDFF	28E7FF

## 5. 선그래프 - 마커 지정

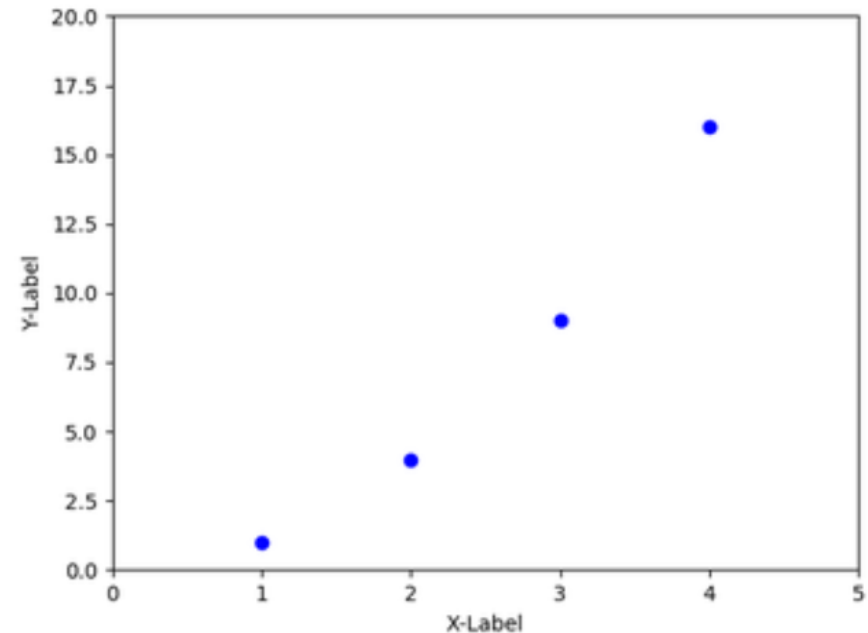
python

[https://matplotlib.org/stable/api/markers\\_api.html?highlight=marker#module-matplotlib.markers](https://matplotlib.org/stable/api/markers_api.html?highlight=marker#module-matplotlib.markers)

### 예제

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'bo')
plt.xlabel('X-Label')
plt.ylabel('Y-Label')
plt.axis([0, 5, 0, 20])
plt.show()
```



마커 지정하기.

plot() 함수에 'bo'를 입력하면 파란색의 원형 마커로 그래프가 그려진다.  
'b' 는 blue, 'o' circle를 나타내는 문자이다.

## 5. 선그래프 - 마커 지정

python

### matplotlib.markers

Functions to handle markers; used by the marker functionality of `plot`, `scatter`, and `errorbar`.

All possible markers are defined here:

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)

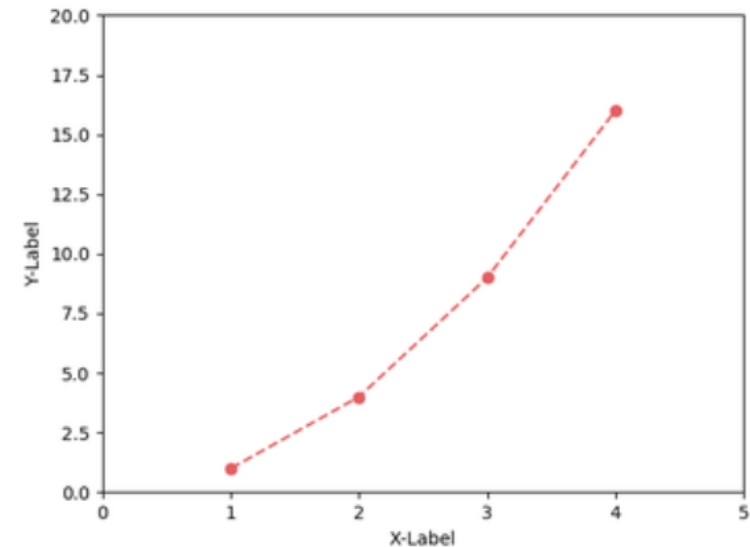
## 5. 선그래프 - 선 스타일

python

[https://matplotlib.org/stable/api/markers\\_api.html?highlight=marker#module-matplotlib.markers](https://matplotlib.org/stable/api/markers_api.html?highlight=marker#module-matplotlib.markers)

### 선 종류

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



### 예제

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16], color='#e35f62', marker='o', linestyle='--')
plt.xlabel('X-Label')
plt.ylabel('Y-Label')
plt.axis([0, 5, 0, 20])
plt.show()
```



## 5. 선그래프 - 영역 채우기

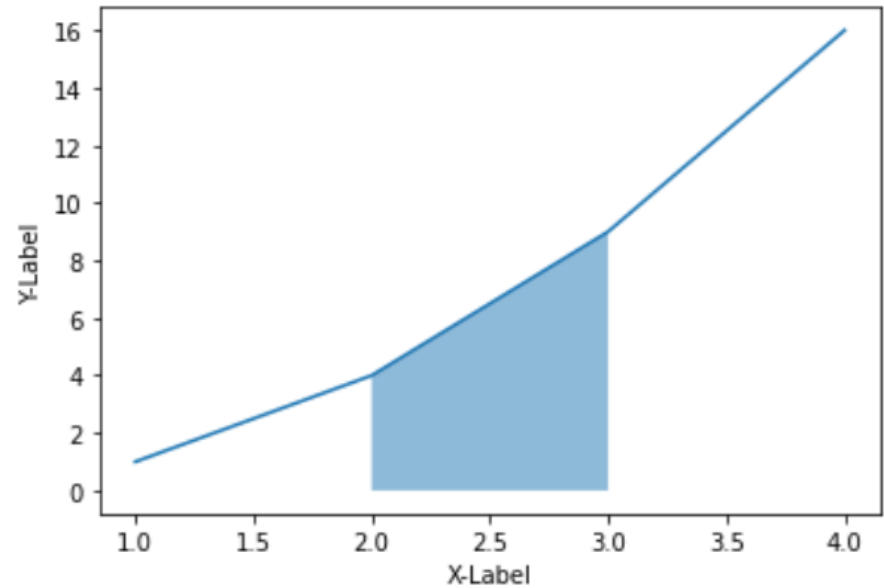
python

### 예제

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [1, 4, 9, 16]

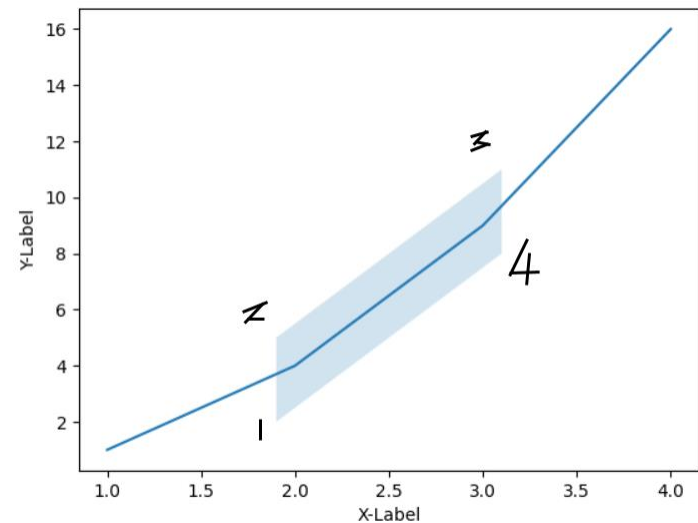
plt.plot(x, y)
plt.xlabel('X-Label')
plt.ylabel('Y-Label')
plt.fill_between(x[1:3], y[1:3], alpha=0.5)
```



### 임의의 영역

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [1, 4, 9, 16]
plt.plot(x, y)
plt.xlabel("X-Label")
plt.ylabel("Y-Label")
plt.fill([1.9, 1.9, 3.1, 3.1], [2, 5, 11, 8], alpha=0.2)
plt.show()
```



## 5. 선그래프 - 영역 채우기

python

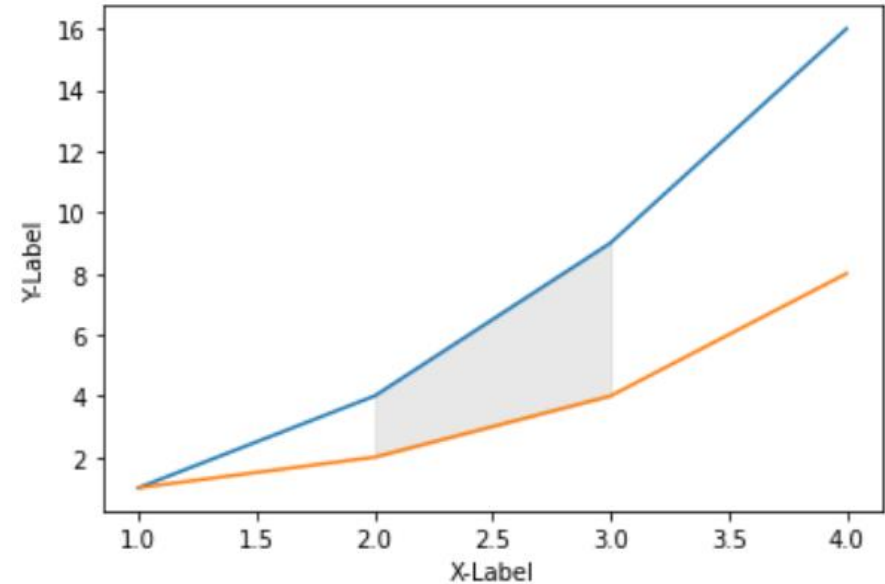
### 예제

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y1 = [1, 4, 9, 16]
y2 = [1, 2, 4, 8]

plt.plot(x, y1)
plt.plot(x, y2)
plt.xlabel('X-Label')
plt.ylabel('Y-Label')
plt.fill_between(x[1:3], y1[1:3], y2[1:3], color='lightgray', alpha=0.5)

plt.show()
```



## 5. 선그래프 - 여러 곡선 그리기-1

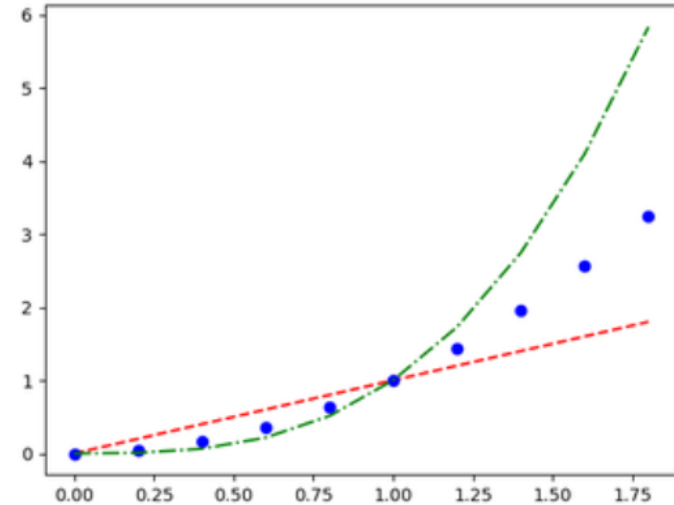
python

### 예제

```
import matplotlib.pyplot as plt
import numpy as np

a = np.arange(0, 2, 0.2)

plt.plot(a, a, 'r--', a, a**2, 'bo', a, a**3, 'g-.')
plt.show()
```



여러 곡선 그리기.

하나의 plot() 함수에 x값, y값, 스타일을 순서대로 여러 번 지정하면 하나의 그래프에 여러 개의 곡선을 그릴 수 있다.

## 5. 선그래프 - 여러 곡선 그리기-2

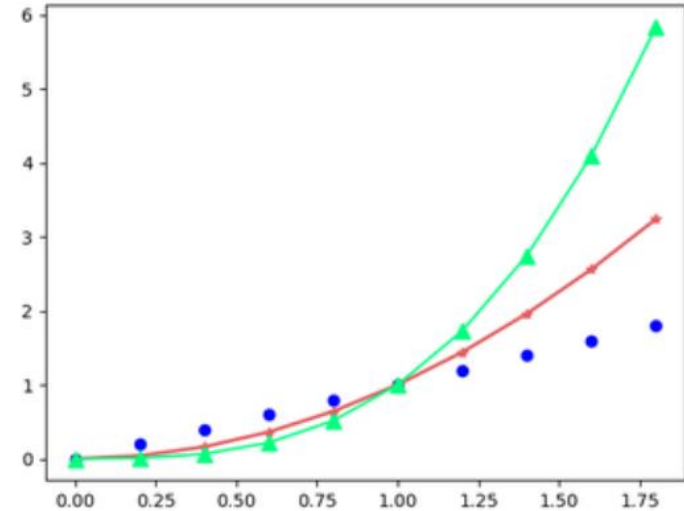
python

### 예제

```
import matplotlib.pyplot as plt
import numpy as np

a = np.arange(0, 2, 0.2)

plt.plot(a, a, 'bo')
plt.plot(a, a**2, color='#e35f62', marker='*', linewidth=2)
plt.plot(a, a**3, color='springgreen', marker='^', markersize=9)
plt.show()
```



여러 개의 plot() 함수에 x값, y값, 스타일을 각각 지정해도 하나의 그래프에 여러 개의 곡선을 그릴 수 있다.

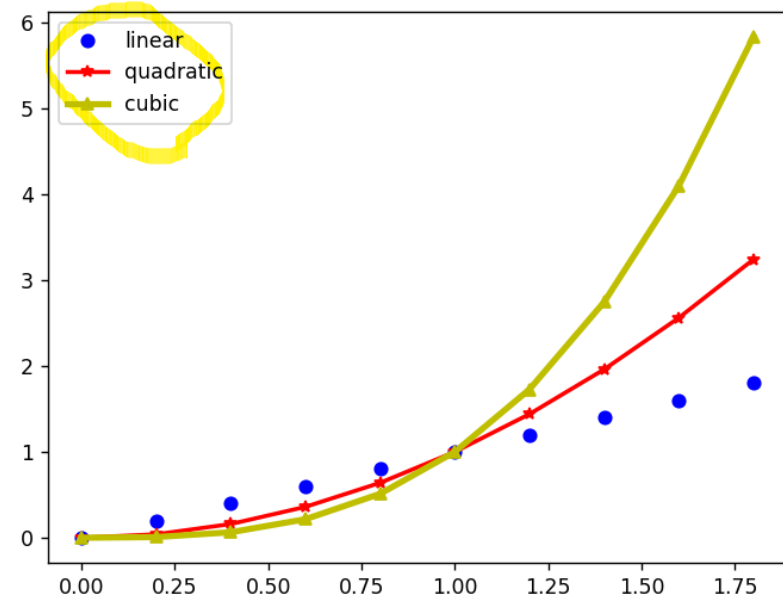
## 5. 선그래프 - 범례 ( legend )

python

### 범례 (legend)

```
plt.plot(n,n, 'bo', label='linear')
plt.plot(n,n**2, 'r', marker="*", linewidth=2, label='quadratic')
plt.plot(n,n**3, 'y', marker="^", linewidth=3, label='cubic')
plt.legend()
plt.show()
```

```
plt.plot(n,n, 'bo')
plt.plot(n,n**2, 'r', marker="*", linewidth=2)
plt.plot(n,n**3, 'y', marker="^", linewidth=3)
plt.legend(["linear", "quadratic", "cubic"])
plt.show()
```



## 5. 선그래프 - 그리드(grid)

python

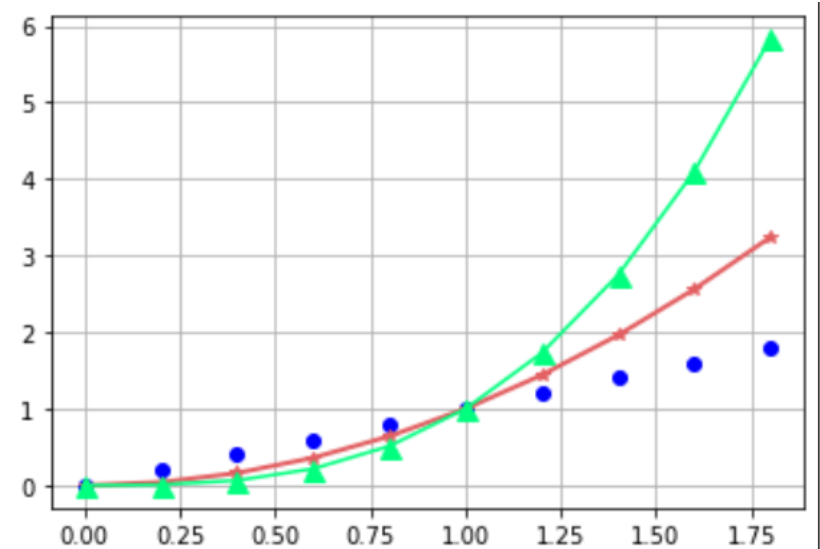
### 예제

```
import matplotlib.pyplot as plt
import numpy as np

a = np.arange(0, 2, 0.2)

plt.plot(a, a, 'bo')
plt.plot(a, a**2, color='#e35f62', marker='*', linewidth=2)
plt.plot(a, a**3, color='springgreen', marker='^', markersize=9)
plt.grid(True)

plt.show()
```



plt.grid(True) 값을 지정하면, 그래프의 x축과 y축에 그리드가 표시된다.  
axis="both|x|y" 속성을 이용하여 축 지정도 가능하다.

## 5. 선그래프 - 타이틀 지정

python

### 예제

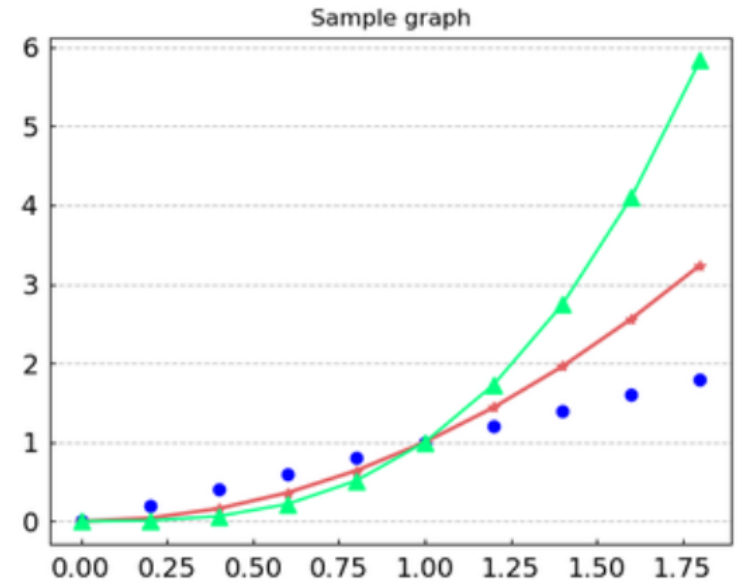
```
import matplotlib.pyplot as plt
import numpy as np

a = np.arange(0, 2, 0.2)

plt.plot(a, a, 'bo')
plt.plot(a, a**2, color='#e35f62', marker='*', linewidth=2)
plt.plot(a, a**3, color='springgreen', marker='^', markersize=9)
plt.grid(True, axis='y', color='gray', alpha=0.5, linestyle='--')

plt.title('Sample graph')

plt.show()
```



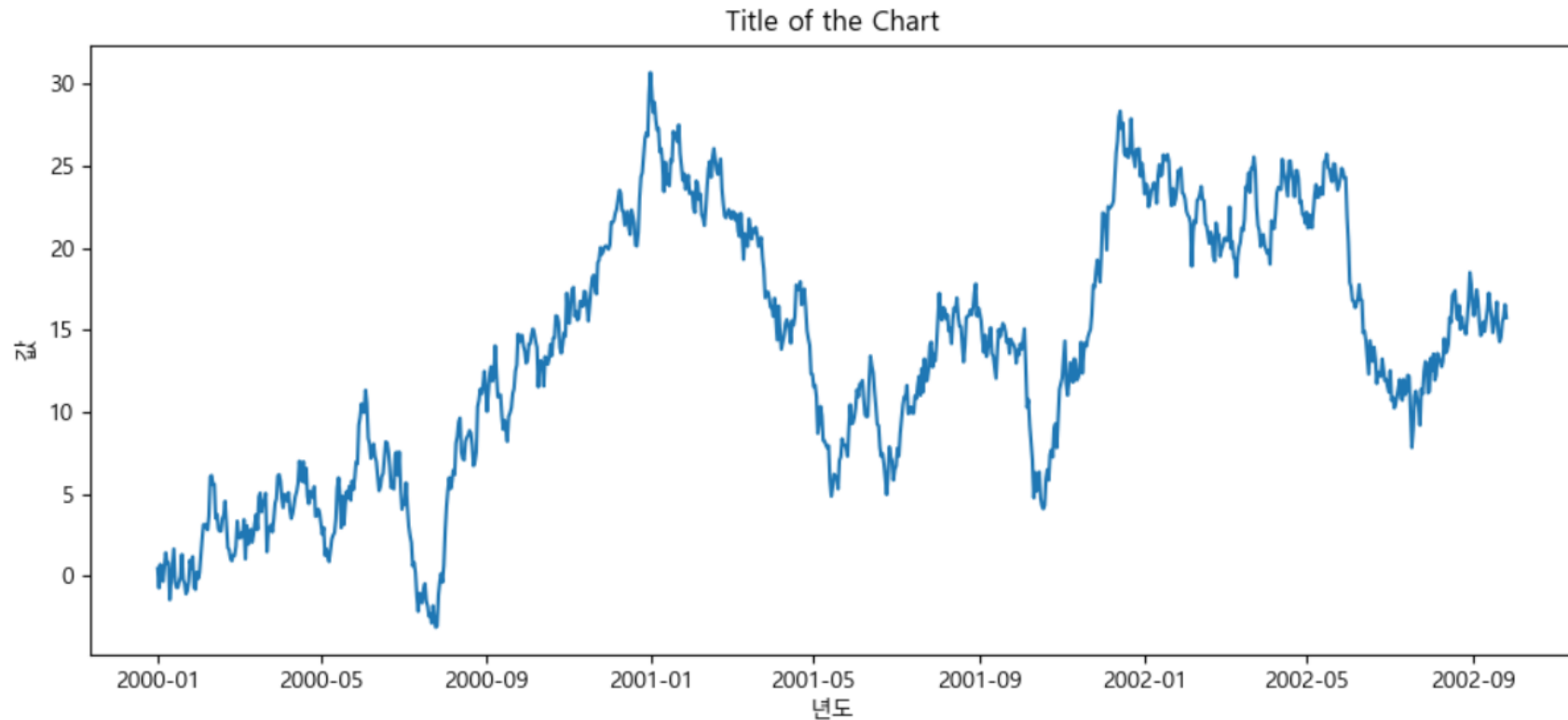
title() 함수로 그래프의 타이틀 지정이 가능하고  
loc="right|left|center" 속성 및 pad=수치 속성을 이용하여 위치와 오프셋을 지정할 수 있다.

## 5. 선그래프 - 활용1

python

2000/1/1 부터 1000개의 랜덤 데이터를 누적하고 시각화하기

```
np.random.seed(1234)
ts = pd.Series(np.random.randn(1000),
               index=pd.date_range('1/1/2000', periods=1000))
```



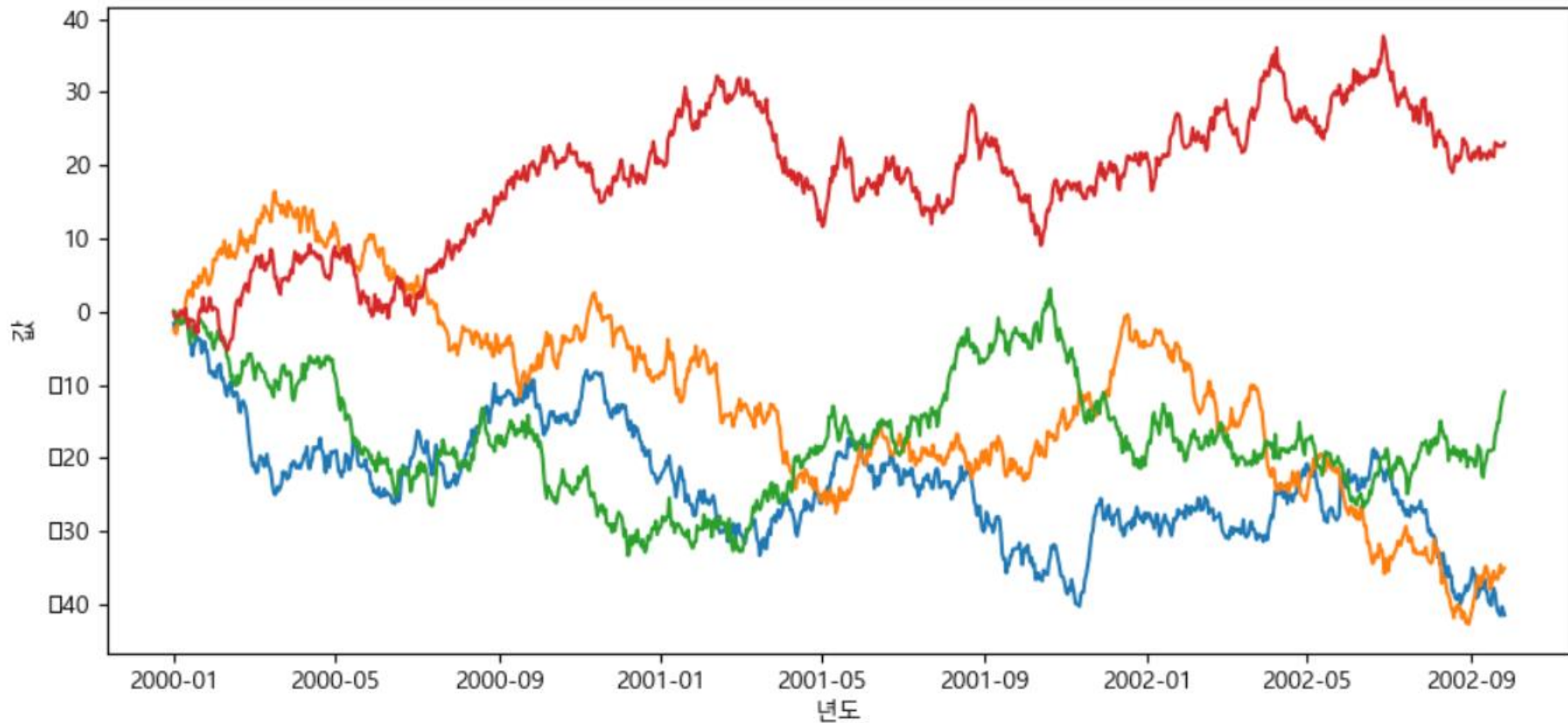


## 5. 선그래프 - 활용2

python

2000/1/1 부터 1000개의 랜덤 데이터를 누적하고 시각화하기

```
df = pd.DataFrame(np.random.randn(1000, 4),  
                  index=pd.date_range('1/1/2000', periods=1000),  
                  columns=list('ABCD'))
```



## 3장. 막대그래프

# 1. 막대 그래프

python

[https://matplotlib.org/2.0.2/api/pyplot\\_api.html?highlight=bar#matplotlib.pyplot.bar](https://matplotlib.org/2.0.2/api/pyplot_api.html?highlight=bar#matplotlib.pyplot.bar)

```
matplotlib.pyplot.bar(left, height, width=0.8, bottom=None, hold=None, data=None, **kwargs)
```

Make a bar plot.

Make a bar plot with rectangles bounded by:

left, left + width, bottom, bottom + height

(left, right, bottom and top edges)

## Parameters:

**left** : sequence of scalars

the x coordinates of the left sides of the bars

**height** : sequence of scalars

the heights of the bars

**width** : scalar or array-like, optional

the width(s) of the bars default: 0.8

**bottom** : scalar or array-like, optional

the y coordinate(s) of the bars default: None

**color** : scalar or array-like, optional

the colors of the bar faces

**edgecolor** : scalar or array-like, optional

the colors of the bar edges

**linewidth** : scalar or array-like, optional

width of bar edge(s). If None, use default linewidth; If 0, don't draw edge

**tick\_label** : string or array-like, optional

the tick labels of the bars default: None

**xerr** : scalar or array-like, optional

if not None, will be used to generate errorbar(s) on the bar chart default: None

**yerr** : scalar or array-like, optional

if not None, will be used to generate errorbar(s) on the bar chart default: None

**ecolor** : scalar or array-like, optional

specifies the color of errorbar(s) default: None

**capsize** : scalar, optional

determines the length in points of the error bar caps default: None, which will take the value from the `errorbar.capsize rcParam`.

**error\_kw** : dict, optional

dictionary of kwargs to be passed to errorbar method. `ecolor` or `capsize` may be specified here rather than as independent kwargs.

**align** : {'center', 'edge'}, optional

If 'edge', aligns bars by their left edges (for vertical bars) and by their bottom edges (for horizontal bars). If 'center', interpret the `left` argument as the coordinates of the centers of the bars. To align on the right edge pass a negative width.

**orientation** : {'vertical', 'horizontal'}, optional

The orientation of the bars.

**log** : boolean, optional

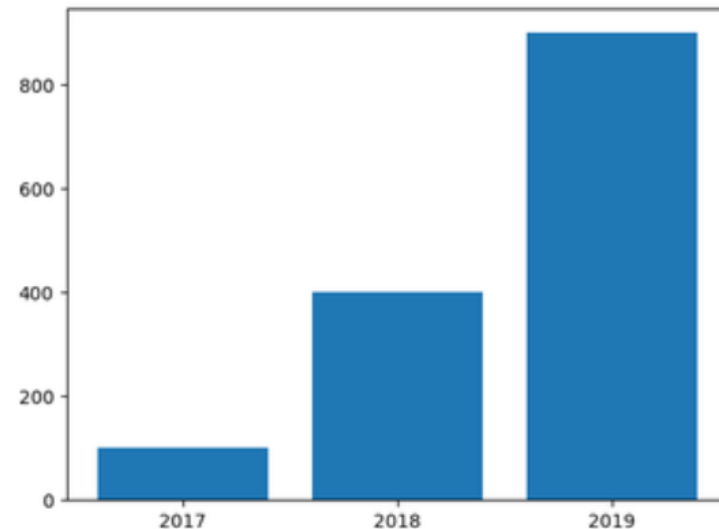
bar() 함수는 범주가 있는 데이터 값을 직사각형의 막대로 표현하는 그래프이다.

## 예제

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(3)
years = ['2017', '2018', '2019']
values = [100, 400, 900]

plt.bar(x, values)
plt.xticks(x, years)
plt.show()
```



막대 그래프 그리기.

# 1. 막대 그래프 - 스타일 지정

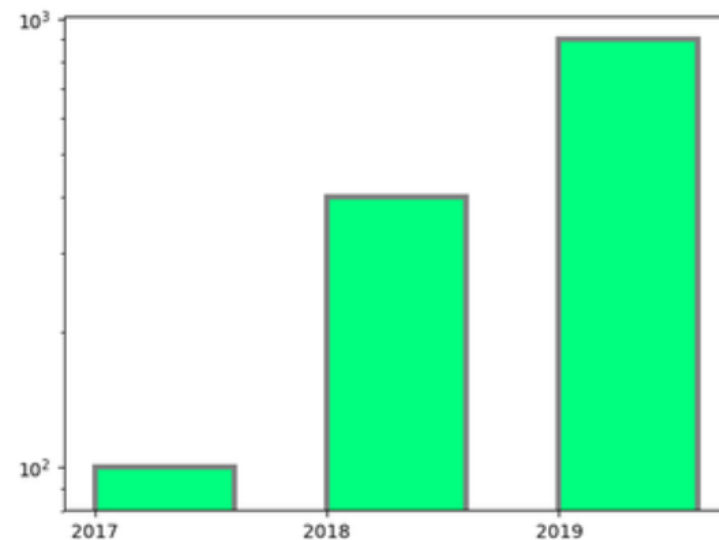
python

## 예제

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(3)
years = ['2017', '2018', '2019']
values = [100, 400, 900]

plt.bar(x, values, width=0.6, align='edge', color="springgreen",
        edgecolor="gray", linewidth=3, tick_label=years, log=True)
plt.show()
```



속성	
width	막대의 너비, 기본값은 0.8
align	tick과 막대의 위치 조절, 기본값은 center, edge는 막대 왼쪽 끝에 tick이 표시, {'center', 'edge'}
color	막대의 색상
edgecolor	막대의 테두리 색상
linewidth	테두리 두께
tick_label	tick 라벨, 리스트로 지정 가능
log	True로 지정하면 y축이 로그 스케일로 표시

# 1. 막대 그래프 – stacked bar

python

```
import numpy as np
import matplotlib.pyplot as plt

plt.rc("font", family="Malgun Gothic") # 한글 처리

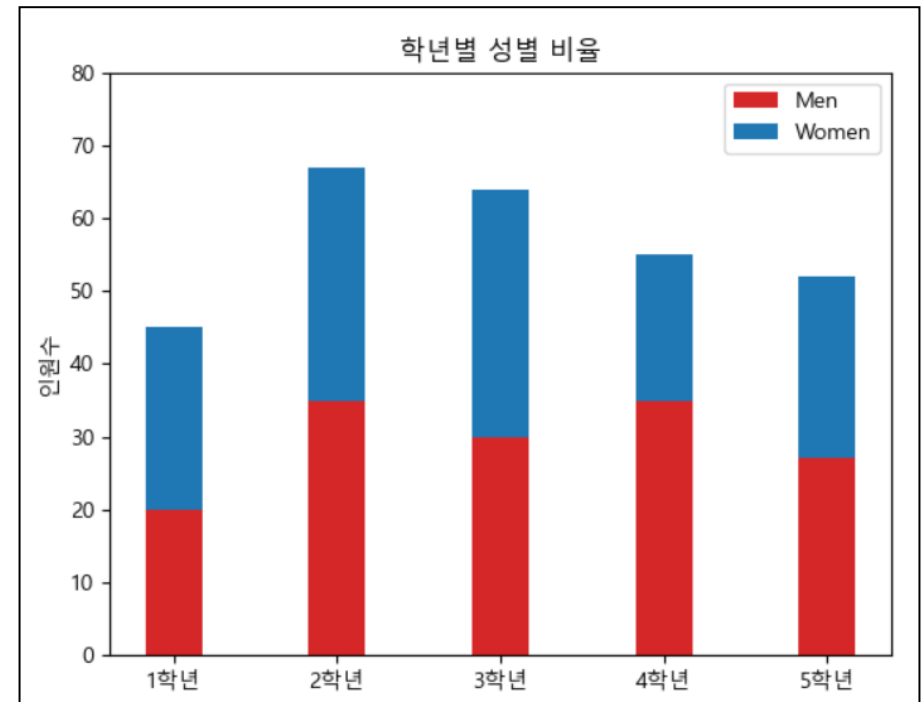
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)

ind = np.arange(5)
width = 0.35

p1 = plt.bar(ind, menMeans, width, color='#d62728')
p2 = plt.bar(ind, womenMeans, width, bottom=menMeans)

plt.ylabel('인원수')
plt.title('학년별 성별 비율')
plt.xticks(ind, ('1학년', '2학년', '3학년', '4학년', '5학년'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```



# 1. 막대 그래프 - 수평 막대

python

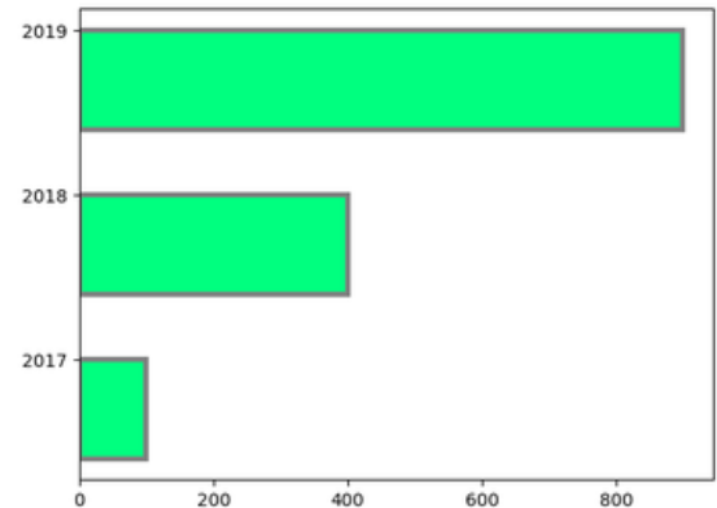
barh() 함수는 범주가 있는 데이터 값을 직사각형의 수평 막대로 표현하는 그래프이다.

## 예제

```
import matplotlib.pyplot as plt
import numpy as np

y = np.arange(3)
years = ['2017', '2018', '2019']
values = [100, 400, 900]

plt.barh(y, values, height=-0.6, align='edge', color="springgreen",
         edgecolor="gray", linewidth=3, tick_label=years, log=False)
plt.show()
```



## 4장. 산점도 ( scatter )



# 1. 산점도 그래프

python

[https://matplotlib.org/2.0.2/api/pyplot\\_api.html?highlight=scatter#matplotlib.pyplot.scatter](https://matplotlib.org/2.0.2/api/pyplot_api.html?highlight=scatter#matplotlib.pyplot.scatter)

```
matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None,
verts=None, edgecolors=None, hold=None, data=None, **kwargs)
```

Make a scatter plot of x vs y

Marker size is scaled by s and marker color is mapped to c

## Parameters:

**x, y** : array\_like, shape (n, )

Input data

**s** : scalar or array\_like, shape (n, ), optional

size in points<sup>2</sup>. Default is rcParams['lines.markersize'] \*\* 2.

**c** : color, sequence, or sequence of color, optional, default: 'b'

c can be a single color format string, or a sequence of color specifications or numbers to be mapped to colors using the cmap and norm specified via kwargs. c should not be a single numeric RGB or RGBA sequence because that is indistinguishable from an array of values to be colormapped. c can be a 2-D array in which the rows are RGB or RGBA and the columns are the values to be mapped. c can be a 1-D array in which the rows are the values to be mapped and the columns are the values to be mapped.

**marker** : MarkerStyle, optional, default: 'o'

See [markers](#) for more information on the different styles of markers scatter plot supports. A string instance of the class or the text shorthand for a particular marker.

**cmap** : Colormap, optional, default: None

A Colormap instance or registered name. cmap is only used if c is an array of values to be mapped.

**norm** : Normalize, optional, default: None

A Normalize instance is used to scale luminance data to 0, 1. norm is only used if c is an array of values to be mapped. None, use the default normalize().

**vmin, vmax** : scalar, optional, default: None

**alpha** : scalar, optional, default: None

The alpha blending value, between 0 (transparent) and 1 (opaque)

**linewidths** : scalar or array\_like, optional, default: None

If None, defaults to (lines.linewidth,).

**verts** : sequence of (x, y), optional

If marker is None, these vertices will be used to construct the marker. The center of the marker is located at (0,0) in normalized units. The overall marker is rescaled by s.

**edgecolors** : color or sequence of color, optional, default: None

If None, defaults to 'face'

If 'face', the edge color will always be the same as the face color.

If it is 'none', the patch boundary will not be drawn.

For non-filled markers, the edgecolors kwarg is ignored and forced to 'face' internally.

# 1. 산점도 그래프

python

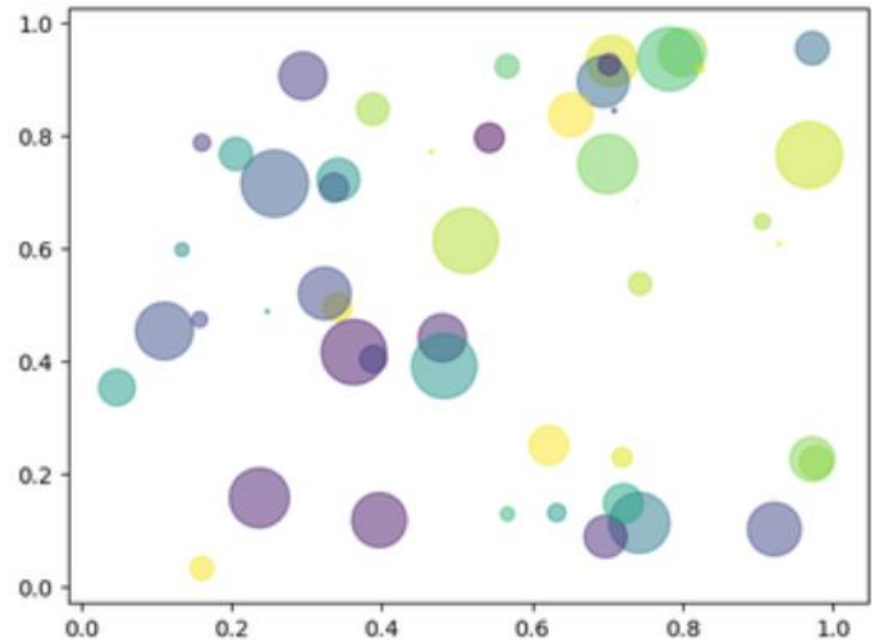
산점도(scatter plot)는 두 변수의 상관 관계를 직교 좌표계의 평면에 데이터를 점으로 표현하는 그래프이다.  
plt.scatter() 함수를 사용한다.

## 예제

```
import matplotlib.pyplot as plt
import numpy as np

N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



## 5장. 파이 차트

# 1. 파이 차트

python

[https://matplotlib.org/2.0.2/api/pyplot\\_api.html?highlight=pie#matplotlib.pyplot.pie](https://matplotlib.org/2.0.2/api/pyplot_api.html?highlight=pie#matplotlib.pyplot.pie)

```
matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=None, radius=None, counter-clockwise=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False, hold=None, data=None)
```

Plot a pie chart.

Make a pie chart of array *x*. The fractional area of each wedge is given by  $x/\text{sum}(x)$ . If  $\text{sum}(x) \leq 1$ , then the values of *x* give the fractional area directly and the array will not be normalized. The wedges are plotted counterclockwise, by default starting from the x-axis.

## Parameters:

**x** : array-like

The input array used to make the pie chart.

**explode** : array-like, optional, default: None

If not *None*, is a 1 $\times$ len(*x*) array which specifies the fraction of the radius with which to

**labels** : list, optional, default: None

A sequence of strings providing the labels for each wedge

**colors** : array-like, optional, default: None

A sequence of matplotlib color args through which the pie chart will cycle. If *None*, currently active cycle.

**autopct** : None (default), string, or function, optional

If not *None*, is a string or function used to label the wedges with their numeric value inside the wedge. If it is a format string, the label will be `fmt%pct`. If it is a function,

**pctdistance** : float, optional, default: 0.6

The ratio between the center of each pie slice and the start of the text generated by *autopct* is *None*.

**shadow** : bool, optional, default: False

Draw a shadow beneath the pie.

**labeldistance** : float, optional, default: 1.1

**startangle** : float, optional, default: None

If not *None*, rotates the start of the pie chart by *angle* degrees counterclockwise from the x-axis.

**radius** : float, optional, default: None

The radius of the pie, if *radius* is *None* it will be set to 1.

**counterclock** : bool, optional, default: True

Specify fractions direction, clockwise or counterclockwise.

**wedgeprops** : dict, optional, default: None

Dict of arguments passed to the wedge objects making the pie. For example, you can pass in `wedgeprops = {'linewidth': 3}` to set the width of the wedge border lines equal to 3. For more details, look at the doc/arguments of the wedge object. By default `clip_on=False`.

**textprops** : dict, optional, default: None

Dict of arguments to pass to the text objects.

**center** : list of float, optional, default: (0, 0)

Center position of the chart. Takes value (0, 0) or is a sequence of 2 scalars.

**frame** : bool, optional, default: False

Plot axes frame with the chart if true.

# 1. 파이 차트

파이 차트(원그래프)는 범주별 구성 비율을 원형으로 표현한 그래프이다.

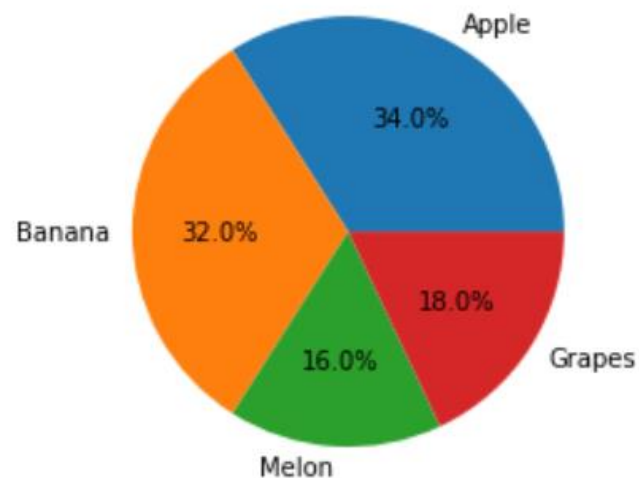
plt.pie() 함수를 사용한다.

## 예제

```
import matplotlib.pyplot as plt

ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']

plt.pie(ratio, labels=labels, autopct='%.1f%%')
plt.show()
```



# 1. 파이 차트 - 속성

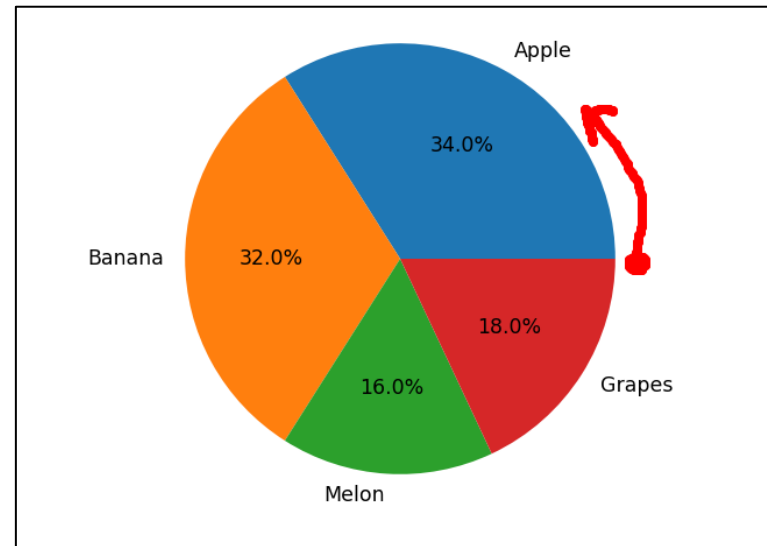
python

속성	설명
autopct	부채꼴 안에 표시될 숫자의 형식
startangle	부채꼴이 그려지는 시작 각도. 기본은 0 이고 시계반대 방향으로 표시됨.
counterclock	False로 지정하면 시계방향순으로 표시됨
explode	부채꼴이 파이차트의 중심에서 벗어나는 정도 (비율)
shadow	True로 지정하면 그림자 효과

## 실습1

```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']

plt.pie(ratio,
        labels=labels,
        autopct='%.1f%%')
plt.show()
```



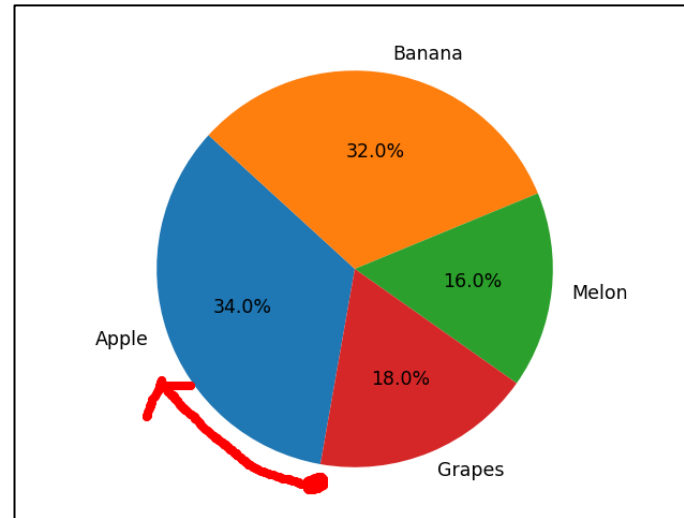
# 1. 파이 차트 - 속성

python

## 실습2

```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']

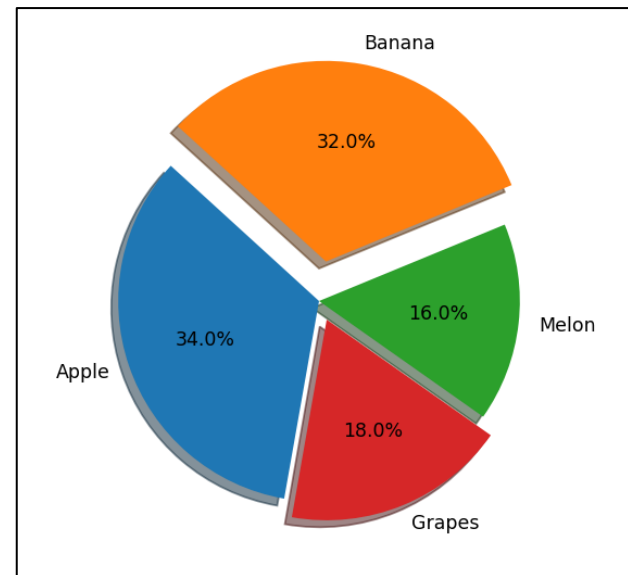
plt.pie(ratio,
        labels=labels,
        autopct='%.1f%%',
        startangle=260,
        counterclock=False)
plt.show()
```



## 실습3

```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']
explode = [0, 0.20, 0, 0.10]

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



## 6장. 히스토그램



# 1. 히스토그램

python

[https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.hist.html?highlight=t=hist#matplotlib-pyplot-hist](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html?highlight=t=hist#matplotlib-pyplot-hist)

## matplotlib.pyplot.hist ¶

```
matplotlib.pyplot.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)
```

Plot a histogram.

Compute and draw the histogram of *x*. The return value is a tuple (*n*, *bins*, *patches*) or (*[n0, n1, ...]*) contains multiple data. See the documentation of the *weights* parameter to draw a histogram of all

Multiple data can be provided via *x* as a list of datasets of potentially different length (*[x0, x1, ...]*), *x* is a dataset. Note that the ndarray form is transposed relative to the list form.

Masked arrays are not supported.

The *bins*, *range*, *weights*, and *density* parameters behave as in `numpy.histogram`.

### Parameters:

**x**: (n,) array or sequence of (n,) arrays

Input values, this takes either a single array or a sequence of arrays of the same length.

**bins**: int or sequence or str, default: `rcParams["hist.bins"]` (default: 10)

If *bins* is an integer, it defines the number of equal-width bins in the range.

If *bins* is a sequence, it defines the bin edges, including the left edge of the first bin, and the right edge of the last bin; in this case, bins may be unequally spaced. All but the last bin are open (by default), in other words, if *bins* is:

```
[1, 2, 3, 4]
```

then the first bin is `[1, 2)` (including 1, but excluding 2) and the last bin is `[3, 4]`, which includes 4.

If *bins* is a string, it is one of the binning strategies supported by `numpy.histogram`: 'auto', 'fd', 'doane', 'scott', 'stone', 'rice', 'sturges', or 'sqrt'.

**range**: tuple or None, default: None

The lower and upper range of the bins. Lower and upper outliers are ignored. If not provided, *range* is `(x.min(), x.max())`. Range has no effect if *bins* is a sequence.

If *bins* is a sequence or *range* is specified, autoscaling is based on the specified bin range instead of the range of *x*.

**density**: bool, default: False

If True, draw and return a probability density: each bin will display the bin's raw count divided by the total number of counts and the bin width ( $\text{density} = \text{counts} / (\text{sum}(\text{counts}) * \text{np.diff}(\text{bins}))$ ), so that the area under the histogram integrates to 1 ( $\text{np.sum}(\text{density} * \text{np.diff}(\text{bins})) == 1$ ).

If *stacked* is also True, the sum of the histograms is normalized to 1.

**weights**: (n,) array-like or None, default: None

An array of weights, of the same shape as *x*. Each value in *x* only contributes its associated weight towards the bin count (instead of 1). If *density* is True, the weights are normalized, so that the integral of the density over the range remains 1.

This parameter can be used to draw a histogram of data that has already been binned, e.g. using `numpy.histogram` (by treating each bin as a single point with a weight equal to its count)

```
counts, bins = np.histogram(data)
plt.hist(bins[:-1], bins, weights=counts)
```

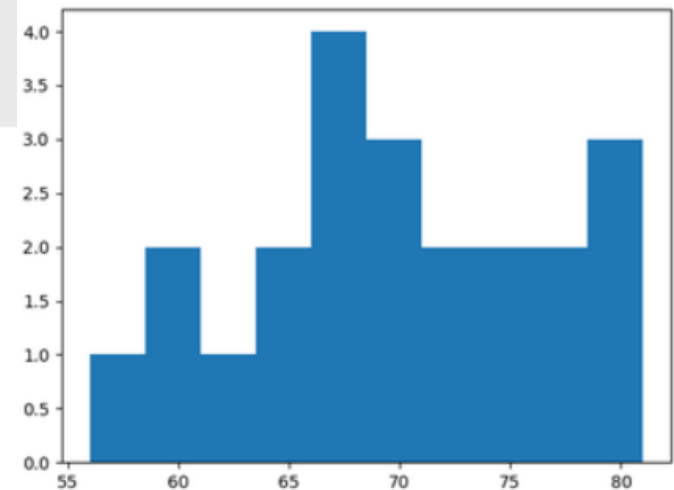
(or you may alternatively use `bar()`).

## 예제

```
import matplotlib.pyplot as plt

weight = [68, 81, 64, 56, 78, 74, 61, 77, 66, 68, 59, 71, 80, 59, 67, 81, 69, 73, 69, 74, 70, 65]

plt.hist(weight)
plt.show()
```



도수분포표: 특정 구간에 속하는 자료의 개수를 표현

히스토그램: 도수분포표를 시각화 하여 막대 그래프로 표현.

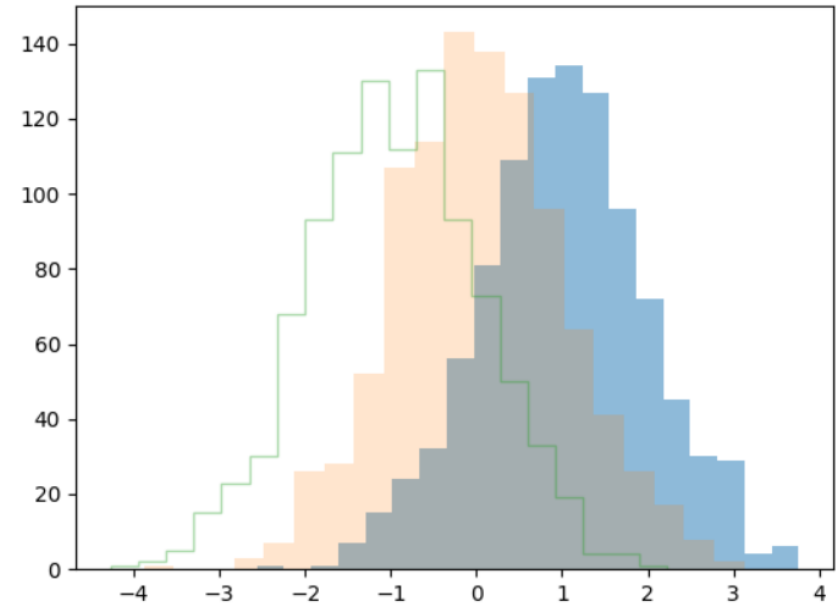
막대 그래프는 y값만 고려하지만, 히스토그램은 x,y 모두 고려한다.

# 1. 히스토그램

python

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1234)
a = np.random.randn(1000) + 1
b = np.random.randn(1000)
c = np.random.randn(1000) - 1

plt.hist(a, bins=20, alpha=0.5)
plt.hist(b, bins=20, alpha=0.2)
plt.hist(c, bins=20, alpha=0.4, histtype="step")
plt.show()
```



## 7장. Hexagonal bin plot (육각형 bin plot)

# 1. hexbin

[https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.hexbin.html?highlight=hexbin#matplotlib-pyplot-hexbin](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hexbin.html?highlight=hexbin#matplotlib-pyplot-hexbin)

## matplotlib.pyplot.hexbin

```
matplotlib.pyplot.hexbin(x, y, C=None, gridsize=100, bins=None, xscale='linear', yscale='linear', extent=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors='face', reduce_C_function=<function mean at 0x7f1cbe2b700>, mincnt=None, marginals=False, *, data=None, **kwargs)
```

[source]

Make a 2D hexagonal binning plot of points  $x, y$ .

If  $C$  is *None*, the value of the hexagon is determined by the number of points in the hexagon. ( $x[i]$ ,  $y[i]$ ). For each hexagon, these values are reduced using *reduce\_C\_function*.

### Parameters:

**x, y** : array-like

The data positions.  $x$  and  $y$  must be of the same length.

**C** : array-like, optional

If given, these values are accumulated in the bins. Otherwise, the same length as  $x$  and  $y$ .

**gridsize** : int or (int, int), default: 100

If a single int, the number of hexagons in the  $x$ -direction. The chosen such that the hexagons are approximately regular.

Alternatively, if a tuple  $(nx, ny)$ , the number of hexagons in the

**bins** : 'log' or int or sequence, default: None

Discretization of the hexagon values.

- If *None*, no binning is applied; the color of each hexagon is the number of points in the hexagon.
- If 'log', use a logarithmic scale for the colormap. Internally, the hexagon color is `norm(LogNorm())`.
- If an integer, divide the counts in the specified number of bins accordingly.
- If a sequence of values, the values of the lower bound of each bin.

**xscale** : {'linear', 'log'}, default: 'linear'

Use a linear or log10 scale on the horizontal axis.

**yscale** : {'linear', 'log'}, default: 'linear'

Use a linear or log10 scale on the vertical axis.

**mincnt** : int > 0, default: None

If not *None*, only display cells with more than *mincnt* number of points in the cell.

**marginals** : bool, default: False

If *marginals* is *True*, plot the marginal density as colormapped rectangles along the bottom of the  $x$ -axis and left of the  $y$ -axis.

**extent** : float, default: None

The limits of the bins. The default assigns the limits based on *gridsize*,  $x$ ,  $y$ , *xscale* and *yscale*.

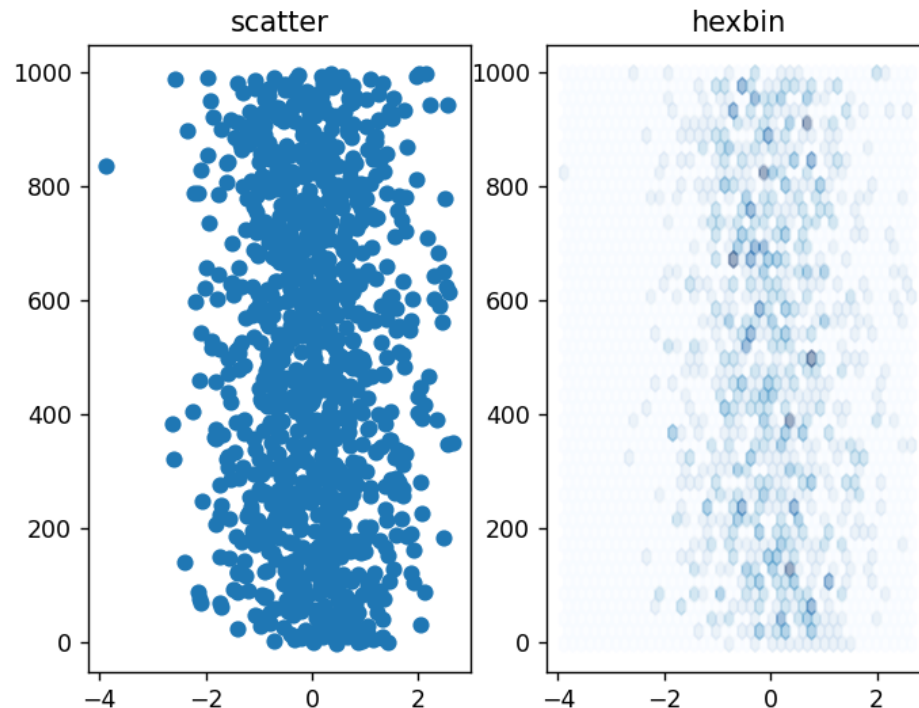
If *xscale* or *yscale* is set to 'log', the limits are expected to be the exponent for a power of 10. E.g. for  $x$ -limits of 1 and 50 in 'linear' scale and  $y$ -limits of 10 and 1000 in 'log' scale, enter (1, 50, 1, 3).

Order of scalars is (left, right, bottom, top).

# 1. hexbin

python

hexbin plot은 데이터가 많을 때, 각각의 점을 산점도로 표현할 때의 단점을 보완할 수 있다.  
기본적으로 (x,y)점 주변의 개수에 대한 히스토그램이 계산된다.  
따라서 데이터가 많으면 색상이 짙어진다.



## 8장. box plot

# 1. box plot

[https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.boxplot.html?highlight=boxplot#matplotlib-pyplot-boxplot](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html?highlight=boxplot#matplotlib-pyplot-boxplot)

## matplotlib.pyplot.boxplot

```
matplotlib.pyplot.boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None, widths=None, patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=None, meanline=None, showmeans=None, showcaps=None, showbox=None, showfliers=None, boxprops=None, labels=None, flierprops=None, medianprops=None, meanprops=None, capprops=None, whiskerprops=None, manage_ticks=True, autorange=False, zorder=None, *, data=None)
```

Make a box and whisker plot.

Make a box and whisker plot for each column of *x* or each vector in sequence *x*. The box extends from the lower quartile to the upper quartile, with a line at the median. The whiskers extend from the box to show the range of the data. Flier points are plotted outside the whiskers.

### Parameters:

**x**: Array or a sequence of vectors.

The input data.

**notch**: bool, default: False

Whether to draw a notched box plot (*True*), or a rectangular box plot (*False*). The notch indicates the confidence interval (CI) around the median. The documentation for *bootstrap* locations of the notches are computed by default, but their locations may also be controlled by the *conf\_intervals* parameter.

### Note

In cases where the values of the CI are less than the lower quartile or greater than the upper quartile, the notches will extend beyond the box, giving it a distinctive "flipped" appearance. This is expected behavior and consistent with other statistical visualization packages.

**sym**: str, optional

The default symbol for flier points. An empty string (") hides the fliers. If *None*, then the fliers default to 'b+'. More control is provided by the *flierprops* parameter.

**vert**: bool, default: True

If *True*, draws vertical boxes. If *False*, draw horizontal boxes.

**whis**: float or (float, float), default: 1.5

The position of the whiskers.

If a float, the lower whisker is at the lowest datum above  $Q1 - whis * (Q3 - Q1)$ , and the upper whisker at the highest datum below  $Q3 + whis * (Q3 - Q1)$ , where  $Q1$  and  $Q3$  are the first and third quartiles. The default value of *whis* = 1.5 corresponds to Tukey's original definition of boxplots.

If a pair of floats, they indicate the percentiles at which to draw the whiskers (e.g., (5, 95)). In particular, setting this to (0, 100) results in whiskers covering the whole range of the data.

In the edge case where  $Q1 == Q3$ , *whis* is automatically set to (0, 100) (cover the whole range of the data) if *autorange* is *True*.

Beyond the whiskers, data are considered outliers and are plotted as individual points.

**bootstrap**: int, optional

Specifies whether to bootstrap the confidence intervals around the median for notched boxplots. If *bootstrap* is *None*, no bootstrapping is performed, and notches are calculated using a Gaussian-based asymptotic approximation (see McGill, R., Tukey, J.W., and Larsen, W.A., 1978, and Kendall and Stuart, 1967). Otherwise, *bootstrap* specifies the number of times to bootstrap the median to determine its 95% confidence intervals. Values between 1000 and 10000 are recommended.

**usermedians**: 1D array-like, optional

A 1D array-like of length  $\text{len}(x)$ . Each entry that is not *None* forces the value of the median for the corresponding dataset. For entries that are *None*, the medians are computed by Matplotlib as normal.

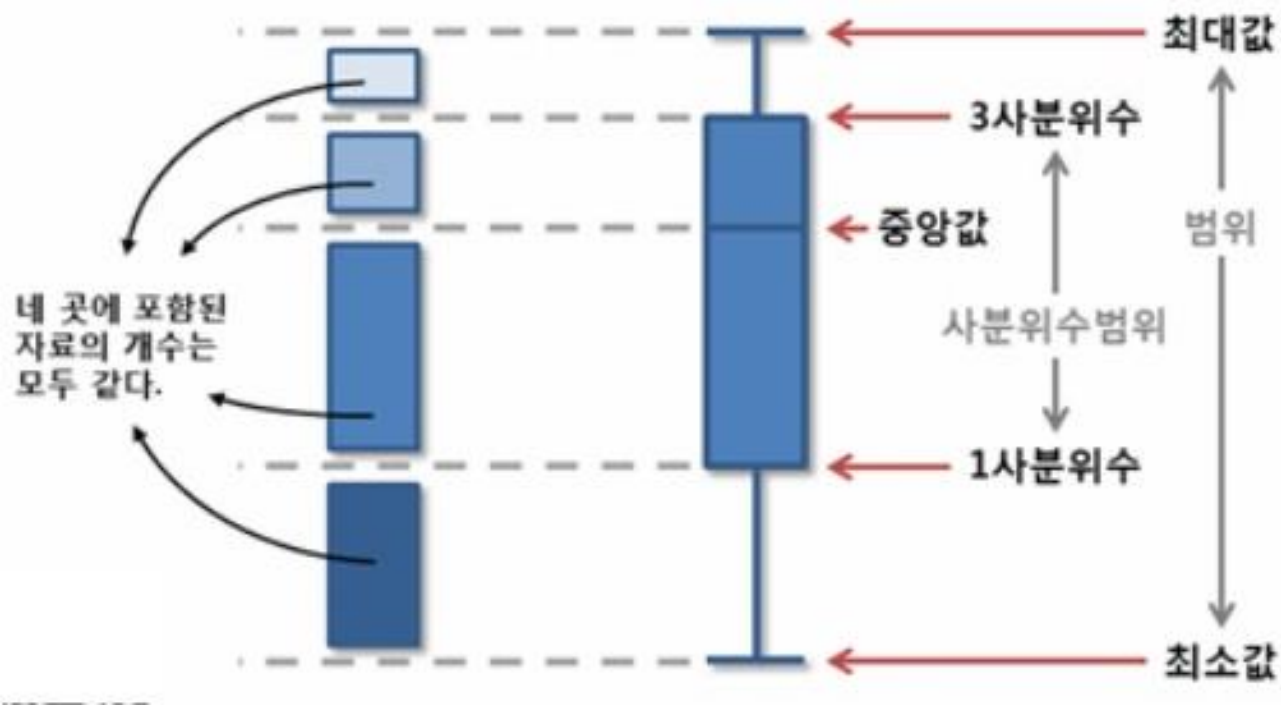


# 1. box plot

python

사분위수(quartile)는 자료를 동일한 비율로 4등분 할 때의 세 위치를 의미한다.

1사분위수( 25% 지점, Q1), 2사분위수( 50% 지점, Q2, 중앙값), 3사분위수( 75% 지점, Q3)이고  $Q3 - Q1$  값을 사분위범위(InterQuartile Range: IQR)이라고 부른다.



# 1. box plot

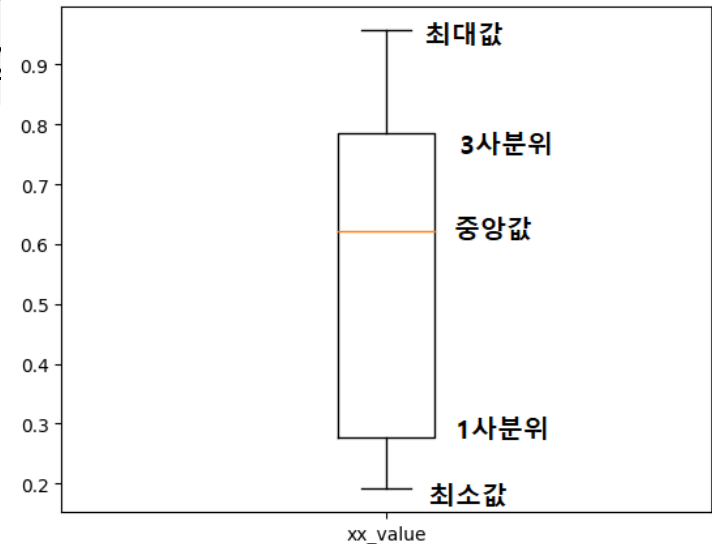
python

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
np.random.seed(1234)
```

```
x = np.random.random(9)
plt.boxplot(x, labels=['xx_value'])
plt.show()
```

```
print("3사분위:", np.quantile(x, 0.75)) # 0.7853585837137692
print("2사분위:", np.quantile(x, 0.5)) # 0.6221087710398319
print("1사분위:", np.quantile(x, 0.25)) # 0.2764642551430967
```

```
'''
[0.9581393536837052, 0.8018721775350193,
0.7853585837137692, # ( 3사분위 )
0.7799758081188035,
0.6221087710398319, # 중앙값 ( 2사분위 )
0.4377277390071145,
0.2764642551430967, # ( 1사분위 )
0.2725926052826416, 0.1915194503788923]
'''
```

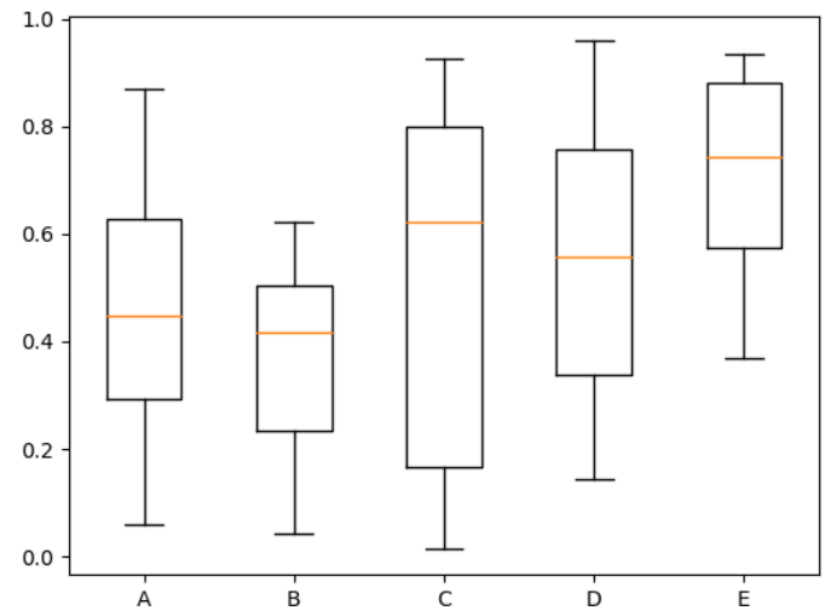


# 1. box plot

*python*

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
np.random.seed(1234)

df = pd.DataFrame(np.random.rand(10, 5), columns=['A', 'B', 'C', 'D', 'E'])
print(df.describe())
plt.boxplot(df, labels=df.columns)
plt.show()
```





감사합니다.

---