



IT 파이썬. 딥러닝

시각화1

강사 이수현

데이터 시각화

- Matplotlib
- 맷플롯립의 역할
- 데이터 종류

머신러닝에 필요한 라이브러리



1. 배열 연산에 강한, NumPy
2. 데이터 분석에 유용한, Pandas
3. 데이터 시각화에 필요한, Matplotlib

- 머신러닝을 구현하기 위해 수집한 데이터가 학습데이터 필요
- 이 데이터로 의미 있는 결과를 예측할 수 있도록 하는 데 Matplotlib(맷플롯립) 라이브러리를 사용하면 편리
- Matplotlib은 데이터를 다양한 방법으로 시각화
- 데이터의 분포나 경향성, 데이터 간의 관계 등을 파악하는 데 유용한 라이브러리

- 데이터 시각화는 데이터 분석 결과를 쉽게 이해할 수 있도록 차트나 그래프로 표현하는 것이다.
- 가장 많이 사용되는 Python 라이브러리가 맷플롯립(Matplotlib)이다.
- 맷플롯립을 사용하여 데이터를 시각화하면 데이터의 분포 및 데이터 간의 관계와 패턴 등을 한눈에 볼 수 있어 데이터의 특성을 쉽게 파악할 수 있다.

- **맷플롯립 라이브러리의 핵심 역할**

- **시각화: 데이터의 2차원 시각화, 산점도, 막대그래프, 파이 차트, 히스토그램, 상자 그림**
- **데이터 해석: 데이터 분포 확인, 데이터 간의 관계 파악, 패턴 분석**
- **다양한 그래프 스타일: 그래프를 그린 후 점과 선의 색, 크기 등 그래프 스타일 설정**

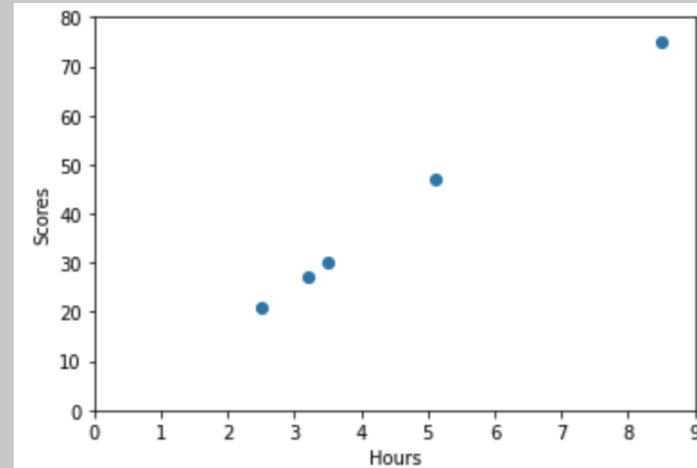
- 맷플롯립 라이브러리의 핵심 역할 요약

Matplotlib

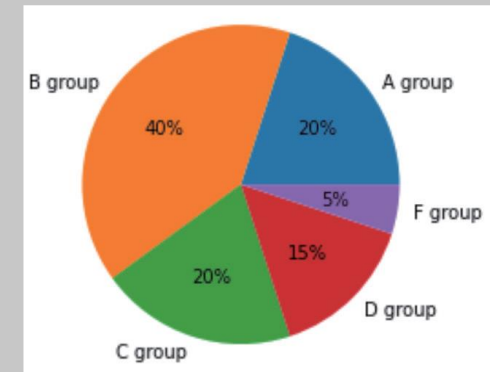


	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

Group	Per
A group	20
B group	40
C group	20
D group	15
F group	5

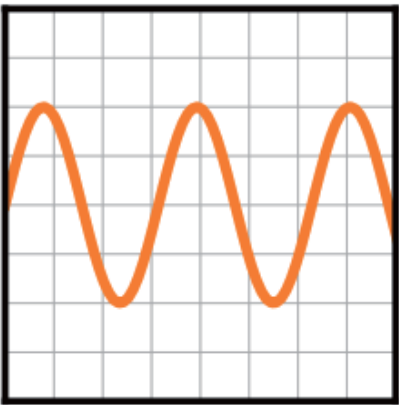
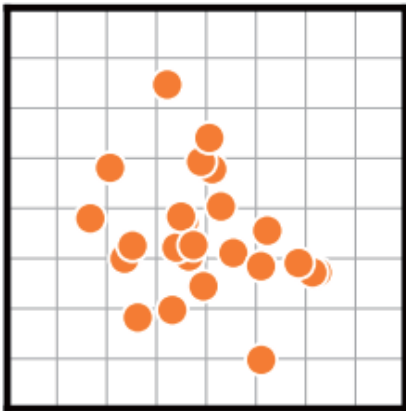
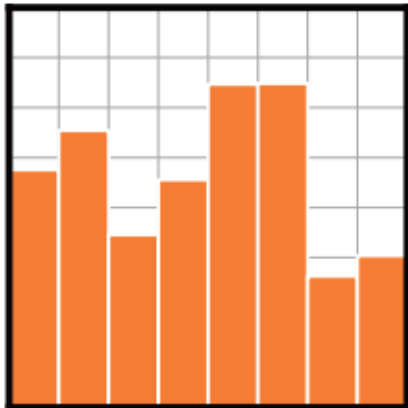


산점도

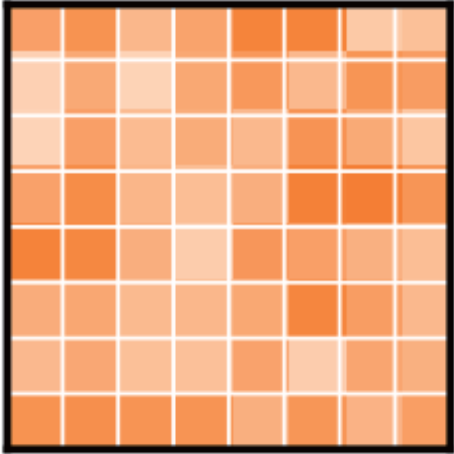
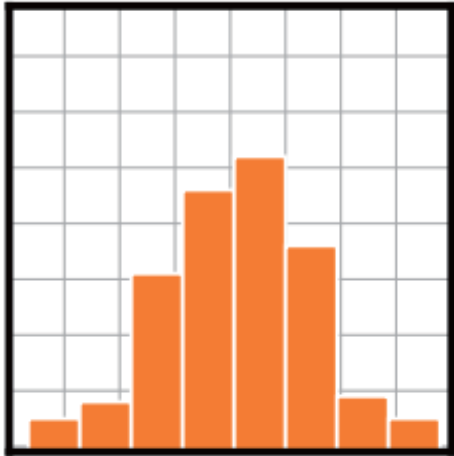
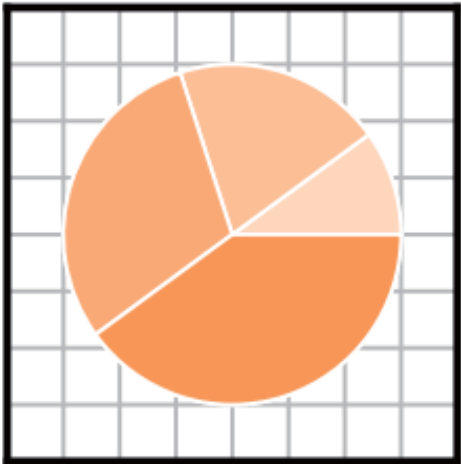


파이 차트


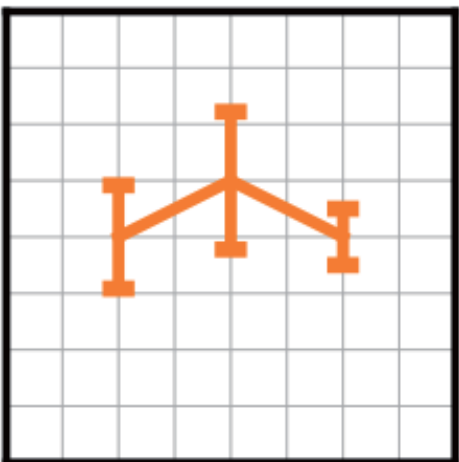
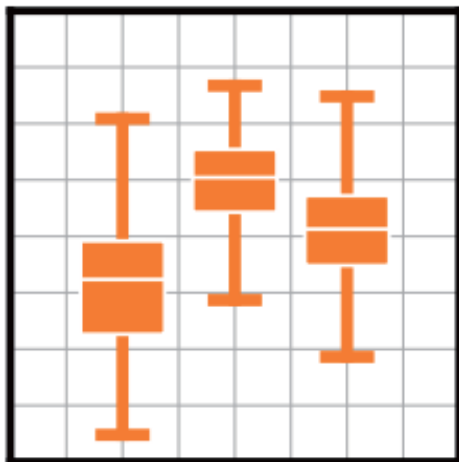
- 맷플롯립으로 표현할 수 있는 그래프는 다양하다.
- 소괄호() 안의 영문은 맷플롯립을 사용하여 각 그래프를 그리는 명령어

선그래프(plot)	산점도(scatter)	막대그래프(bar)
		
두 변수의 관계를 선으로 연결하여 시각화	두 변수 간의 관계를 나타내기 위해 좌표상의 점으로 시각화	범주형 데이터를 막대그래프로 시각화

- 소괄호() 안의 영문은 맷플롯립을 사용하여 각 그래프를 그리는 명령어

이미지 표시(imshow)	히스토그램(hist)	파이 차트(pie)
		
행렬을 만들어 각 칸에 색을 칠한 후 이미지로 시각화	수치형 데이터를 동일한 폭으로 그룹화하여 시각화	범주형 데이터의 비율을 원형의 여러 조각으로 시각화

- 소괄호() 안의 영문은 맷플롯립을 사용하여 각 그래프를 그리는 명령어

등고선(contourf)	오차 막대(errorbar)	상자 그림 또는 상자 수염 그림 박스플롯(boxplot)
		
공간상 동일한 값을 갖는 지점을 선으로 연결하여 시각화	데이터의 편차 시각화	자료로부터 얻은 통계값(최솟값, 최댓값, 중앙값, 1·3사분위)시각 화

- heatmap, quiver, text, fill, violinplot, eventplot, hexbin, xcorr 등 다양한 그래프를 그릴 수 있다.

히트 맵(heat map)은 열을 뜻하는 히트(heat)와 지도를 뜻하는 맵(map)을 결합시킨 단어로, 일반적으로 숫자 데이터를 색상으로 표현하는 것이 특징입니다. matplotlib 모듈에서는 heatmap 함수는 존재하지 않습니다.

- Python에서 그래프를 그릴 때는 맷플롯립의 파이플롯 (pyplot) 모듈을 사용하므로 파이플롯만 불러온다.



```
1 #맷플롯립 중 그래프를 그리는 pyplot 모듈을 불러와 plt라는 별명으로 사용  
2 from matplotlib import pyplot as plt
```

- 맷플롯립 모듈을 불러오면 다음의 과정을 거치게 됨.

데이터 준비



그래프 시각화



결과 해석

선그래프

- 선그래프를 그리기 위해 다음과 같이 `plot()` 함수와 `show()` 함수를 사용
- 파이플롯 객체는 앞에서 설정한 'plt' 이다.

파이플롯 객체.`plot([x값], [y값])`

파이플롯 객체.`show()`

선그래프

- `plot()` 함수 안에 `[]`는 리스트 자료형
- `plot()` 함수를 사용하기 전에 `x값`과 `y값`의 범위를 사전에 설정할 수 있다.

파이플롯 객체.`plot([x값],[y값])`

파이플롯 객체.`show()`

선그래프

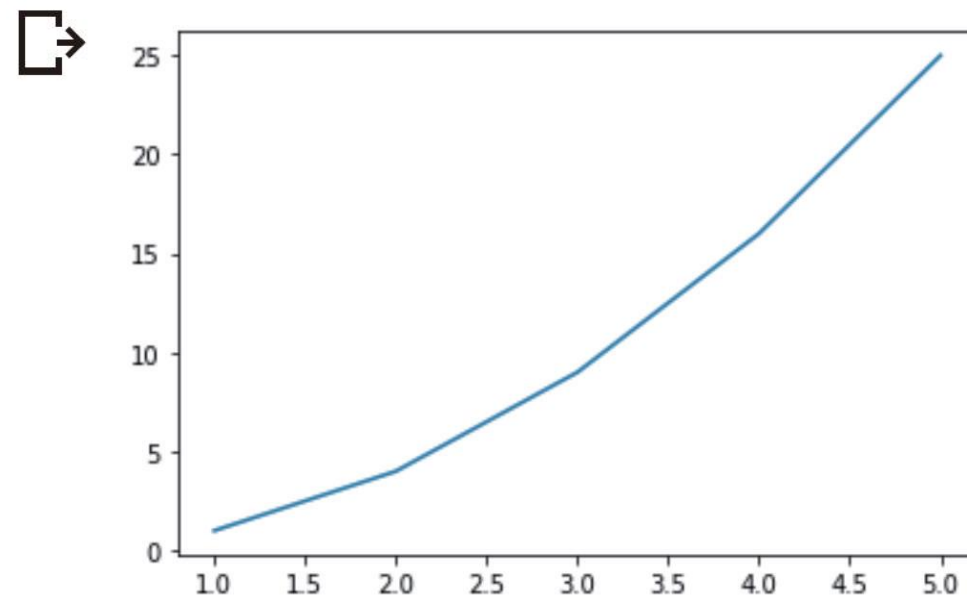
- x의 범위가 1,2,3,4,5이면서 $y = x^2$ 관계를 나타내는 선그래프는 리스트 또는 변수를 이용하여 그림
- 리스트 설정

```
1 from matplotlib import pyplot as plt
2 plt.plot([1,2,3,4,5],[1,4,9,16,25])
3 plt.show()
```

선그래프

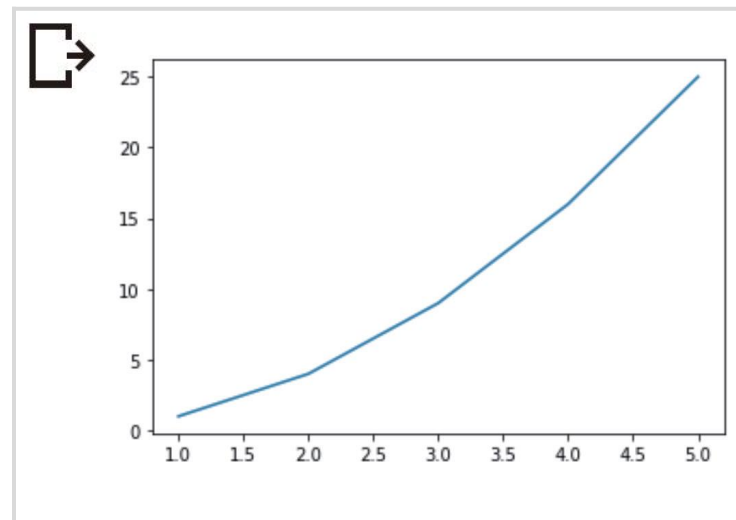
• 변수 설정

```
▶ 1 from matplotlib import pyplot as plt
   2 import numpy as np
   3 x = np.arange(1,6)
   4 y = x**2
   5
   6 plt.plot(x,y)
   7 plt.show()
```



선그래프

- `plot()` 함수 내에 `x`와 `y`의 범위를 직접 리스트 형태로 제시
- 넘파이 라이브러리의 `arange()` 함수와 변수를 이용하여 `x`의 범위를 설정하고 `y`를 `x`와의 관계를 나타내는 식으로 작성한 후 그래프를 그릴 수도 있다.



- 맷플롯립을 이용한 그래프와 명령어:

- 선그래프(plot), 산점도(scatter), 막대그래프(bar), 이미지 표시(imshow), 히스토그램(hist), 파이 차트(pie), 등고선(contourf), 오차 막대(errorbar), 박스플롯(boxplot) 등의 그래프를 그릴 수 있다.

- **넘파이의 `arange()` 함수**

넘파이의 `arange()` 함수는 그래프의 x 와 y 의 범위를 설정하는 데 사용

결과는 `range()` 함수와 결과가 유사

`range()` 함수는 리스트를 생성하고, 넘파이의 `arange()` 함수는 배열을 생성

넘파이의 `arange()` 함수

- 넘파이의 `arange()` 함수를 사용하는 방법

넘파이 객체.`arange(start, stop, step)`

- `start`: 생성할 배열의 시작 숫자, 생략할 경우 0부터 시작
- `stop`: 배열 생성을 중지하는 숫자, 제시된 정지값의 이전 값(`stop-1`)까지만 생성
- `step`: 출력할 숫자 사이의 간격, 생략할 경우 1

넘파이의 `arange()` 함수

넘파이 `arange()` 함수의 코드 예시와 그 결과는 다음과 같다.

코드 예시	결과
<code>np.arange(5)</code>	<code>array([0, 1, 2, 3, 4])</code>
<code>np.arange(2, 5)</code>	<code>array([2, 3, 4])</code>
<code>np.arange(1, 5, 2)</code>	<code>array([1, 3])</code>

산점도 막대그래프

- 앞에서 그래프를 그리기 위하여 `plot()` 함수를 사용하여 코드를 작성
- 그 밖에 산점도, 막대그래프, 이미지 표시, 파이 차트, 히스토그램, 히트맵 그리는 방법 알아보기

산점도(scatter)	막대그래프(bar)
파이플롯 객체 <code>.scatter(x,y)</code> <ul style="list-style-type: none">• <code>x,y</code>: 모두 수치형 데이터	파이플롯 객체 <code>.bar(x,y)</code> <ul style="list-style-type: none">• <code>x</code>: 수치형/범주형 데이터, <code>y</code>: 수치형 데이터

산점도 막대그래프

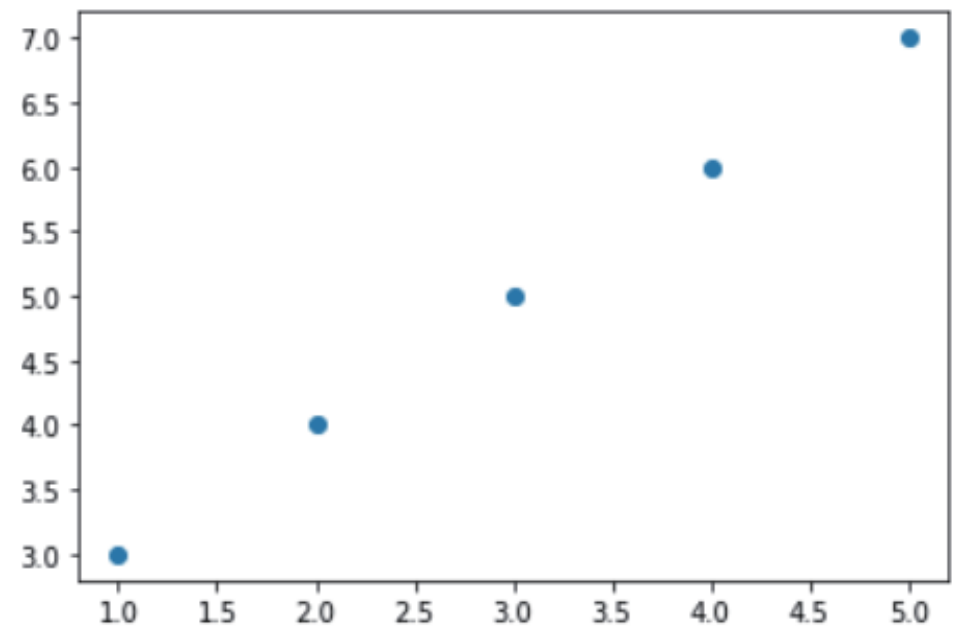
산점도(scatter)

파이플롯 객체 `.scatter(x,y)`

- `x,y`: 모두 수치형 데이터



```
1 from matplotlib import pyplot as plt
2 import numpy as np
3
4 x = np.arange(1,6)
5 y = np.arange(3,8)
6 plt.scatter(x,y)
7 plt.show()
```



산점도 막대그래프

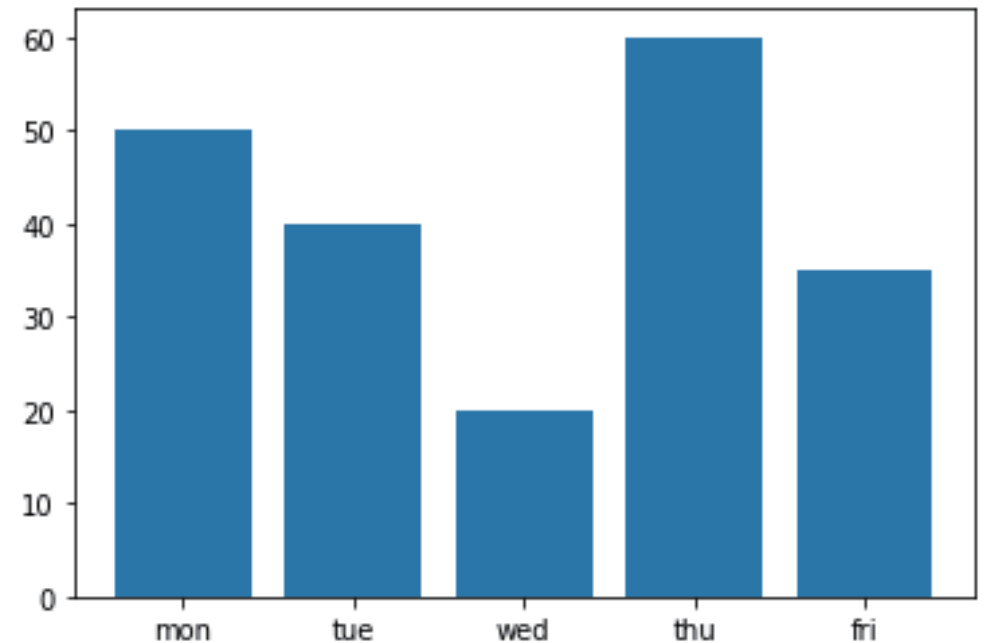
막대그래프(bar)

파이플롯 객체 .bar(x,y)

- x: 수치형/범주형 데이터, y: 수치형 데이터



```
1 from matplotlib import pyplot as plt
2 import numpy as np
3
4 x = ['mon', 'tue', 'wed', 'thu', 'fri']
5 y = [50, 40, 20, 60, 35]
6 plt.bar(x, y)
7 plt.show()
```



이미지 표시, 파이 차트

이미지 표시(imshow)	파이 차트(pie)
<p>파이플롯 객체 <code>.imshow(이미지)</code></p> <ul style="list-style-type: none">• 이미지: 이미지가 저장된 경로 또는 배열	<p>파이플롯 객체 <code>.pie(영역별 비율, labels=영역별 레이블, autopct='표시할 숫자 형식')</code></p> <ul style="list-style-type: none">• <code>autopct</code>: 부채꼴 안에 표시할 비율의 형식을 말하며, 소수점 아래 첫째 자리까지 표시하려면 <code>'%1.1f%%'</code>로 표기합니다.

이미지 표시 파이썬

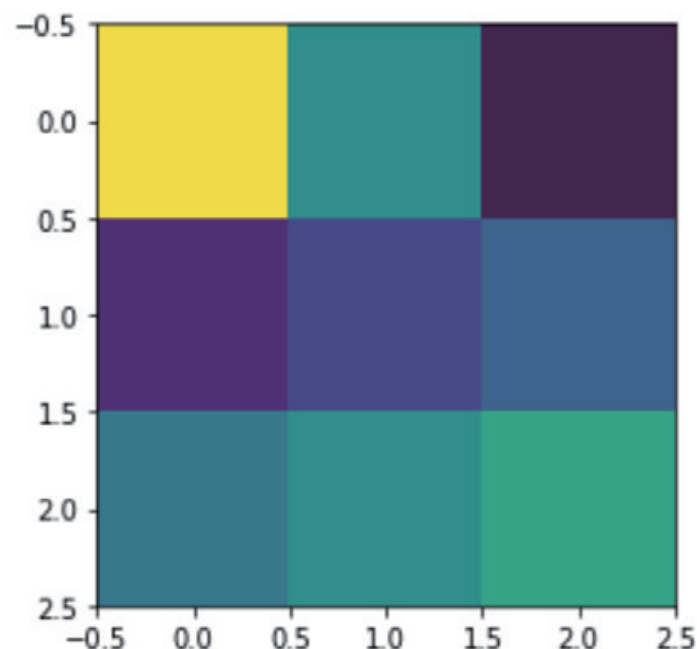
이미지 표시(imshow)

파이썬 객체 `.imshow(이미지)`

- 이미지: 이미지가 저장된 경로 또는 배열



```
1 from matplotlib import pyplot as plt
2 import numpy as np
3
4 image = np.array([[1,0.5,0],
5                   [0.1,0.2,0.3],
6                   [0.4,0.5,0.6]])
7 plt.imshow(image)
8 plt.show()
```



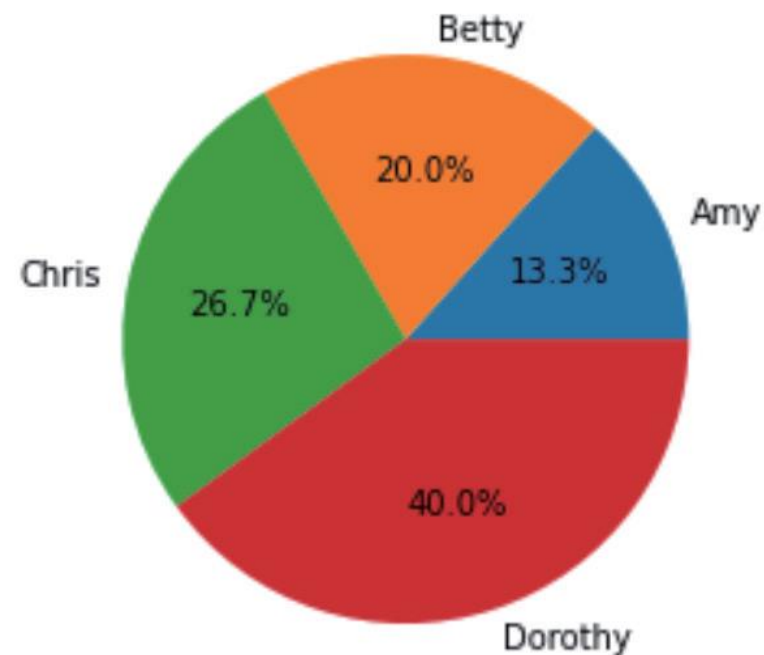
이미지 표시 파이 차트

파이 차트(pie)

파이플롯 객체.pie(영역별 비율, labels=영역별 레이블, autopct='표시할 숫자 형식')



```
1 from matplotlib import pyplot as plt
2
3 ratio=[20,30,40,60]
4 label=['Amy', 'Betty', 'Chris', 'Dorothy']
5 plt.pie(ratio, labels=label,
6         autopct='%1.1f%%')
7 plt.show()
```



히스토그램, 히트맵

히스토그램(hist)	히트맵
<p>파이플롯 객체.hist(데이터 리스트, bins=계급 개수)</p> <ul style="list-style-type: none">• 각 톤의 빈도를 막대 차트의 값으로 표시하여 이미지의 밝기를 측정하는 그래프• 계급 개수: 기본값은 10입니다.	<p>파이플롯 객체.matshow(배열명)</p>

히스토그램 히트맵

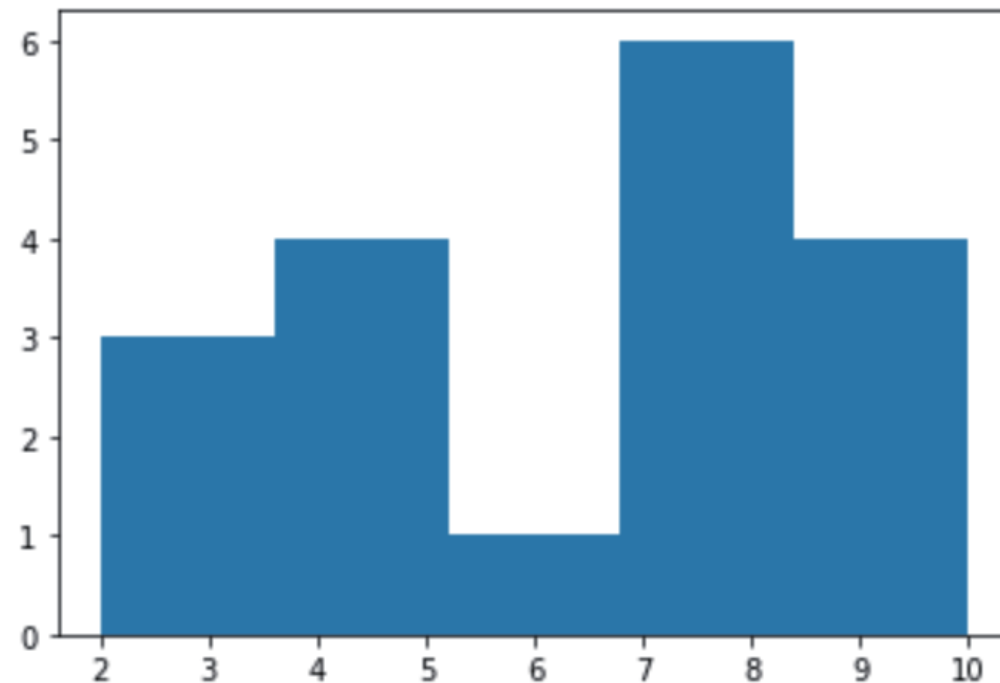
히스토그램(hist)

파이플롯 객체.hist(데이터 리스트, bins=계급 개수)

- 계급 개수: 기본값은 10입니다.



```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 data = [3,4,5,5,5,6,7,8,3,
5         2,7,7,7,8,9,10,10,9]
6 plt.hist(data, bins=5)
7 plt.show()
```



히스토그램 히트맵

히트맵

파이플롯 객체 `.matshow(배열명)`



```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 x=np.arange(1,5)
4 y=np.arange(2,6)
5 arr = ((x,y))
6 plt.matshow(arr)
7 plt.show()
```
















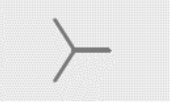






















속성 설정

- 다양한 방법으로 그래프의 속성 설정하기
- 마커는 각 데이터를 표시한 지점이고, 기본적으로 작은 동그라미 모양이지만 다음의 모양으로 변경 가능

마커 지정
















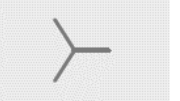




















- 그림 밑에 있는 '마커 기호'로 마커 지정한다.

											
'.'	'o'	's'	'p'	'X'	'*'	'p'	'D'	'<'	'>'	'^'	'v'
											
'1'	'2'	'3'	'4'	'+'	'x'	' '	'_'	4	5	6	7
											
'\$♠\$'	'\$♣\$'	'\$♥\$'	'\$♦\$'	'\$→\$'	'\$←\$'	'\$↑\$'	'\$↓\$'	'\$◐\$'	'\$◑\$'	'\$◒\$'	'\$◓\$'

마커 지정

• 파이플롯 마커 기호를 바꿔 출력하기

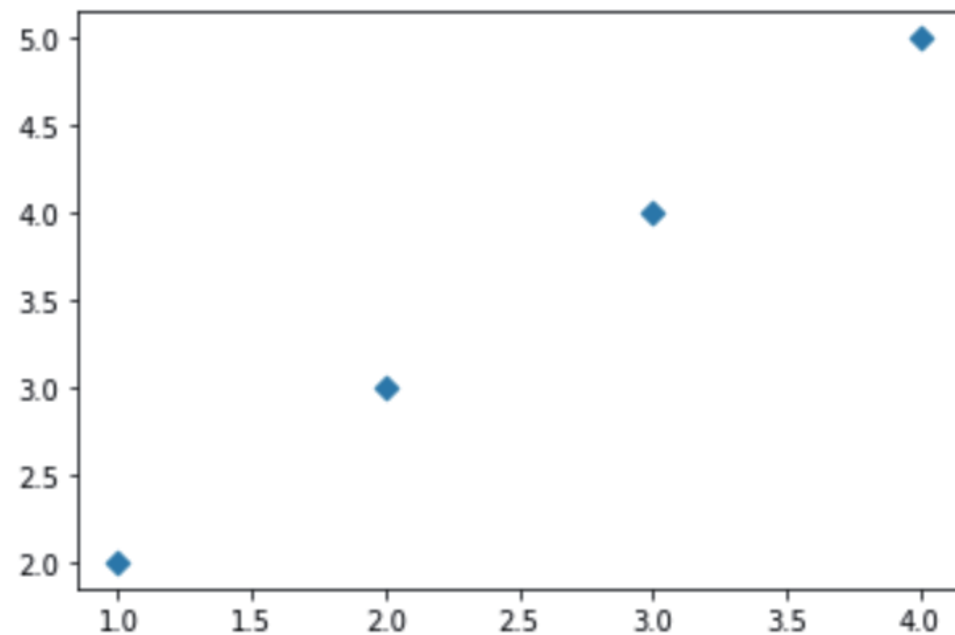
파이플롯 객체.그래프명(..., marker='마커 기호')

											
'.'	'o'	's'	'p'	'X'	'*'	'p'	'D'	'<'	'>'	'^'	'v'
											
'1'	'2'	'3'	'4'	'+'	'x'	' '	'_'	4	5	6	7
											
'\$♠\$'	'\$♣\$'	'\$♥\$'	'\$♦\$'	'\$→\$'	'\$←\$'	'\$↑\$'	'\$↓\$'	'\$◐\$'	'\$◑\$'	'\$◒\$'	'\$◓\$'

마커 지정 • 파이플롯 마커 기호를 바꿔 출력하기



```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 x = np.arange(1,5)
4 y = np.arange(2,6)
5 plt.scatter(x,y,marker='D')
6 plt.show()
```



그래프 색상 • 그래프의 색상을 다양하게 설정하기

- 각 색상에 적합한 알파벳은 해당 색으로 설정할 때 사용하는 '색상 기호' 이다.

b	g	r	c	m	y	k	w
---	---	---	---	---	---	---	---

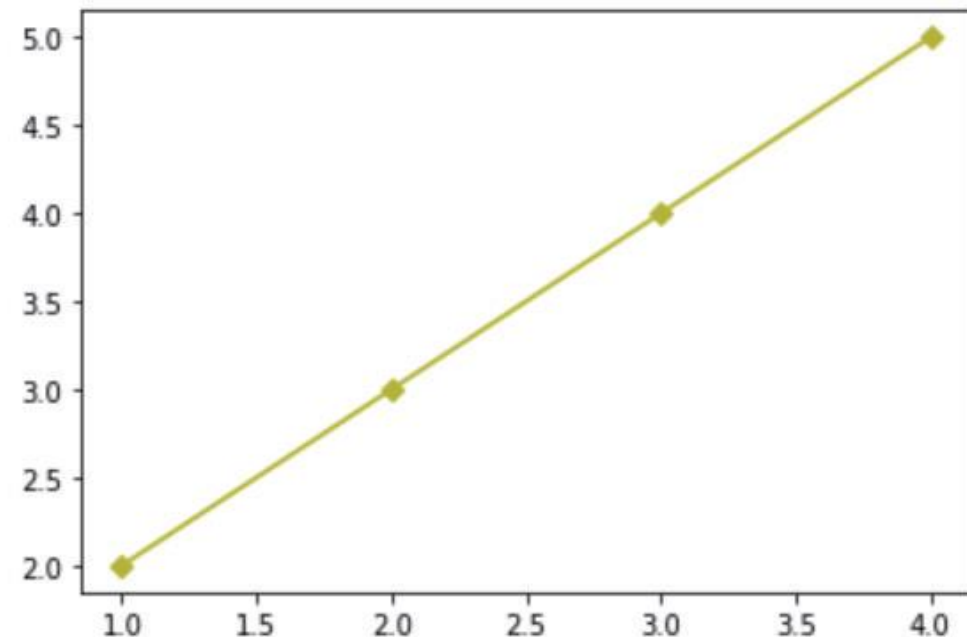
파이플롯 객체.그래프명(..., color='색상 기호')

그래프 색상 · 노란색으로 그래프 설정하기

- `plot()` 그래프는 마커를 설정하지 않으면 실선만 나타난다.



```
1 x = np.arange(1,5)
2 y = np.arange(2,6)
3 plt.plot(x,y,color='y',
4           marker='D')
5 plt.show()
```



선 종류

- 그래프의 선을 다양하게 설정하기
- 각 선 아래에 적힌 기호는 해당 선으로 설정할 때 사용하는 '선 기호'이다.

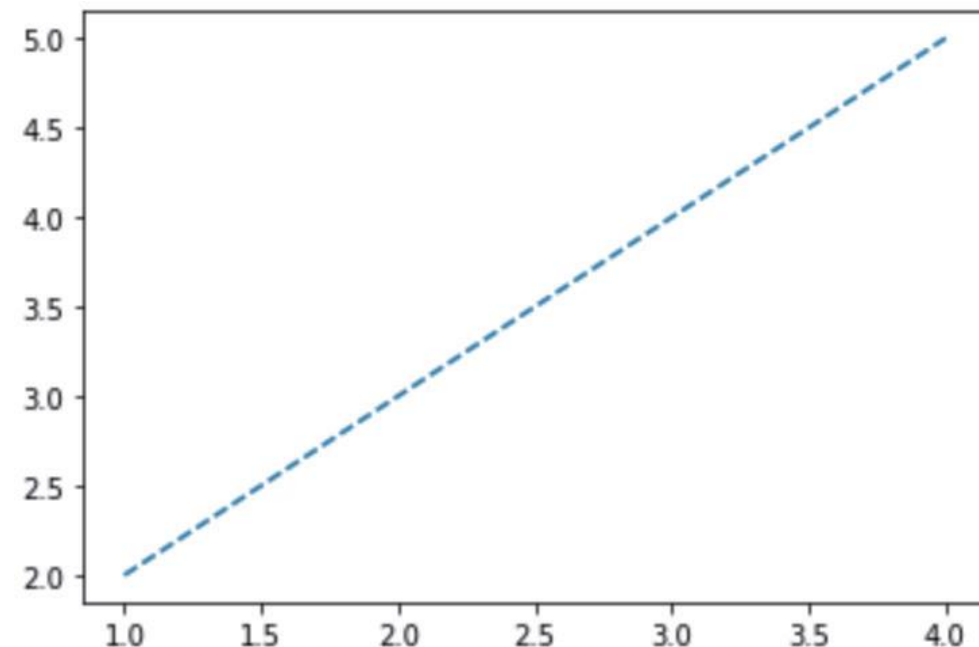


파이플롯 객체.그래프명(..., linestyle='선 기호')

선 종류 • 점선('--')으로 그래프를 그리기



```
1 x = np.arange(1,5)
2 y = np.arange(2,6)
3 plt.plot(x,y,linestyle='--')
4 plt.show()
```



축 레이블

- 맷플롯립을 이용하면 그래프의 x축과 y축의 레이블(이름)뿐만 아니라 글자 크기와 정렬할 위치도 함께 설정할 수 있다.
- 글자 크기와 여백 크기는 숫자로 설정한다.

파이플롯 객체.xlabel('레이블명', fontsize=글자 크기, labelpad=여백 크기)

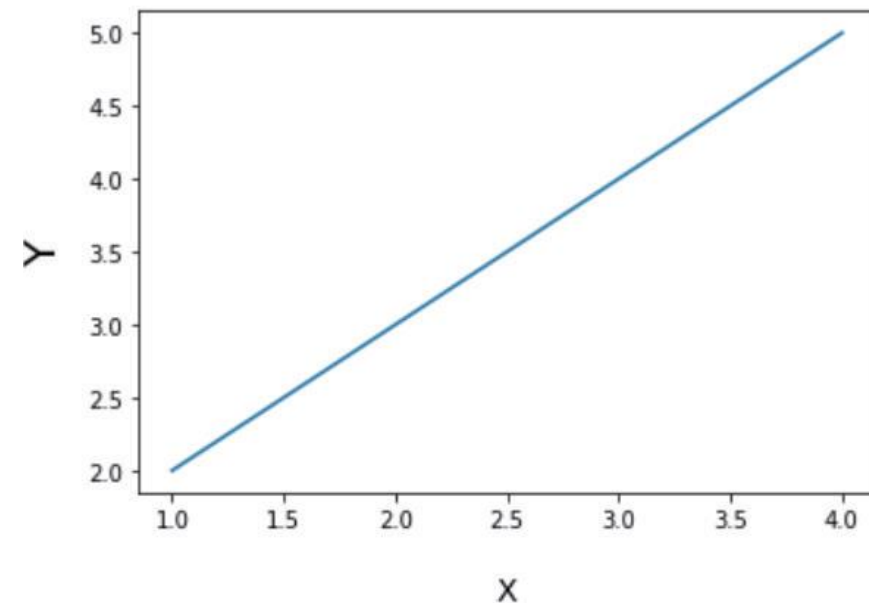
파이플롯 객체.ylabel('레이블명', fontsize=글자 크기, labelpad=여백 크기)

축 레이블

- x축과 y축의 레이블(이름)을 각각 설정하기



```
1 x = np.arange(1,5)
2 y = np.arange(2,6)
3
4 plt.plot(x,y)
5 plt.xlabel('X',fontsize=14,labelpad=20)
6 plt.ylabel('Y',fontsize=20,labelpad=10)
7 plt.show()
```



축 레이블

- x축과 y축의 레이블(이름)을 각각 설정하기

오른쪽 코드에서
글자 크기와 여백
크기를 설정하지
않고 출력해 보세
요. 어떤 차이가
있나요?



```
1 x = np.arange(1,5)
2 y = np.arange(2,6)
3
4 plt.plot(x,y)
5 plt.xlabel('X',fontsize=14,labelpad=20)
6 plt.ylabel('Y',fontsize=20,labelpad=10)
7 plt.show()
```


- 부엉이 가족 데이터프레임을 생성한 후 조건에 맞게 체중을 비교하는 그래프를 그려보기

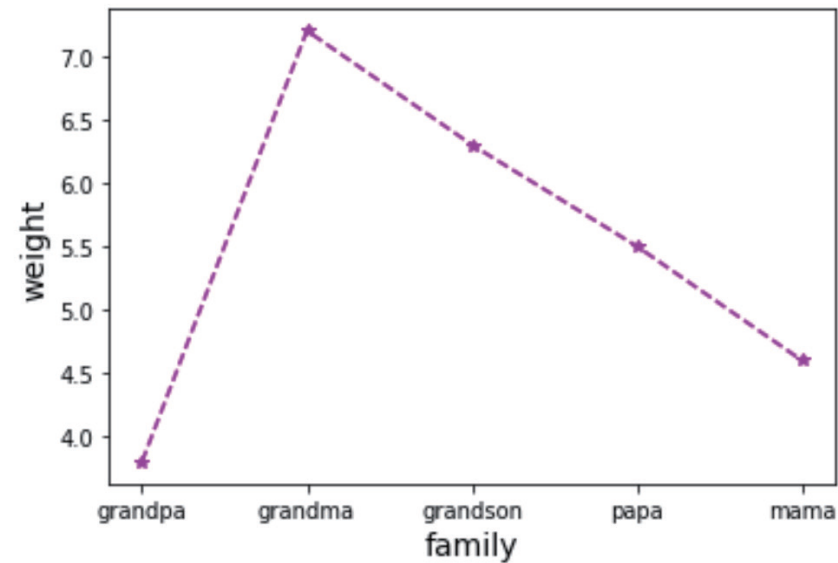
〈작성 조건〉

마커:★, 그래프 색상: 마젠타(m), 선 종류: --

〈작성 조건〉

마커:★, 그래프 색상: 마젠타(m), 선 종류: --

	age	weight
grandpa	60	3.8
grandma	70	7.2
grandson	12	6.3
papa	45	5.5
mama	42	4.6



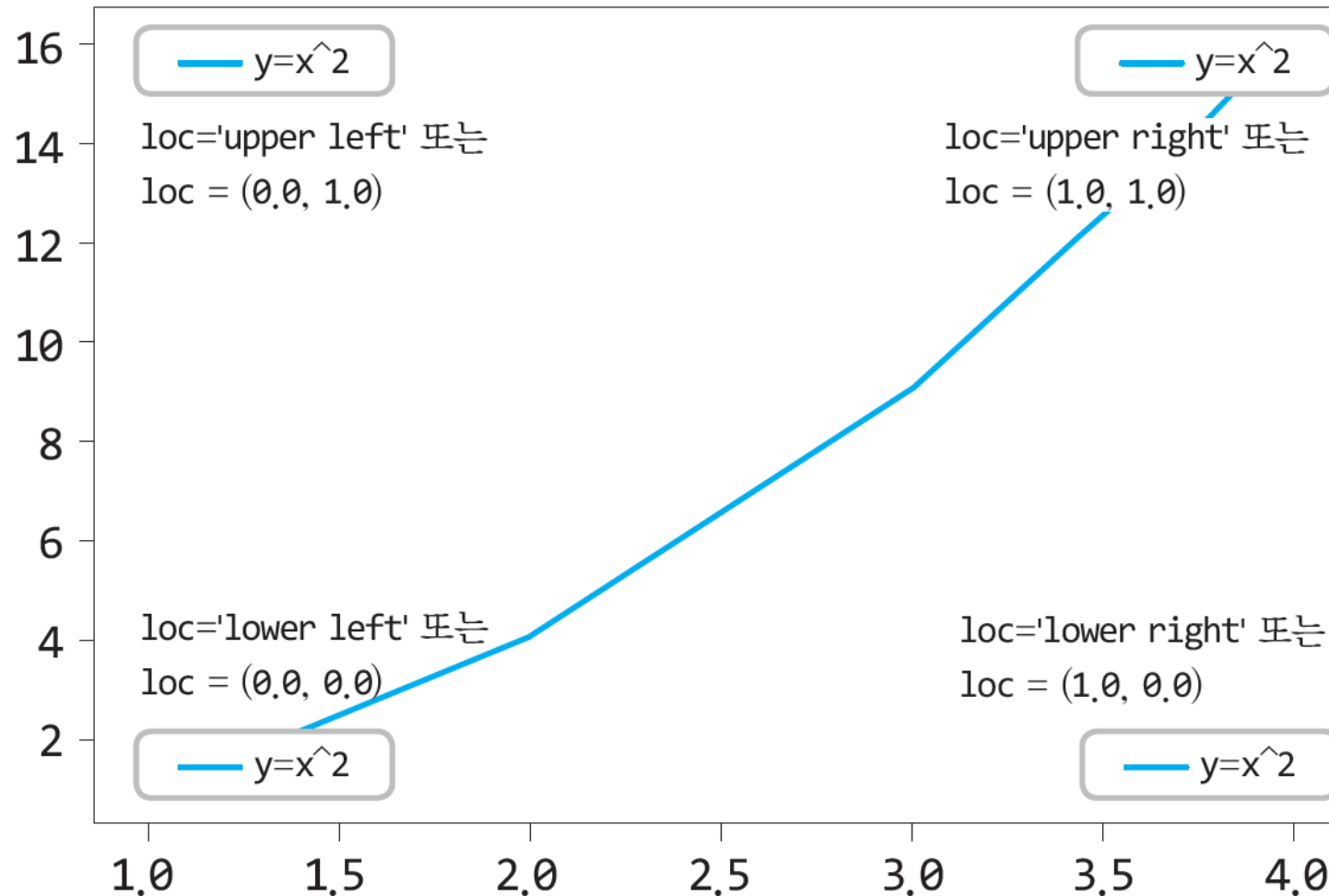
범례

- 범례(legend)는 그래프에서 데이터가 나타내는 것을 표시하는 영역이다.
- $Y = x^2$ 그래프를 그릴 때, 해당 그래프가 나타내는 것이 $y = x^2$ 임을 화면에 표시할 수 있으며, 표시하려는 위치 또한 loc 속성을 사용하여 설정할 수 있다.

파이플롯 객체.그래프명(..., label='범례명')

파이플롯 객체.legend(loc=(x,y) 또는 loc='위치')

범례



법례

- 법례를 표시할 위치를 바꿔가면서 선그래프 출력하기



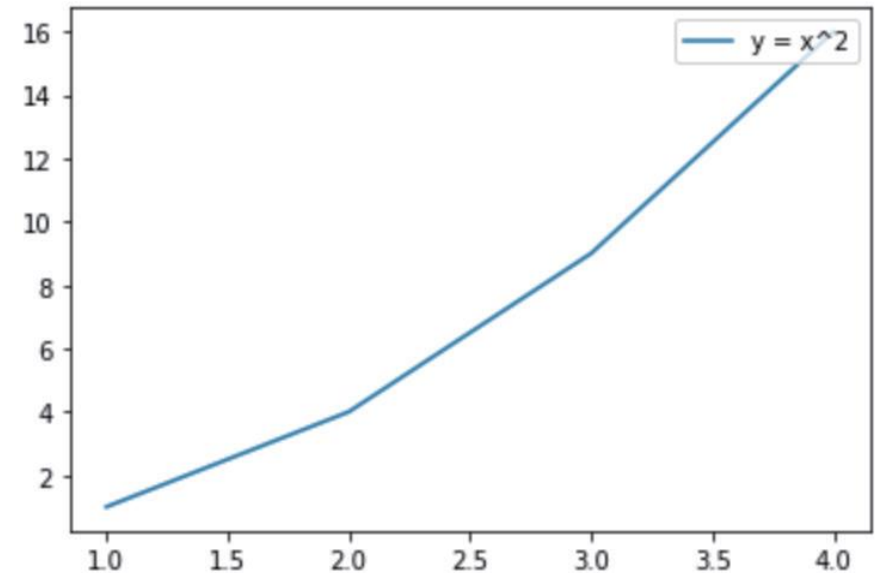
```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 x=np.arange(1,5)
4 y=x**2
5
6 plt.plot(x,y,label='y=x^2')
7 plt.legend(loc='upper right')
8 plt.show()
```

법례

• 법례를 표시할 위치를 바꿔가면서 선그래프 출력하기



```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 x=np.arange(1,5)
4 y=x**2
5
6 plt.plot(x,y,label='y=x^2')
7 plt.legend(loc='upper right')
8 plt.show()
```



축 범위

- x축과 y축이 표시되는 범위를 `xlim()`과 `ylim()` 또는 `axis()`를 사용하여 임의로 설정할 수 있다.
- 범위는 리스트 또는 튜플로 나타낼 수 있으며 축의 범위를 설정하지 않으면 자동으로 범위를 설정한다.

축 범위

- `xlim()`과 `ylim()` 활용

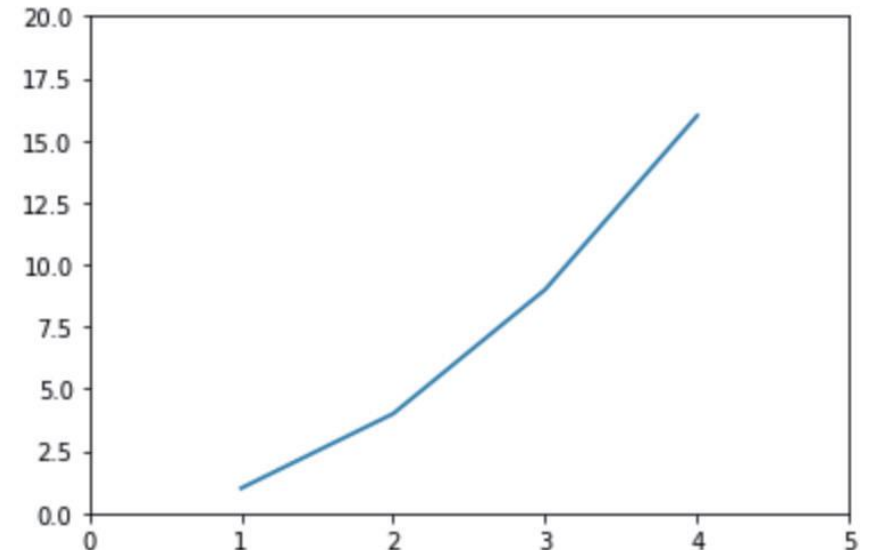
x축: 파이플롯 객체.`xlim(최솟값, 최댓값)` 또는
파이플롯 객체.`xlim([최솟값, 최댓값])` 또는
파이플롯 객체.`xlim((최솟값, 최댓값))`

y축: 파이플롯 객체.`ylim(최솟값, 최댓값)` 또는
파이플롯 객체.`ylim([최솟값, 최댓값])` 또는
파이플롯 객체.`ylim((최솟값, 최댓값))`

축 범위 • 표시할 x축과 y축의 범위를 각각 설정



```
1 x=np.arange(1,5) #x축값 범위
2 y=x**2
3
4 plt.plot(x,y)
5 plt.xlim([0,5]) #x축 표시 범위
6 plt.ylim([0,20]) #y축 표시 범위
7 plt.show()
```



축 범위

- `axis()` 활용

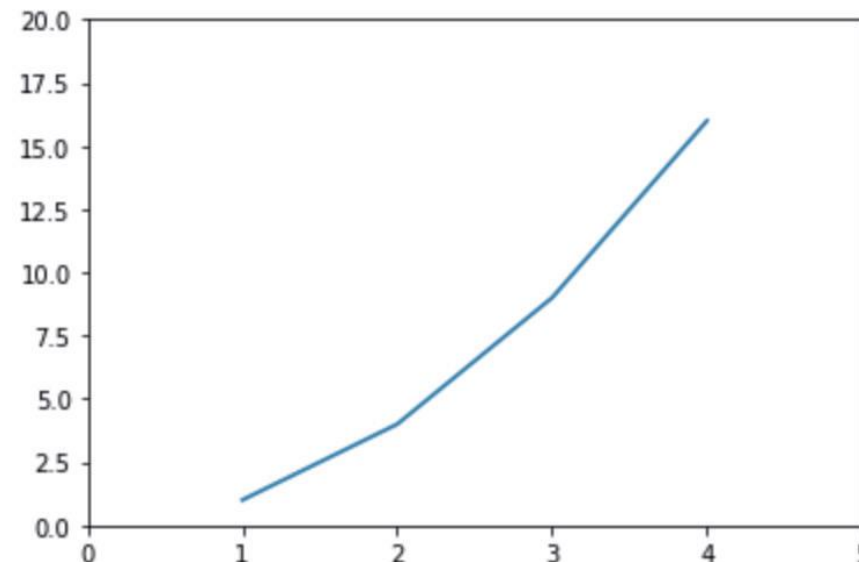
파이플롯 객체.`axis([xmin,xmax,ymin,ymax])` 또는
파이플롯 객체.`axis((xmin,xmax,ymin,ymax))`

축 범위

- 표시할 x축과 y축의 범위를 한번에 설정



```
1 x=np.arange(1,5)
2 y=x**2
3
4 plt.plot(x,y)
5 plt.axis([0,5,0,20])
6 plt.show()
```



한 그래프에 여러 개의 선 그리기

- 한 그래프 내에 여러 개의 선(또는 점)을 그려 데이터를 비교하거나 x축과 y축을 공유하는 여러 개의 선을 그릴 때는 `plt.show()`로 나타내기 전에 그리려는 그래프의 함수를 여러 번 사용한다.

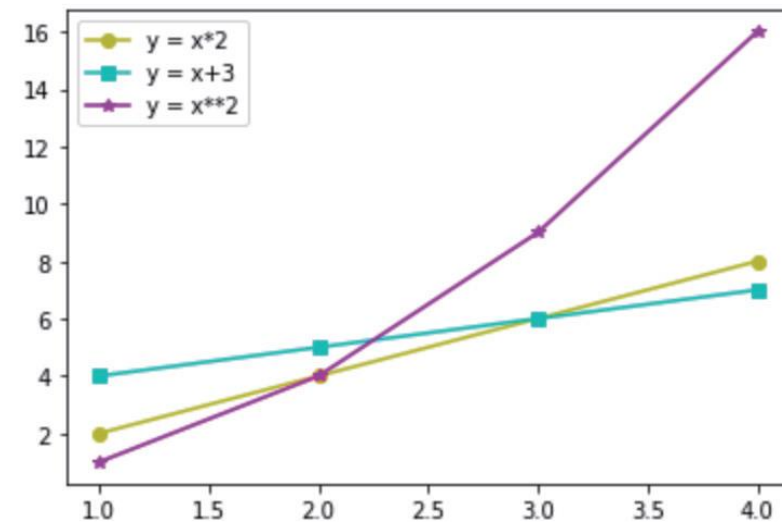
파이플롯 객체.`plot(그래프 속성, ...)`

파이플롯 객체.`plot(그래프 속성, ...)`

⋮

한 그래프에 여러 개의 선 그리기

```
▶ 1 x=np.arange(1,5)
2 #세 개 그래프 속성 설정
3 plt.plot(x,x*2,marker='o',color='y',label='y=x*2')
4 plt.plot(x,x+3,marker='s',color='c',label='y=x+3')
5 plt.plot(x,x**2,marker='*',color='m',label='y=x**2')
6 plt.legend()
7 plt.show()
```



- **속성 설정:** 마커 기호, 그래프 색상, 선 종류, 축 레이블, 범례, 축 범위 등을 설정
- **마커:** marker 옵션에 마커 기호를 설정하면 여러 모양으로 변경할 수 있다.
- **그래프 색상:** color 옵션에 색상 기호(b, g, r, c, m, y, k, w)를 설정하여 그래프의 색을 변경할 수 있다.

- **선 종류:** `linestyle` 옵션에 선 기호(`-`, `:`, `-.`, `--`)를 사용하여 선 종류를 변경할 수 있다.
- **축 레이블:** `xlabel()`, `ylabel()`를 사용하여 x축과 y축의 레이블(이름)을 표시할 수 있다.
- **범례:** `legend()`를 사용하여 그래프가 나타내는 것을 표시하는 영역의 범례명과 위치를 설정할 수 있다.

- **축 범위:** `xlim()`, `ylim()` 또는 `axis()`를 사용하여 그래프의 x축과 y축의 표시 범위를 설정할 수 있다.
- **한 그래프 내에 여러 개의 선(또는 점) 그리기:** `plt.show()`로 나타내기 전에 그리려는 그래프의 함수를 여러 번 사용한다.