동국대학교
dongguk university

데이터 처리/분석-3

엄진영

데이터 시각화

- **Matplotlib로 그래프 그리기**
  - 파이썬에서 데이터를 효과적으로 시각화하기 위해 만든 라이브러리
  - MATLAB(과학 및 공학 연산을 위한 소프트웨어)의 시각화 기능을 모델링해서 만듬
  - 간단하게 2차원 선 그래프(plot), 산점도(scatter plot), 막대 그래프(bar chart), 히스토그램(histogram), 파이 그래프(pie chart) 등을 그릴 수 있음
  - 아나콘다를 설치할 때 이미 설치 되었으므로 따로 설치할 필요 없음

import matplotlib.pyplot as plt

  - 추가 정보가 필요하다면 matplotlib 홈페이지 (http://matplotlib.org/) 를 방문

기본적인 선 그래프 그리기

- **순서가 있는 숫자 데이터를 시각화하거나 시간에 따라 변화하는 숫제 데이터를 시각화하는데 많이 사용**

plt.plot([x,] y [,fmt])

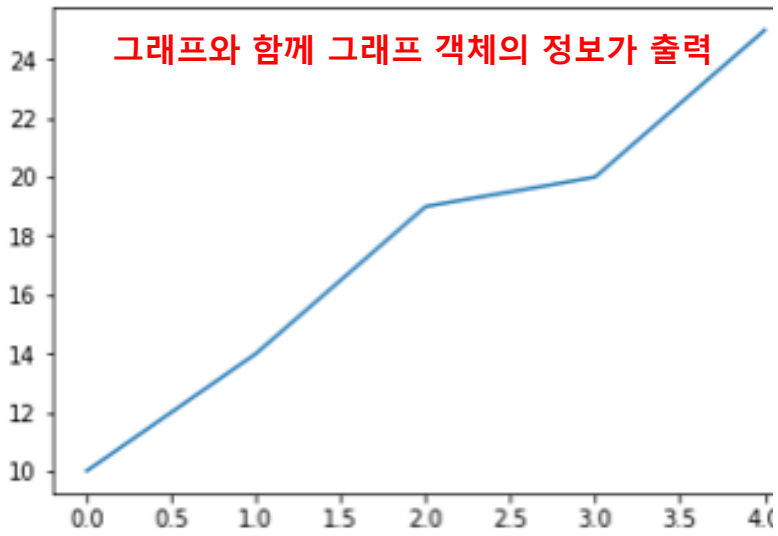
- x와 y는 각각 x축과 y축 좌표의 값을 의미
- x와 y는 각각 2차원 좌표 집합 $\{(x_0,y_0), (x_1,y_1), \ldots ,(x_n,y_n)\}$에서 x축 좌표의 요소만 모아서 만든 시퀀스 데이터 $[x_0, x_1, \ldots , x_n]$과 y축 좌표의 요소만 모아서 만든 시퀀스 데이터 $[y_0, y_1, \ldots , y_n]$ 을 의미
- x 와 y는 시퀀스의 길이가 같아야 함
- x 는 생략 가능
  - ✓ x가 없다면 0부터 y의 개수만큼 1씩 증가하는 값으로 자동할당
- fmt는 format string으로 다양한 형식으로 그래프를 그릴 수 있는 옵션
  - ✓ Fmt가 없다면 기본형식으로 그래프를 그림

기본적인 선 그래프 그리기

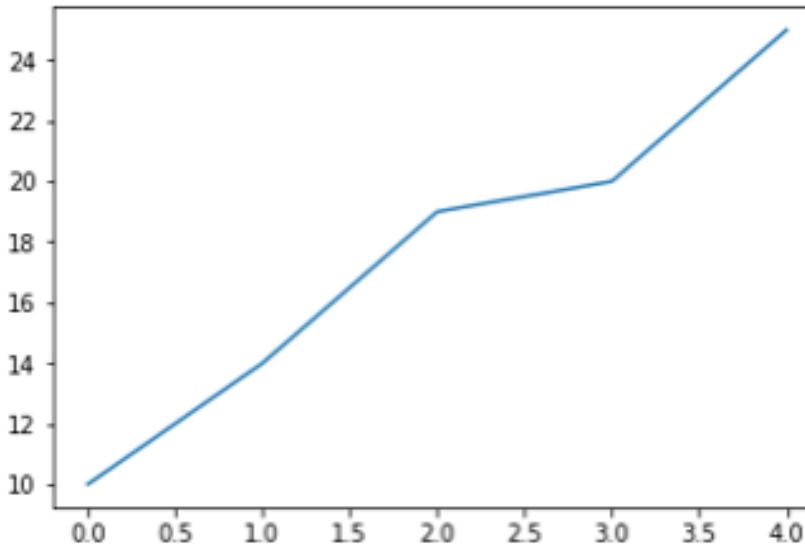
```
In [19]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt

         data1 = [10, 14, 19, 20, 25]
         plt.plot(data1)
```

```
Out[19]: [<matplotlib.lines.Line2D at 0x2ba8c7355f8>]
```

그래프와 함께 그래프 객체의 정보가 출력




```
In [20]: plt.plot(data1)
         plt.show()
```



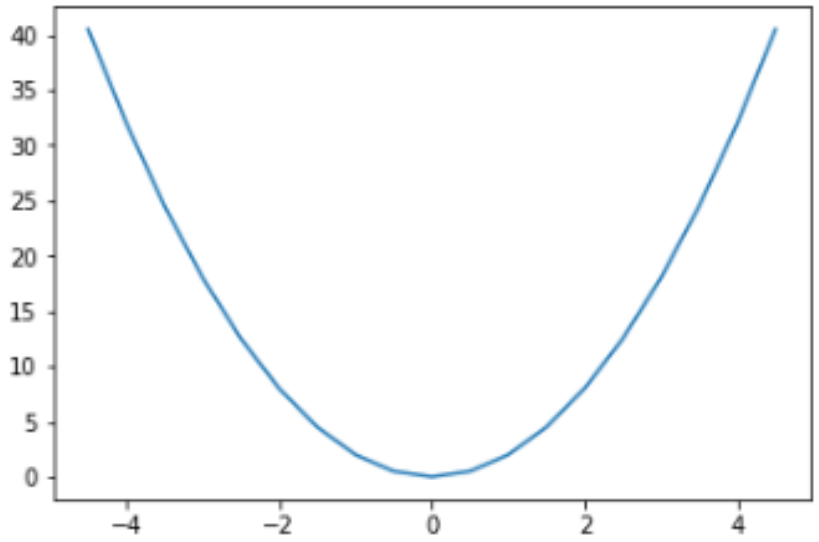
기본적인 선 그래프 그리기

```
In [21]: #배열 x 생성 범위 [-4.5,5)
         x = np.arange(-4.5,5,0.5)
         y = 2 * x**2
         [x,y]
```

```
Out[21]: [array([-4.5, -4. , -3.5, -3. , -2.5, -2. , -1.5, -1. , -0.5,  0. ,  0.5,
                  1. ,  1.5,  2. ,  2.5,  3. ,  3.5,  4. ,  4.5]),
          array([40.5, 32. , 24.5, 18. , 12.5,  8. ,  4.5,  2. ,  0.5,  0. ,  0.5,
                  2. ,  4.5,  8. , 12.5, 18. , 24.5, 32. , 40.5])]
```

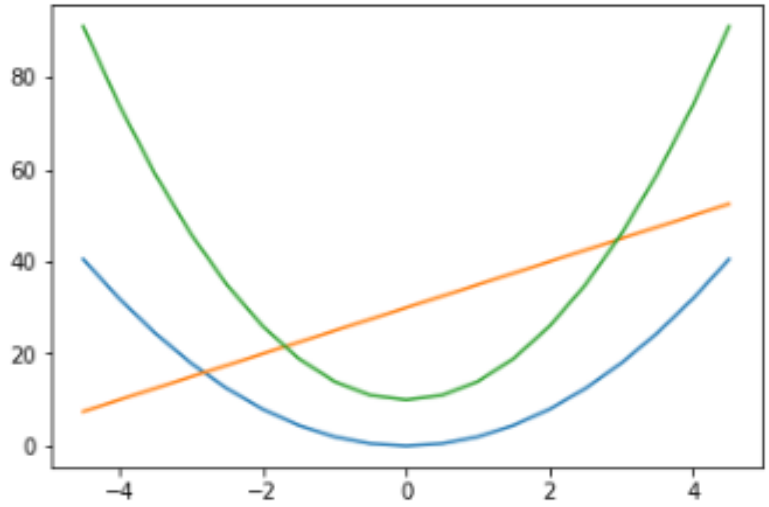
```
In [22]: plt.plot(x,y)
         plt.show()
```

X와 y 데이터를 숫자로 출력하는 것보다 그래프로 보니 데이터 특성을 한번에 파악 가능

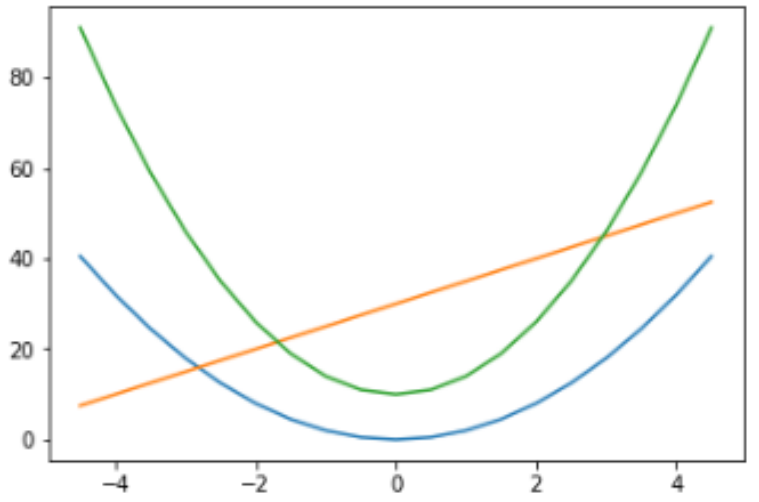


여러 그래프 그리기

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = np.arange(-4.5,5,0.5)
y1 = 2*x**2
y2 = 5*x + 30
y3 = 4*x**2 + 10

plt.plot(x,y1)
plt.plot(x,y2)
plt.plot(x,y3)
plt.show()
```



In [3]:
```
plt.plot(x,y1,x,y2,x,y3)
plt.show()
```

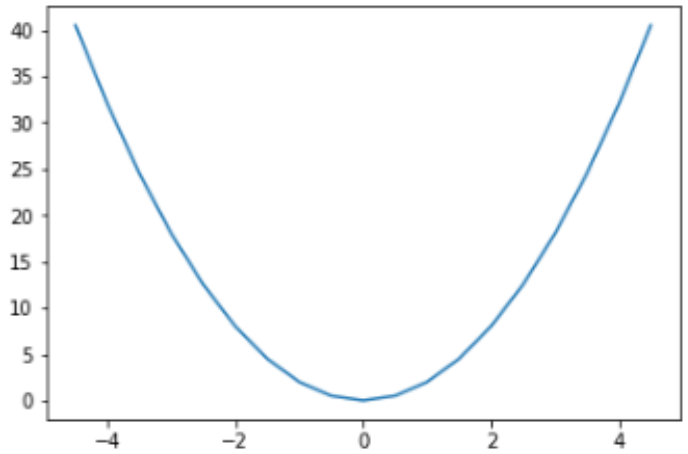


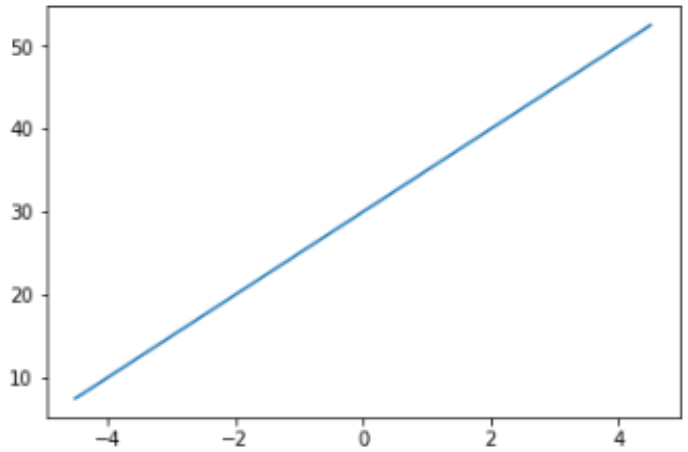
여러 그래프 그리기

- **plot.figure(n)**
  - 그래프 창의 번호를 명시적으로 지정한 후 해당 창에 그래프를 그림
  - n(정수)을 지정하면 지정된 번호로 그래프 창이 지정
  - 지정된 번호의 그래프 창이 없으면 새로 그래프 창을 생성한 후에 그래프가 그려짐
  - figure(n)로 그래프 창을 지정한 후에 '그래프_함수()'를 실행하기 전에 현재 그래프 창의 그래프를 모두 지우려면 clf()를 이용하고, 현재 그래프 창을 닫으려면 close()를 이용

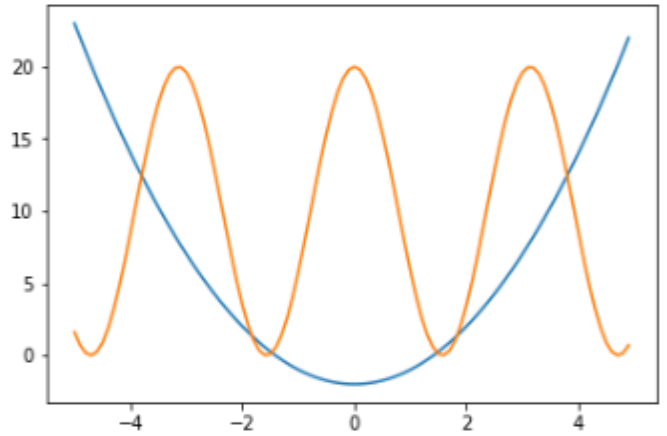
여러 그래프 그리기

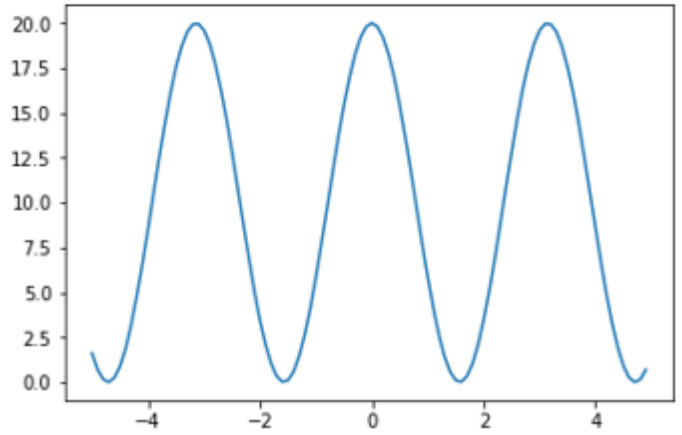
In [4]:
```
plt.plot(x,y1)

plt.figure()
plt.plot(x,y2)

plt.show()
```





In [6]:
```
x = np.arange(-5,5,0.1)
y1 = x**2 -2
y2 = 20*np.cos(x)**2

plt.figure(1)
plt.plot(x,y1)

plt.figure(2)
plt.plot(x,y2)

plt.figure(1)
plt.plot(x,y2)


plt.show()
```

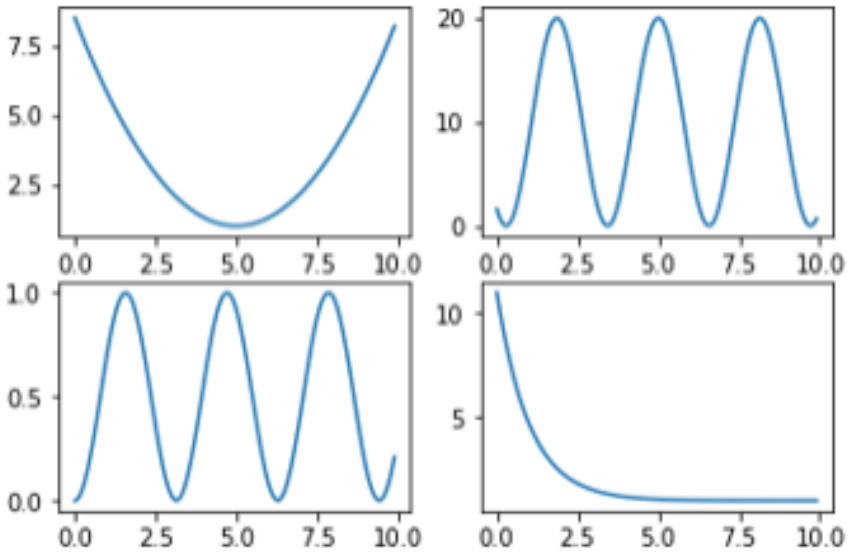




여러 그래프 그리기

- **plt.subplot(m,n,p)**
  - 하나의 그래프 창을 하위 그래프 영역으로 나누기 위해 subplot()를 사용
  - m*n 행렬로 이루어진 하위 그래프 중에서 p번 위치에 그래프가 그려지도록 지정
  - p는 왼쪽에서 오른쪽으로, 위에서 아래로 행렬의 위치를 지정

In [10]:
```
x = np.arange(0,10,0.1)
y1 = 0.3 * (x-5)**2 + 1
t2 = -1.5 * x + 3
y3 = np.sin(x)**2
y4 = 10 * np.exp(-x) + 1

plt.subplot(2,2,1)
plt.plot(x,y1)

plt.subplot(2,2,2)
plt.plot(x,y2)

plt.subplot(2,2,3)
plt.plot(x,y3)

plt.subplot(2,2,4)
plt.plot(x,y4)

plt.show()
```



그래프의 출력 범위 지정하기

- **그래프로 데이터를 분석할 때 전체 그래프 중 관심 영역만 확대해서 보고 싶을 때 좌표 범위 지정**

  plt.xlim(xmin, xmax)

  - **x 축의 좌표 범위 지정(xmin ~ xmax)**

  plt.ylim(ymin, ymax)

  - **y 축의 좌표 범위 지정(ymin ~ ymax)**

  현재 그래프의 x와 y축의 범위를 가져오려면 ?

  [xmin, xmax] = plt.xlim()

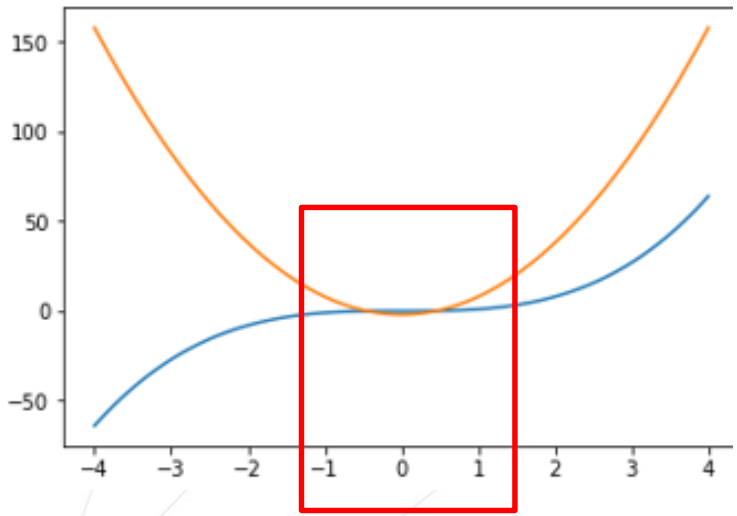
  - x축의 좌표 범위 가져오기

  [ymin, ymax] = plt.ylim()

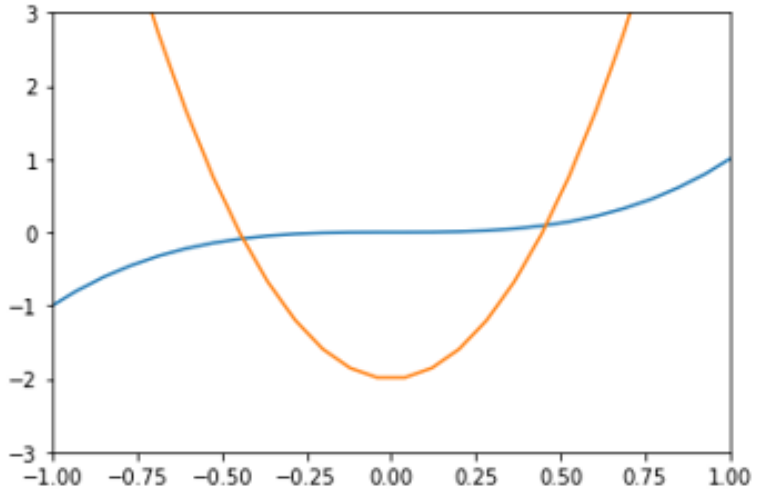
  - y축의 좌표 범위 가져오기

그래프의 출력 범위 지정하기

In [11]:
```
#[-4,4]범위에서 100개의 값 생성
x = np.linspace(-4,4,100)
y1 = x**3
y2 = 10*x**2 -2

plt.plot(x,y1,x,y2)
plt.show()
```



In [12]:
```
plt.plot(x,y1,x,y2)
plt.xlim(-1,1)
plt.ylim(-3,3)
plt.show()
```



그래프 꾸미기

- **출력 형식 지정**
  - plot()에서 fmt옵션을 이용하면 그래프의 컬러, 선의 스타일, 마커를 지정할 수 있음

fmt = '[color][line_style][marker]'

  - 각각 컬러, 선의 스타일, 마커 지정을 위한 약어(문자)

컬러 지정을 위한 약어

컬러 약어	컬러
b	파란색(blue)
g	녹색(green)
r	빨간색(red)
c	청녹색(cyan)
m	자홍색(magenta)
y	노란색(yellow)
k	검은색(black)
w	흰색(white)

선의 스타일 지정을 위한 약어

선 스타일 약어	선 스타일
-	실선(solid line)
--	파선(dashed line)
:	점선(dotted line)
-.	파선 점선 혼합선(dash-dot line)

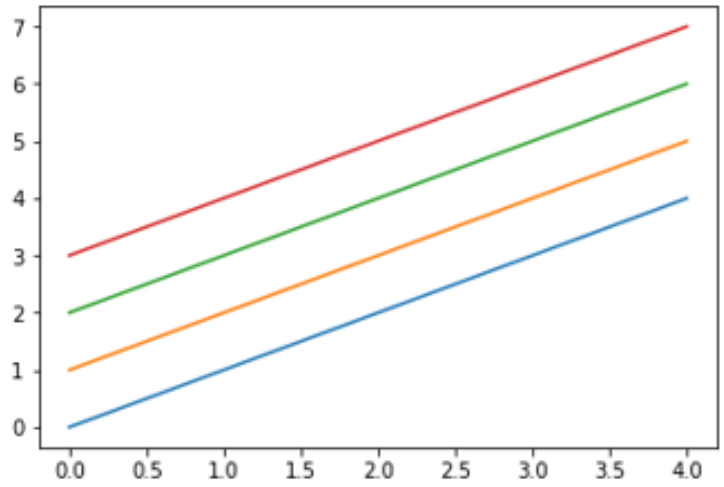
그래프 꾸미기

마커 지정을 위한 약어

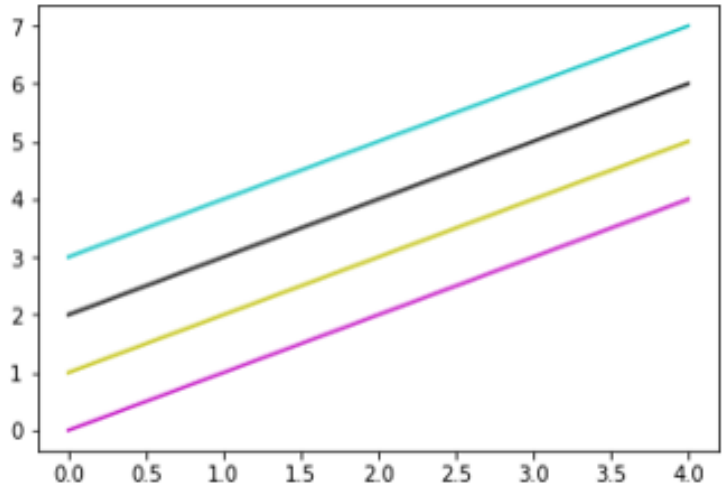
마커 약어	마커
o	원모양
^, v, <, >	삼각형 위쪽, 아래쪽, 왼쪽, 오른쪽 방향
S	사각형
P	오각형
h, H	육각형1, 육각형2
*	별모양
+	더하기
x ,X	x, 채워진 x
D, d	다이아몬드, 얇은 다이아몬드

그래프 꾸미기

In [13]:
```
x = np.arange(0,5,1)
y1 = x
y2 = x+1
y3 = x+2
y4 = x+3

plt.plot(x,y1,x,y2, x,y3, x, y4)
plt.show()
```

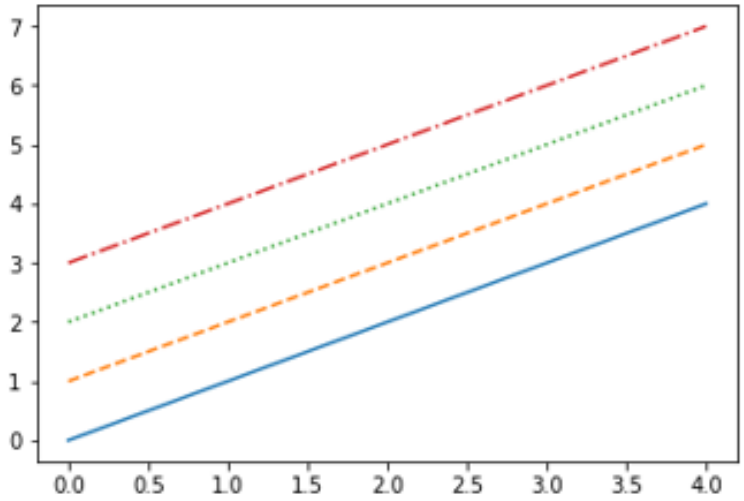


In [14]:
```
plt.plot(x,y1,'m', x,y2,'y', x,y3,'k', x, y4,'c')
plt.show()
```

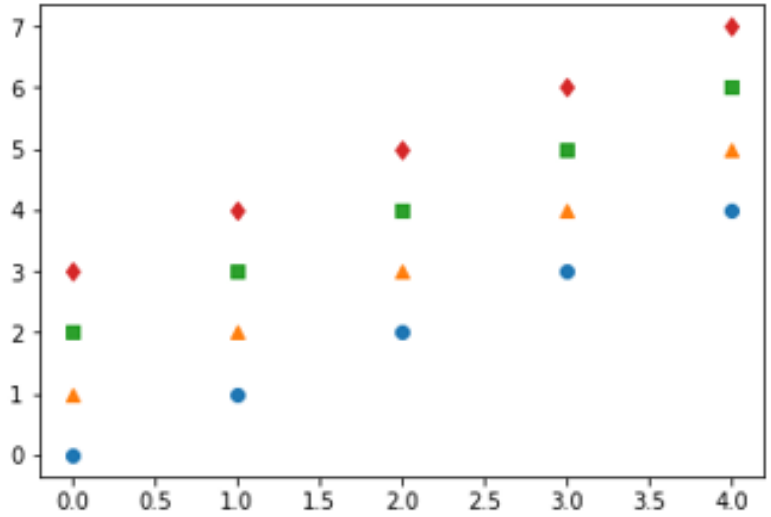


그래프 꾸미기

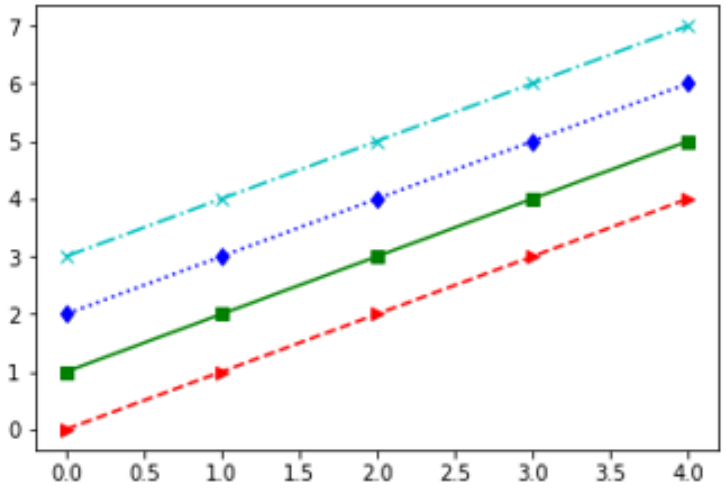
In [15]:
```
plt.plot(x,y1,'-', x,y2,'--', x,y3,':', x, y4,'-.')
plt.show()
```



In [16]:
```
plt.plot(x,y1,'o', x,y2,'^', x,y3,'s', x, y4,'d')
plt.show()
```

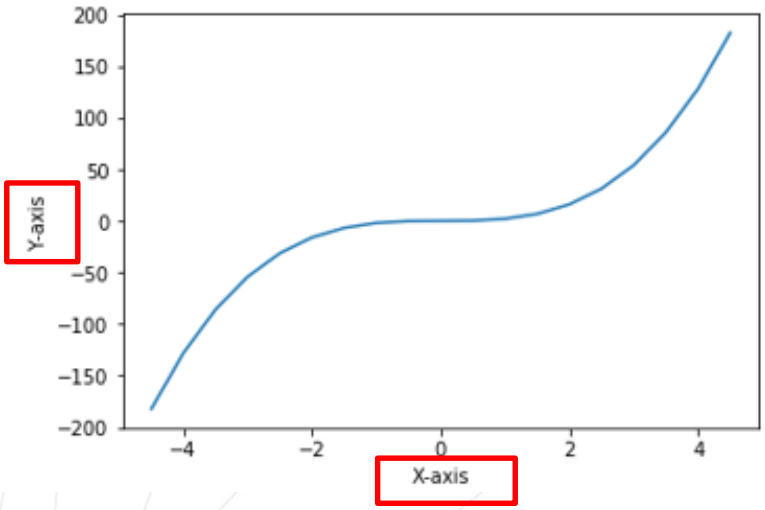


In [17]:
```
plt.plot(x,y1,'>—r', x,y2,'s-g', x,y3,'d:b', x, y4,'-.xc')
plt.show()
```

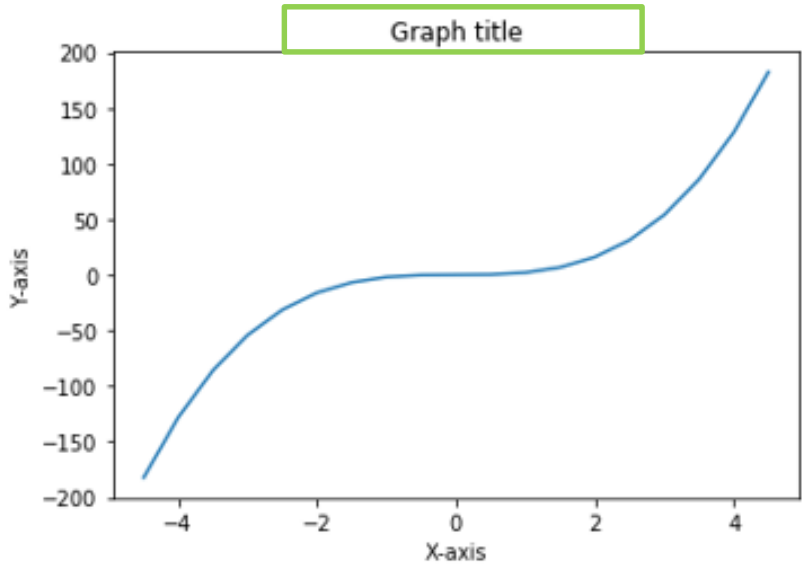


라벨, 제목, 격자, 범례, 문자열 표시

In [23]:
```
x = np.arange(-4.5, 5, 0.5)
y = 2*x**3

plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



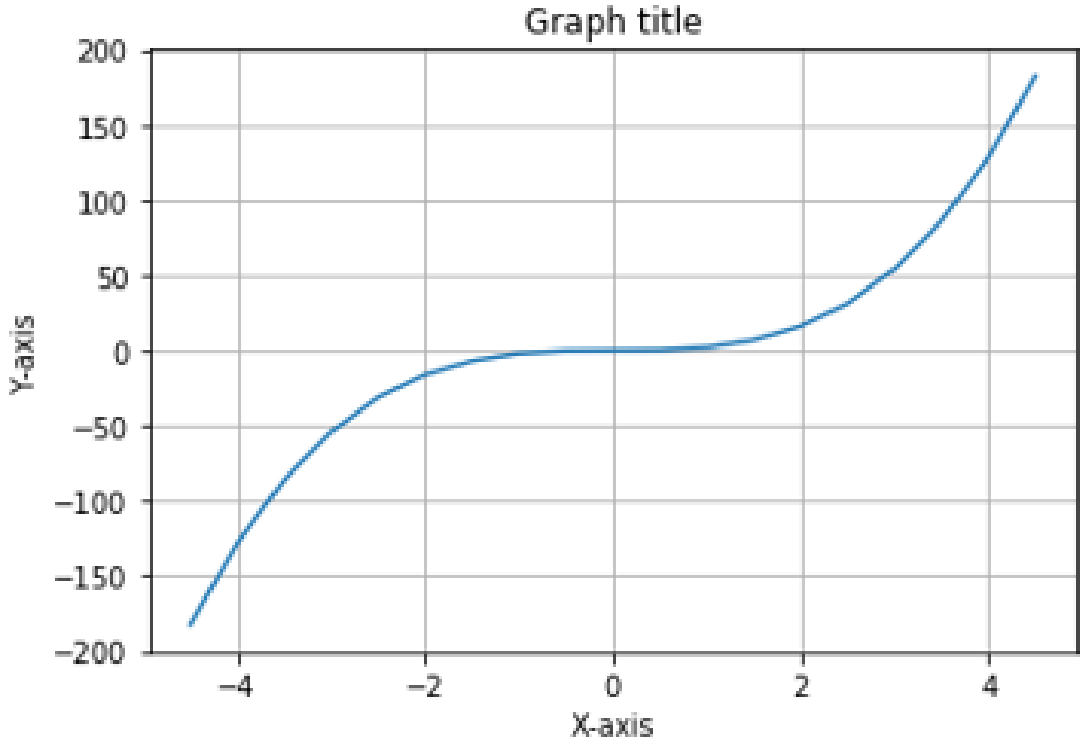
In [24]:
```
plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph title')
plt.show()
```



라벨, 제목, 격자, 범례, 문자열 표시

In [25]:
```
plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph title')
plt.grid(True) #plt.grid()도 가능
```

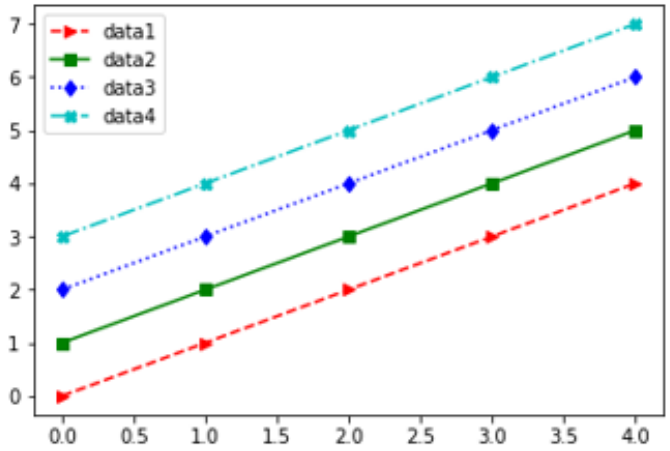
격자를 추가하려면 grid(True)혹은 grid()를 이용
격자가 있는 그래프에서 격자를 제거하고 싶으면 grid(False)를 이용
Grid()를 수행하면 show()를 수행하지 않고도 out[]에 그래프 객체의 정보를 출력하지 않고 그래프만 출력



라벨, 제목, 격자, 범례, 문자열 표시

```
In [27]: x = np.arange(0,5,1)
         y1 = x
         y2 = x+1
         y3 = x+2
         y4 = x+3
         plt.plot(x,y1,'>--r', x,y2,'s-g', x,y3,'d:b', x,y4,'-.Xc')
         plt.legend(['data1','data2', 'data3','data4'])
         plt.show()
```

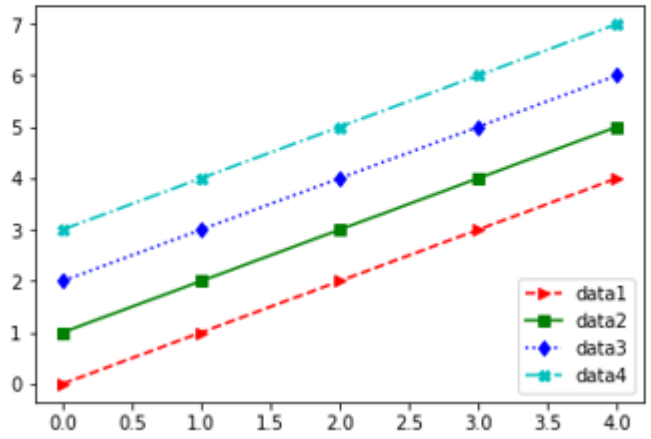
범례 추가
Loc 옵션으로 범례의 위치를 지정할 수 있음
loc='위치 문자열'을 지정하는 방식과 loc=위치코드를 입력하는 방식이 있음



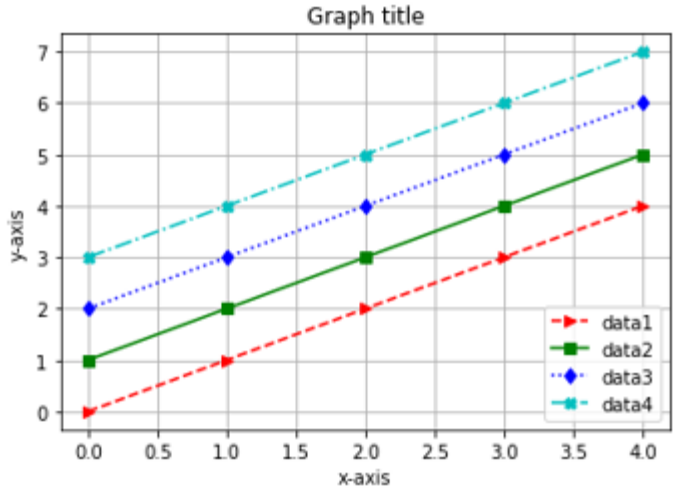
범례위치	위치문자열	위치코드
최적위치 자동선정	Best	0
상단 우측	upper right	1
상단 좌측	upper left	2
하단 좌측	lower left	3
하단 우측	lower right	4
우측	right	5
중앙 좌측	center left	6
중앙 우측	center right	7
하단 중앙	lowercenter	8
상단 중앙	upper center	9
중앙	center	10

라벨, 제목, 격자, 범례, 문자열 표시

In [7]:
```
plt.plot(x,y1,'>--r', x,y2,'s-g', x,y3,'d:b', x,y4,'-.Xc')
plt.legend(['data1','data2','data3','data4'], loc = 'lower right')
plt.show()
```



In [8]:
```
plt.plot(x,y1,'>--r', x,y2,'s-g', x,y3,'d:b', x,y4,'-.Xc')
plt.legend(['data1','data2','data3','data4'], loc = 4)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title("Graph title")
plt.grid(True)
```



라벨, 제목, 격자, 범례, 문자열 표시

- **그래프에서 한글을 표시하고 싶다면?**
  - matplotlib에서 사용하는 폰트를 한글 폰트로 지정해야 함
  - 현재 사용하고 있는 폰트 알아보기

```
In [9]: plt.rcParams['font.family']
Out[9]: ['sans-serif']
```

폰트를 변경하지 않은 기본 폰트

  - 폰트 변경하기

plt.rcParams['font.family'] = '폰트 이름'

 # 한글폰트를 지정한 후에 그래프에서 마이너스 폰트가 깨지는 문제를 해결

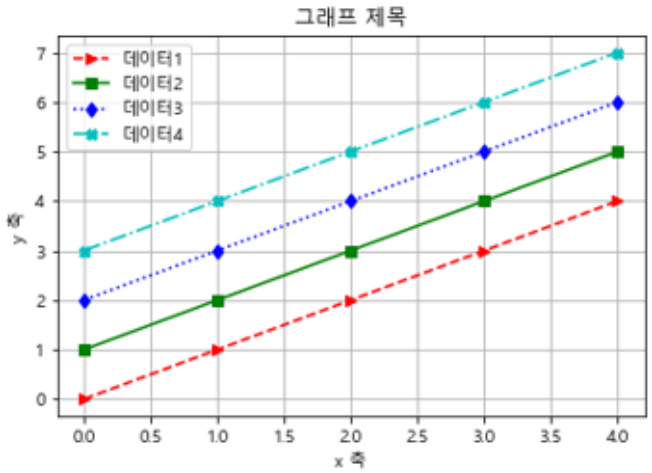
plt.rcParams['axes.unicode_minus']=False

```
In [15]: plt.rcParams['font.family'] = 'Malgun Gothic' #'맑은 고딕'로 지정
         plt.rcParams['axes.unicode_minus']=False
```

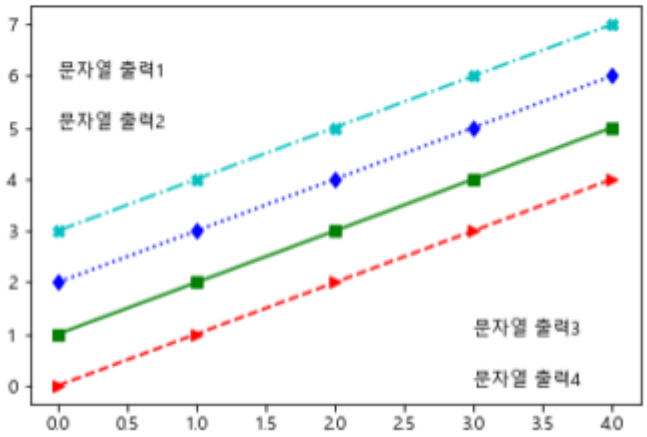
윈도우에 기본으로 설치된 맑은 고딕 지정
한글폰트로 지정한 후에는 x축과 y축 라벨,
그래프 제목, 범례를 한글로 입력 가능

라벨, 제목, 격자, 범례, 문자열 표시

In [16]:
```
plt.plot(x,y1,'>--r', x,y2,'s-g', x,y3,'d:b', x,y4,'-.Xc')
plt.legend(['데이터1','데이터2','데이터3','데이터4'], loc = 'best')
plt.xlabel('x 축')
plt.ylabel('y 축')
plt.title('그래프 제목')
plt.grid(True)
```



In [26]:
```
plt.plot(x,y1,'>--r', x,y2,'s-g', x,y3,'d:b', x,y4,'-.Xc')
plt.text(0,6,"문자열 출력1")
plt.text(0,5,"문자열 출력2")
plt.text(3,1,"문자열 출력3")
plt.text(3,0,"문자열 출력4")
plt.show()
```



그래프 창에 좌표(x,y)를 지정해 문자열 표시

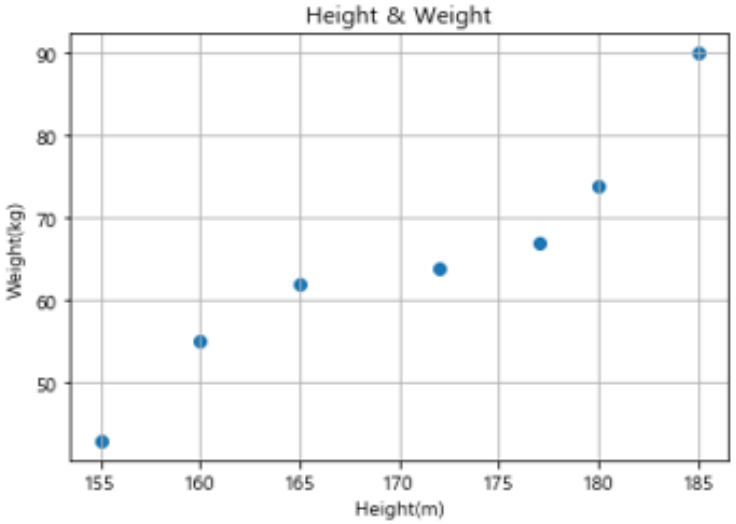
산점도

- **두 개의 요소로 이루어진 데이터 집합의 관계를 시각화 하는데 유용**
  - 키와 몸무게와의 관계, 기온과 아이스크림 판매량과의 관계, 공부시간과 시험 점수와의 관계

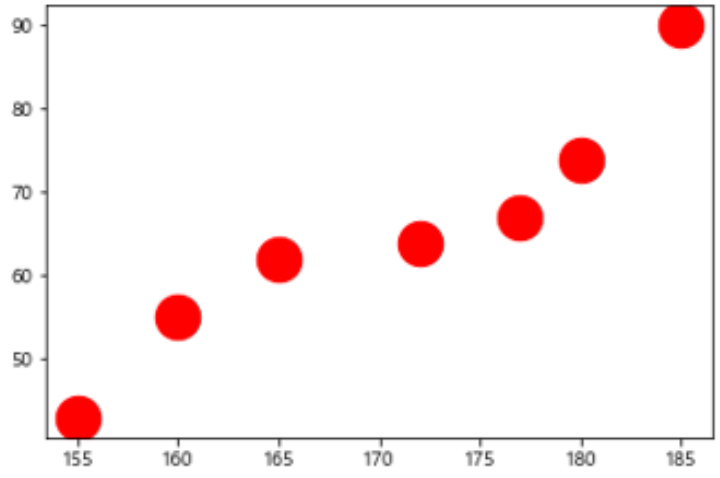
  plt.scatter(x, y [, s=size_n, c=colors, marker ='marker_string', alpha=alpha_f])
  - x축과 y축 좌표의 값
  - 옵션인 s, c, marker, alpha를 이용해 각각 마커의 크기, 컬러, 모양, 투명도를 지정할 수 있음
  - 옵션을 지정하지 않으면 기본 값 s=40, c='b', marker ='o', alpha=1로 지정
  - 옵션을 지정하고 싶으면 s에는 원하는 크기의 값을 넣으면 되고 c와 marker에는 fmt옵션의 컬러와 마커 지정 약어를 선택해서 입력하면 됨
  - s옵션에 하나의 숫자만 입력하면 모든 마커에 동일한 크기가 적용되고 배열이나 시퀀스로 입력하면 마커마다 크기를 다르게 지정할 수 있음
  - c옵션의 경오도 하나의 컬러만 입력하면 모든 마커에 동일한 컬러가 적용되고 시퀀스로 입력하면 마커마다 컬러를 다르게 지정할 수 있음
  - 투명도를 지정하는 alpha에는 [0,1]범위의 실수를 입력
    - ✓ alpha에 0을 지정하면 완전 투명, 1을 지정하면 완전 불투명

산점도

In [18]:
```
import matplotlib.pyplot as plt

height = [165,177,160,180,185,155,172] #키 데이터
weight = [62,67,55,74,90,43,64] # 몸무게 데이터

plt.scatter(height, weight)
plt.xlabel('Height(m)')
plt.ylabel('Weight(kg)')
plt.title('Height & Weight')
plt.grid(True)
```

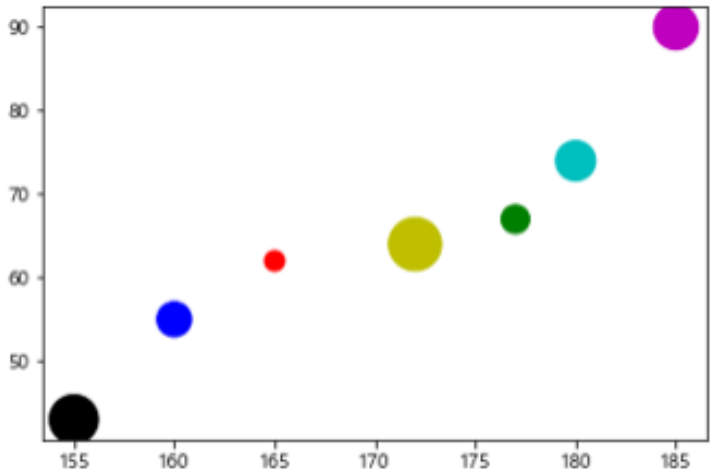


In [28]:
```
plt.scatter(height, weight, s=500, c='r')
plt.show()
```



산점도

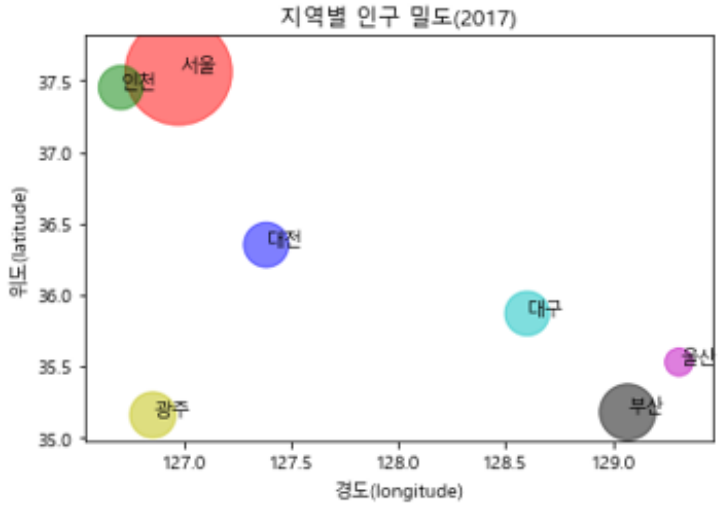
In [30]:
```
size = 100 * np.arange(1,8)
colors = ['r','g','b','c','m','k','y']

plt.scatter(height, weight, s=size, c=colors)
plt.show()
```



In [32]:
```
city = ['서울','인천','대전','대구','울산','부산','광주']

#위도(latitude)와 경도(longitude)
lat = [37.56, 37.45, 36.35, 35.87, 35.53, 35.18, 35.16]
lon = [126.97, 126.70, 127.38, 128.60, 129.31, 129.07, 126.85]

#인구밀도(명/km^2) : 2017년 통계자료
pop_den = [16154, 2751, 2839, 2790, 1099, 4454, 2995]

size = np.array(pop_den) * 0.2 #마커 크기 지정
colors = ['r','g','b','c','m','k','y'] #마커 컬러 지정

plt.scatter(lon, lat, s=size, c=colors, alpha=0.5)
plt.xlabel('경도(longitude)')
plt.ylabel('위도(latitude)')
plt.title('지역별 인구 밀도(2017)')

for x, y, name in zip(lon, lat, city):
    plt.text(x,y,name)

plt.show()
```

우리나라 주요 도시의 인구 밀도를 시각화

각 도시의 경도와 위도를 (x,y) 좌표로 지정하고 도시별 마커의 크기는 인구 밀도에 비례하도록 설정하고 마커가 위치한 곳에 도시의 이름을 표시



막대 그래프

- **값을 막대의 높이로 나타내므로 여러 항목의 수량이 많고 적음을 한눈에 알아볼 수 있음**
- **여러 항목의 데이터를 서로 비교할 때 주로 이용**

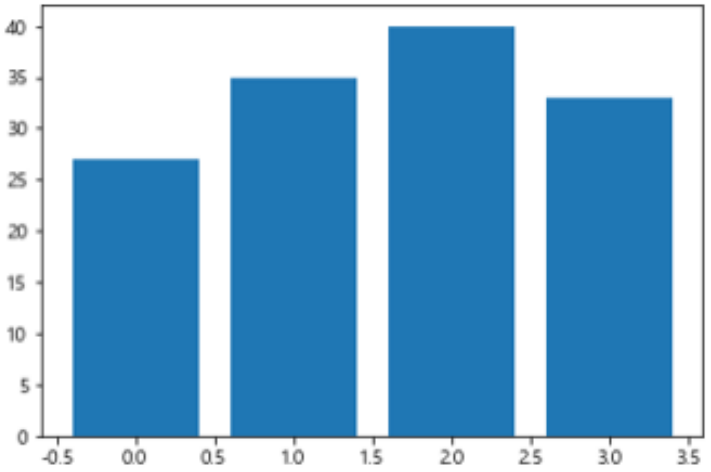
plt.bar(x, height [,width=width_f, color=colors, tick_label = tick_labels, align ='center'(기본) 혹은 'edge', label=labels])

- height는 시각화하고자 하는 막대 그래프의 데이터
- x축에 표시될 위치를 지정
  - ✓ x는 height와 길이가 일치
  - ✓ 순서만 지정하므로 0부터 시작해서 height의 길이만큼 1씩 증가하는 값
- width 옵션으로 [0,1] 사이의 실수를 지정해 막대의 폭을 조절
  - ✓ 입력하지 않으면 0.8
- color 옵션으로는 fmt 옵션의 컬러 지정 약어를 이용해 색 지정
- tick_label 옵션으로 문자열 혹은 문자열 리스트를 입력해 막대 그래프 각각의 이름 지정
  - ✓ 기본적으로 숫자로 라벨이 지정
- align은 막대 그래프의 위치 지정
  - ✓ center : 중앙(기본), edge : 한쪽으로 치우침
- label은 범례에 사용될 문자열 지정

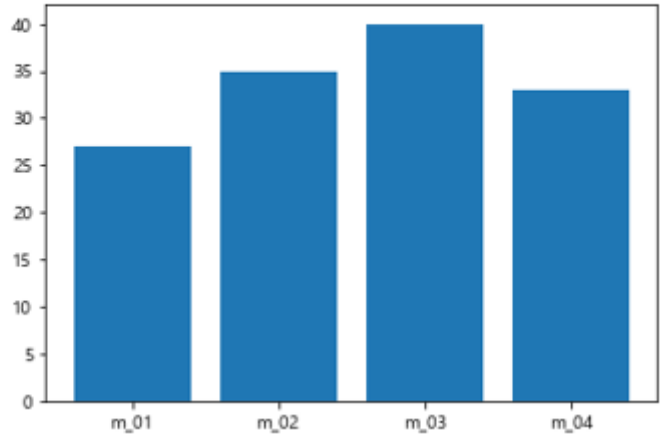
막대 그래프

회원 ID	운동 시작 전	운동 한달 후
m_01	27	30
m_02	35	38
m_03	40	42
m_04	33	37

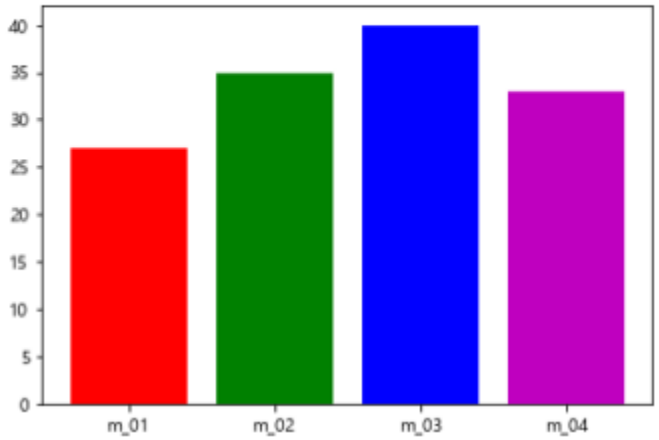
In [33]:
```
member_IDs = ['m_01','m_02','m_03','m_04']
before_ex = [27, 35, 40, 33]
after_ex = [30, 38, 42, 37]

n_data = len(member_IDs)
index = np.arange(n_data)
plt.bar(index, before_ex)
plt.show()
```



In [34]:
```
plt.bar(index, before_ex, tick_label = member_IDs)
plt.show()
```

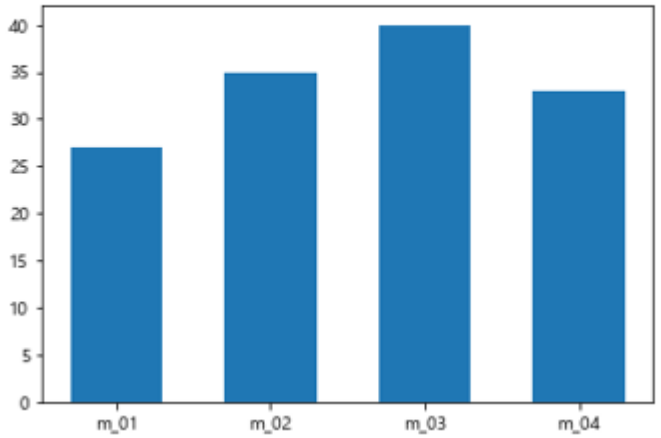


In [35]:
```
colors = ['r', 'g', 'b', 'm']
plt.bar(index, before_ex, color = colors, tick_label = member_IDs)
plt.show()
```

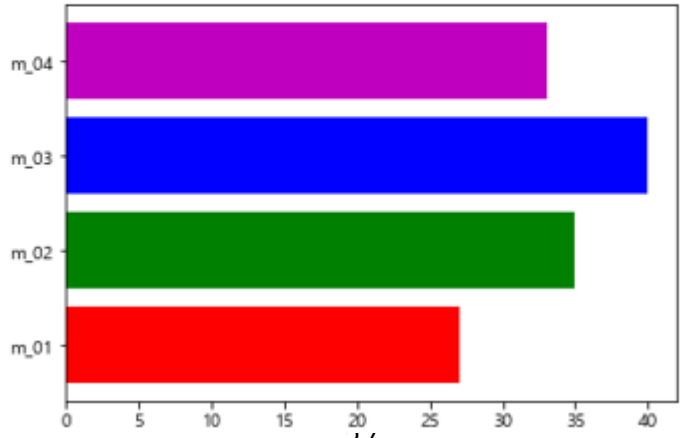


막대 그래프

In [36]:
```
plt.bar(index, before_ex, tick_label = member_IDs, width = 0.6)
plt.show()
```



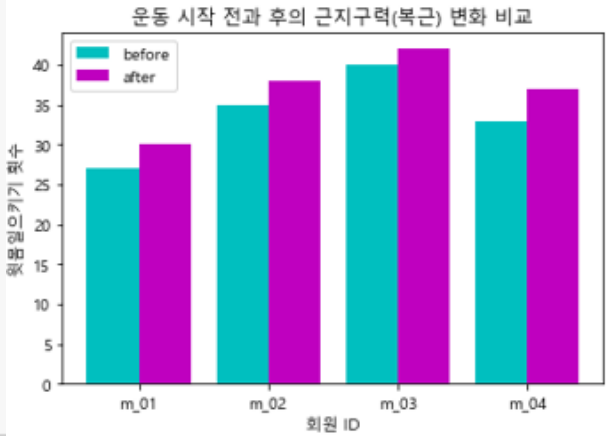
In [39]:
```
colors = ['r', 'g', 'b', 'm']
plt.barh(index, before_ex, color = colors, tick_label = member_IDs)
plt.show()
```



막대 그래프

- **두 개의 데이터에 대한 막대 그래프를 하나의 그래프 창에 비교**
  - bar()의 align옵션은 edge로 지정해 막대 그래프를 한쪽으로 치우치게...
  - Width 옵션은 두 개의 막대 그래프가 들어갈 수 있도록 여유있게 0.4로 지정
  - after_ex 데이터를 이용해 막대데이터를 그릴 때 before_ex 데이터를 이용해 그린 막대 그래프와 겹치지 않게 x좌표에서 막대 그래프의 시작 위치를 막대 그래프의 두께(0.4)만큼 오른쪽으로 이동
  - 범례로 두 데이터를 구분하기 위해 label 옵션에 각각 문자열을 지정
    - ✓ 이후에 legend()를 실행하면 label 옵션에 지정한 문자열이 범례에 표시됨
  - xticks(tick_위치, tick_label)를 이용해 x축의 tick 라벨을 붙임
  - 두 개의 데이터를 그리는 경우 tick_label옵션을 이용해 tick 라벨을 변경할 수 없음

```
member_IDs = ['m_01','m_02','m_03','m_04']
before_ex = [27, 35, 40, 33]
after_ex = [30, 38, 42, 37]
barWidth = 0.4
n_data = len(member_IDs)
index = np.arange(n_data)

plt.bar(index, before_ex, color = 'c', align = 'edge', width = barWidth, label = 'before')
plt.bar(index + barwidth, after_ex, color = 'm', align = 'edge', width = barWidth, label = 'after')

plt.xticks(index + barWidth, member_IDs)
plt.legend()
plt.xlabel('회원 ID')
plt.ylabel('윗몸일으키기 횟수')
plt.title('운동 시작 전과 후의 근지구력(복근) 변화 비교')
plt.show()
```



히스토그램

- **데이터를 정해진 간격으로 나눈 후 그 간격 안에 들어간 데이터 개수를 막대로 표시한 그래프**
  - 데이터가 어떤 분포를 갖는지 볼 때 주로 이용
- **히스토그램은 도수 분포표를 막대 그래프로 시각화한 것**
- **도수 분포표 용어**
  - 변량(variate):자료를 측정해 숫자로 표시한 것
    - ✓ 점수, 키, 몸무게, 판매량, 시간 등
  - 계급(class) : 변량을 정해진 간격으로 나눈 구간
    - ✓ 시험 점수를 60~70, 70~80, 80~90, 90~100점으로 구간으로 나눔
  - 계급의 간격(class width): 계급을 나눈 크기
    - ✓ 앞의 시험 점수를 나눈 간격은 10
  - 도수(frequency): 나눠진 계급에 속하는 변량의 수
    - ✓ 각 계급에서 발생한 수로 3,5,7,4 ...
  - 도수 분포표(frequency distribution table): 계급에 도수를 표시한 표
- **plt.hist(x, [bins = bins_n 혹은 'auto'])**
  - x는 변량 데이터, bins는 계급의 개수로 이 개수 만큼 자동으로 계급이 생성
    - ✓ bins를 입력하지 않으면 기본 10, auto이면 자동으로 bins에 값이 들어감

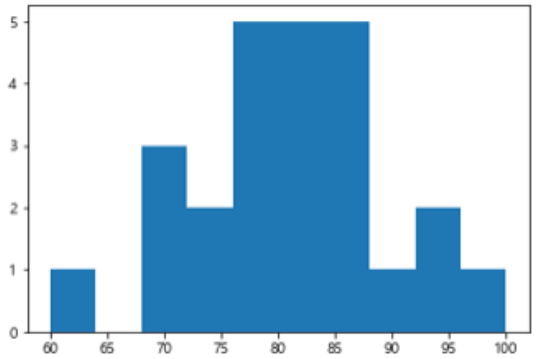
히스토그램 예제

1. **변량 생성**
   - 학생 25명의 수학 시험 결과
   - 76,82,84,83,90,86,85,92,72,71,100,87,81,76,94,78,81,60,79,69,74,87,82,68,79
2. **계급 간격 설정 및 계급 생성**
   - 변량 중 가장 작은 숫자가 60이고, 가장 큰 숫자가 100이므로 60~100의 일정한 간격(여기서는 4로 설정)으로 나눔
   - 60~64,64~68,68~72,72~76,76~80,80~84,84~88,88~92,92~96,96~100
3. **계급별 도수 확인 및 도수 분포표 만들기**
   - 각 계급에 몇 개의 변량이 들어 있는지를 확인해 계급별로 도수를 구함

계급(수학 점수)	도수
60~64	1
64~68	0
68~72	3
72~76	2
76~80	5
80~84	5
84~88	5
88~92	1
92~96	2
96~100	1

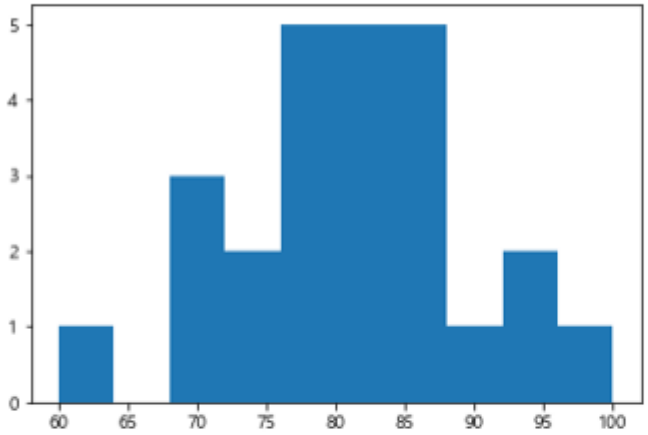
```
math = [76,82,84,83,90,86,85,92,72,71,100,87,81,76,94,78,81,60,79,69,74,87,82,68,79]
plt.hist(math)
```

```
(array([1., 0., 3., 2., 5., 5., 5., 1., 2., 1.]),
 array([ 60.,  64.,  68.,  72.,  76.,  80.,  84.,  88.,  92.,  96., 100.]),
 <a list of 10 Patch objects>)
```



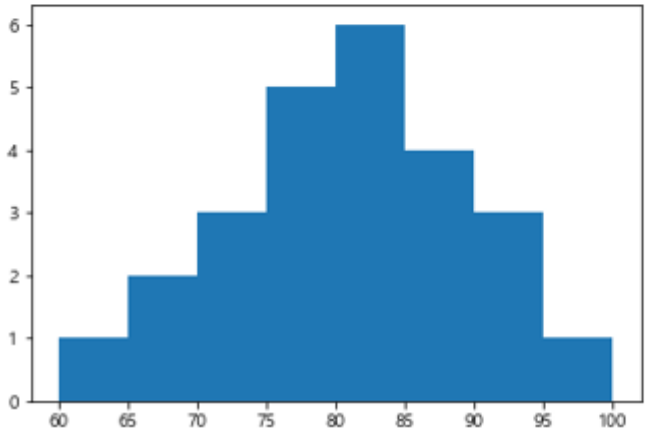
```
math = [76,82,84,83,90,86,85,92,72,71,100,87,81,76,94,78,81,60,79,69,74,87,82,68,79]
plt.hist(math)
```

```
(array([1., 0., 3., 2., 5., 5., 5., 1., 2., 1.]),
 array([ 60.,  64.,  68.,  72.,  76.,  80.,  84.,  88.,  92.,  96., 100.]),
 <a list of 10 Patch objects>)
```



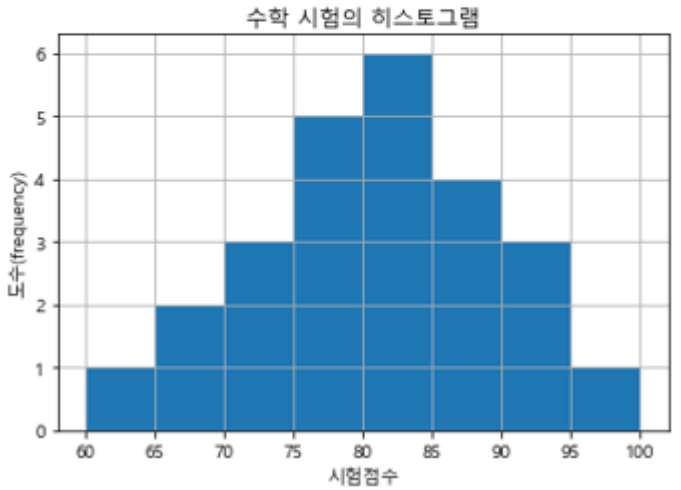
계급(수학 점수)	도수
60~64	1
64~68	0
68~72	3
72~76	2
76~80	5
80~84	5
84~88	5
88~92	1
92~96	2
96~100	1

```
plt.hist(math, bins=8)
plt.show()
```



계급(수학 점수)	도수
60~65	1
65~70	2
70~75	3
75~80	5
80~85	6
85~90	4
90~95	3
95~100	1

```
plt.hist(math, bins=8)
plt.xlabel('시험점수')
plt.ylabel('도수(frequency)')
plt.title('수학 시험의 히스토그램')
plt.grid()
plt.show()
```



파이 그래프

- **원 안에 데이터의 각 항목이 차지하는 비율만큼 부채꼴의 크기를 갖는 영역으로 이뤄진 그래프**
- **파이 그래프에서 부채꼴 부분의 크기는 각 항목 크기에 비례**
- **전체 데이터에서 각 항목이 차지한 비율을 비교할 때 많이 이용**
  - **plt.pie(x, [labels=label_seq, autopct='비율 표시 형식', shadow=False(기본) 혹은 True, explode=explode_seq, counterclock=True(기본) 혹은 False, startangle=각도(기본 0)**
  - X는 배열 혹은 시퀀스 형태의 데이터
  - Pie()는 x를 입력하면 x의 각 요소가 전체에서 차지하는 비율을 계산하고 그 비율에 맞게 부채꼴 부분의 크기를 결정
  - Labels : ㅌ 데이터 항목의 수와 같은 문자열 시퀀스를 지정해 파이그래프의 각 부채꼴 부분에 문자열을 표시
  - Autopct: 각 부채꼴 부분에 항목의 비율이 표시되는 숫자의 형식
    - ✓ '%0.1f'가 입력되면 소수점 첫째 자리까지 표시, '%0.0f'가 입력되면 정수만 표시, '%0.1f%%'가 입력되면 소수점 첫째 자리까지 표시하고 '%'를 추가
  - Shadow : 그림자 효과를 지정하는 것
  - Explode: 부채꼴 부분이 원에서 돌출되는 효과를 주어 특정 부채꼴 부분을 강조할 때 이용
  - counterclock: x 데이터에서 부채꼴 부분이 그려지는 순서가 반시계방향(True, 기본)인지 시계방향인지(False) 지정
  - Startangle: 제일 처음 부채꼴 부분이 그려지는 각도로 x축을 중심으로 반시계방향으로 증가(기본값은 0)

파이 그래프

```
fruit = ['사과','바나나','딸기','오렌지','포도']
result = [7, 6, 3, 2, 2]
plt.pie(result)
plt.show()
```

```
plt.figure(figsize=(5,5))
plt.pie(result)
plt.show()
```

파이 그래프를 그리기 전에 그래프 크기(너비와 높이)를 지정, figsize=(너비, 높이)이며 단위는 inch

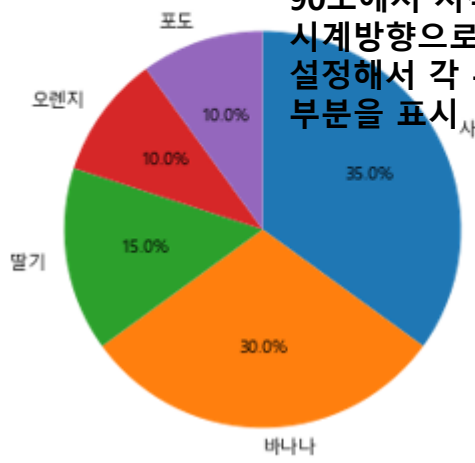
```
plt.figure(figsize=(5,5))
plt.pie(result, labels=fruit, autopct='%.1f%%')
plt.show()
```

라벨 추가

각 부채꼴 부분은 x축 기준 각도 0도를 시작으로 반시계방향으로 각 부채꼴이 그려짐

```
plt.figure(figsize=(5,5))
plt.pie(result, labels=fruit, autopct='%.1f%%',
        startangle=90, counterclock=False)
plt.show()
```

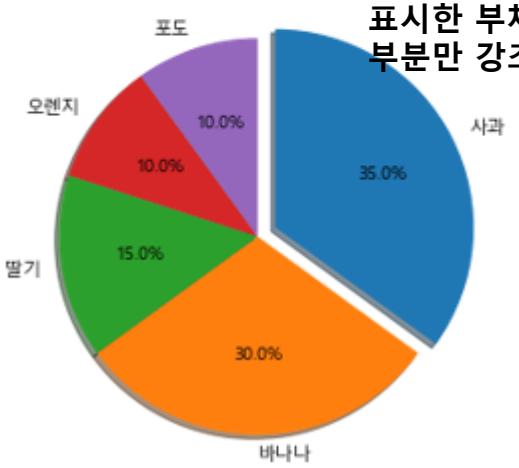
x축 기준 각도 90도에서 시작해서 시계방향으로 설정해서 각 부채꼴 부분을 표시



파이 그래프

```
explode_value = (0.1,0,0,0,0)

plt.figure(figsize=(5,5))
plt.pie(result, labels=fruit, autopct='%.1f%%',
        startangle=90, counterclock=False,
        explode=explode_value, shadow=True)
plt.show()
```

그림자를 추가하고,
특정요소(사과)를
표시한 부채꼴
부분만 강조



그래프 저장하기

- **plt.savefig(file_name, [,dpi=dpi_n(기본은 72)])**
  - File_name은 저장하고자 하는 이미지 파일 이름
    - ✓ 파일 이름은 폴더와 경로를 포함할 수 있음
    - ✓ 저장할 수 있는 이미지 파일의 확장자에는 'eps, ipeg, jpg, pdf, pgf, png, ps, raw, rgba, svg, svgz, tif, tiff'가 있음
  - 옵션 dpi(dots per inch)에는 숫자가 들어감
    - ✓ Dpi에 대입되는 숫자가 클수록 해상도가 높아져서 세밀한 그림이 그려지지만 파일의 크기도 커지므로 적당한 숫자를 설정
    - ✓ 기본은 72로 1인치 안에 72개의 점을찍는 해상도로 그래프를 저장한다
- **그래프의 크기 확인 방법**

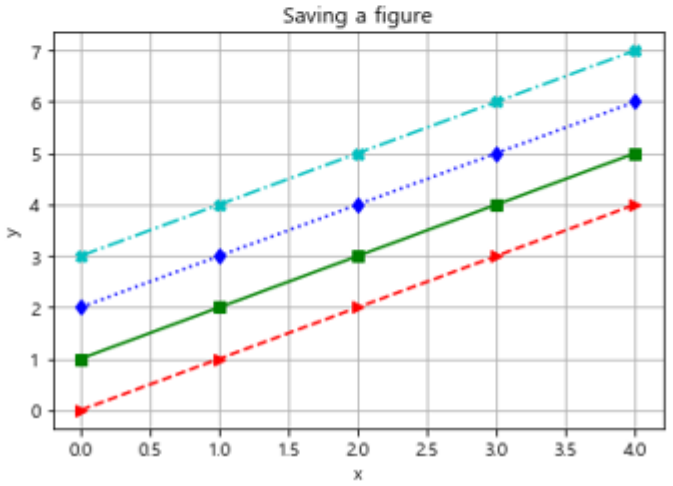
```
In [53]: import matplotlib as mpl
         mpl.rcParams['figure.figsize']

Out[53]: [6.0, 4.0]
```

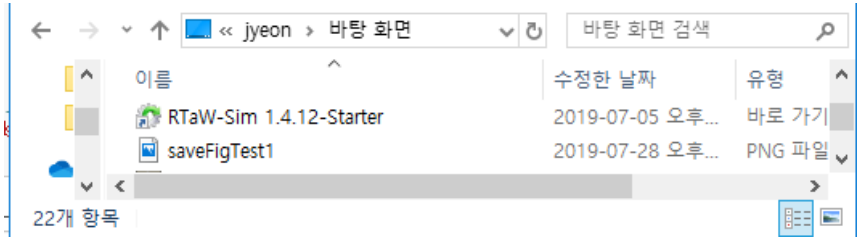
  - 기본적으로 생성되는 크기가 너비 6.0인치, 높이 4.0인치라는 것을 의미
  - 그래프의 크기를 변경하지 않으면 기본으로 이 값을 이용해 그래프를 그리고 이미지 파일을 생성
  - 그래프의 크기는 figure(figsize = (w,h))를 이용해 변경가능
- **dpi 값을 확인 방법**

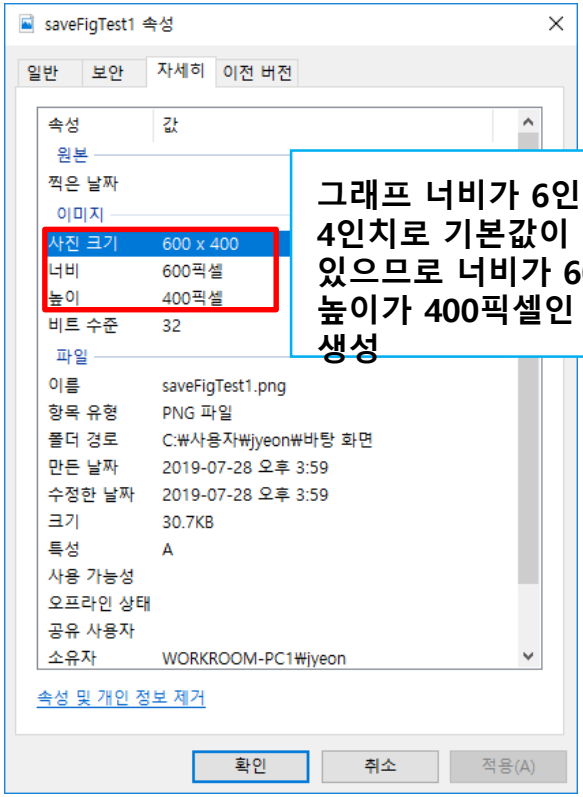
```
In [55]: mpl.rcParams['figure.dpi']

Out[55]: 72.0
```

그래프 저장하기

```
In [56]: x = np.arange(0,5,1)
         y1 = x
         y2 = x+1
         y3 = x+2
         y4 = x+3

         plt.plot(x,y1,'>—r', x,y2,'s-g', x,y3,'d:b', x,y4,'-.Xc')
         plt.grid(True)
         plt.xlabel('x')
         plt.ylabel('y')
         plt.title('Saving a figure')

         #그래프를 이미지 파일로 저장. dpi는 100dmfh tjfwjd
         plt.savefig('C:/Users/jyeon/Desktop/saveFigTest1.png', dpi=100)
         plt.show()
```





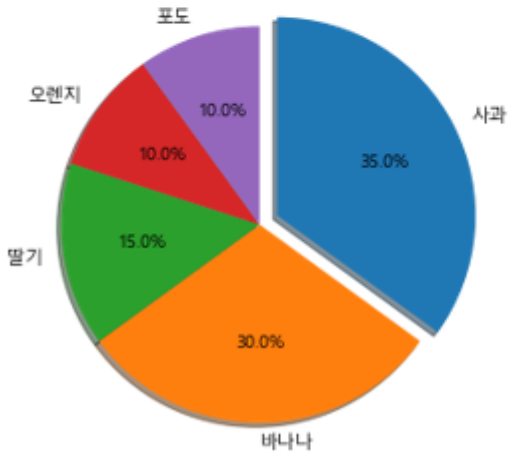


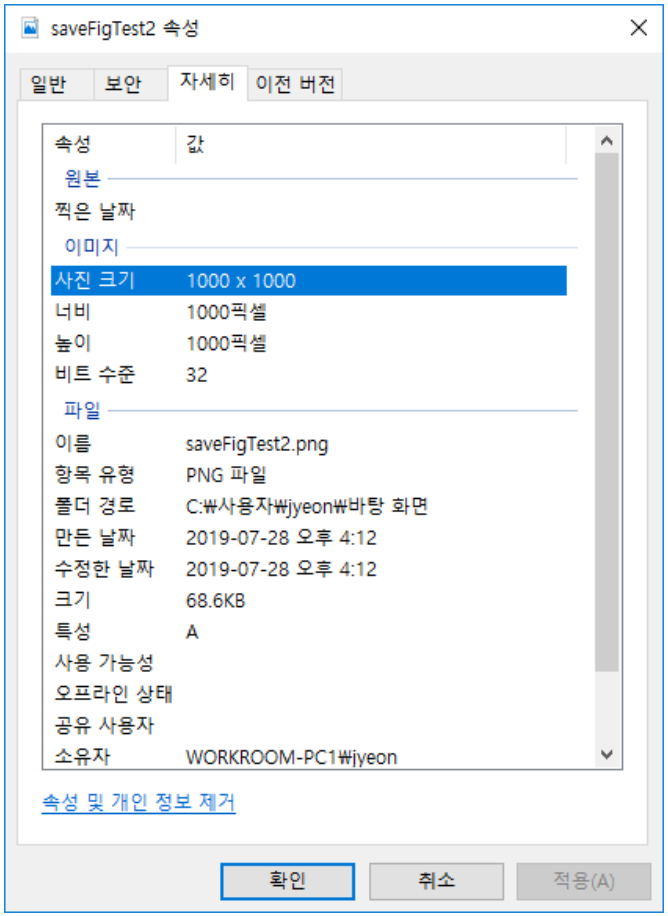
그래프 너비가 6인치이고 높이가 4인치로 기본값이 지정되어 있으므로 너비가 600픽셀, 높이가 400픽셀인 이미지 파일 생성

그래프 저장하기

In [57]:
```
fruit = ['사과','바나나','딸기','오렌지','포도']
result = [7, 6, 3, 2, 2]
explode_value = (0.1,0,0,0,0)

plt.figure(figsize=(5,5))
plt.pie(result, labels=fruit, autopct='%.1f%%',
        startangle=90, counterclock=False,
        explode=explode_value, shadow=True)

#그래프를 이미지 파일로 저장. dpi는 100dmfh tjfwjd
plt.savefig('C:/Users/jyeon/Desktop/saveFigTest2.png', dpi=200)
plt.show()
```

너비와 높이를 모두 5인치로 지정
dpi(해상도)는 인치당 200이므로
저장된 그림파일(saveFigTest2.png)은
1000*1000dml 이미지 크기를 가짐





Pandas로 그래프 그리기

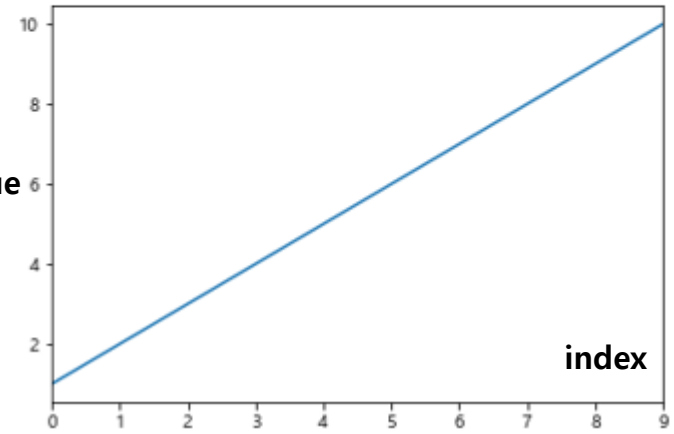
Series_data.plot([kind='graph_kind'][,option])

DataFrame_data.plot([x=label 혹은 position, y=label 혹은 position,][kind='graph_kind'][,option]

Kind 옵션	의미
line	선 그래프(기본)
Scatter	산점도(DataFrame 데이터만 가능)
Bar	수직 바 그래프
Barh	수평 바 그래프
Hist	히스토그램
Pie	파이 그래프

Pandas의 선 그래프

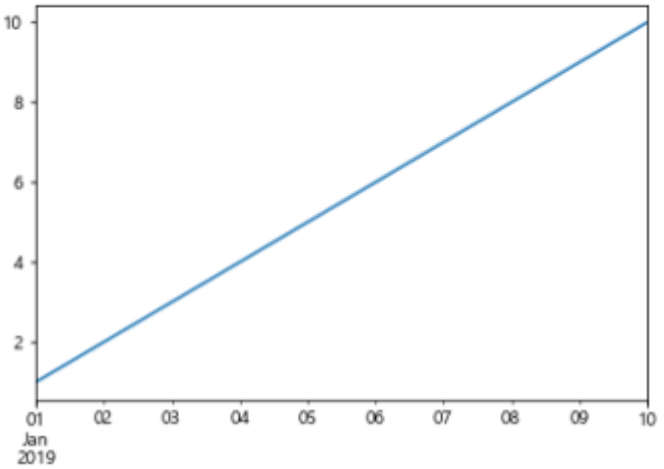
```
import pandas as pd
import matplotlib.pyplot as plt

s1 = pd.Series([1,2,3,4,5,6,7,8,9,10])
s1.plot()
plt.show()
```



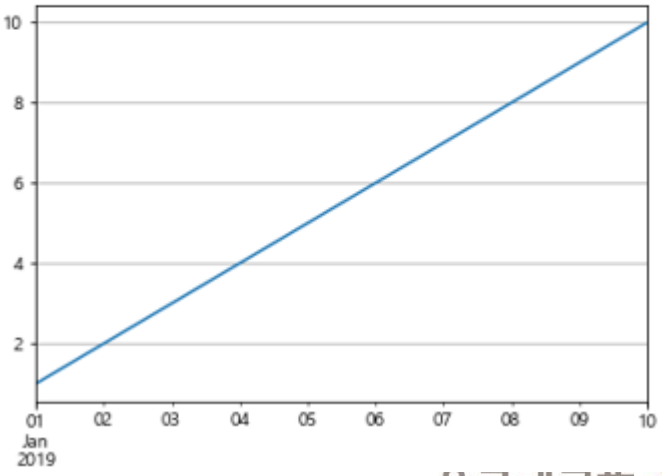

```
s2 = pd.Series([1,2,3,4,5,6,7,8,9,10],
               index=pd.date_range('2019-01-01',periods=10))
s2
```

```
2019-01-01     1
2019-01-02     2
2019-01-03     3
2019-01-04     4
2019-01-05     5
2019-01-06     6
2019-01-07     7
2019-01-08     8
2019-01-09     9
2019-01-10    10
Freq: D, dtype: int64
```

```
s2.plot()
plt.show()
```




```
s2.plot(grid=True)
plt.show()
```



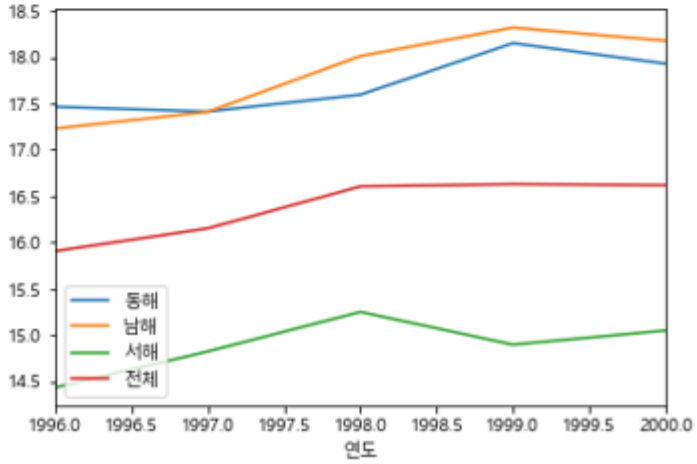
Pandas의 선 그래프

In [71]:
```
df_rain=pd.read_csv('C:/Users/jyeon/Desktop/sea_rain1.csv',
                    encoding='cp949', index_col='연도')
df_rain
```

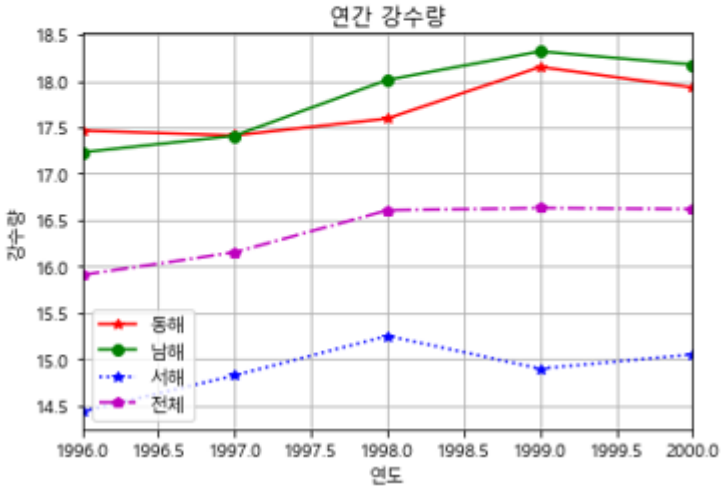
Out[71]:

연도	동해	남해	서해	전체
1996	17.4629	17.2288	14.4360	15.9067
1997	17.4116	17.4092	14.8248	16.1526
1998	17.5944	18.0110	15.2512	16.6044
1999	18.1495	18.3175	14.8979	16.6284
2000	17.9288	18.1766	15.0504	16.6178

In [72]:
```
plt.rcParams['font.family']='Malgun Gothic'
plt.rcParams['axes.unicode_minus'] = False
df_rain.plot()
plt.show()
```



In [74]:
```
rain_plot = df_rain.plot(grid = True,
                         style=['r-*','g-o','b:*','m-.p'])
rain_plot.set_xlabel("연도")
rain_plot.set_ylabel("강수량")
rain_plot.set_title("연간 강수량")
plt.show()
```



Pandas의 선 그래프

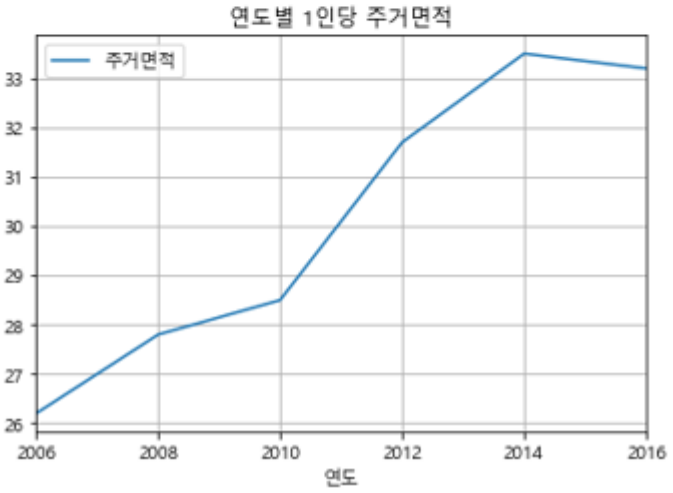
	연도	주거면적
0	2006	26.2
1	2008	27.8
2	2010	28.5
3	2012	31.7
4	2014	33.5
5	2016	33.2

In [75]:
```
year=[2006,2008,2010,2012,2014,2016]
area=[26.2,27.8,28.5,31.7,33.5,33.2]
table={'연도':year,'주거면적':area}
df_area=pd.DataFrame(table, columns=['연도','주거면적'])
df_area
```

Out[75]:

	연도	주거면적
0	2006	26.2
1	2008	27.8
2	2010	28.5
3	2012	31.7
4	2014	33.5
5	2016	33.2

In [77]:
```
df_area.plot(x='연도',y='주거면적',grid=True,
             title='연도별 1인당 주거면적')
plt.show()
```



동국대학교 dongguk university

산점도

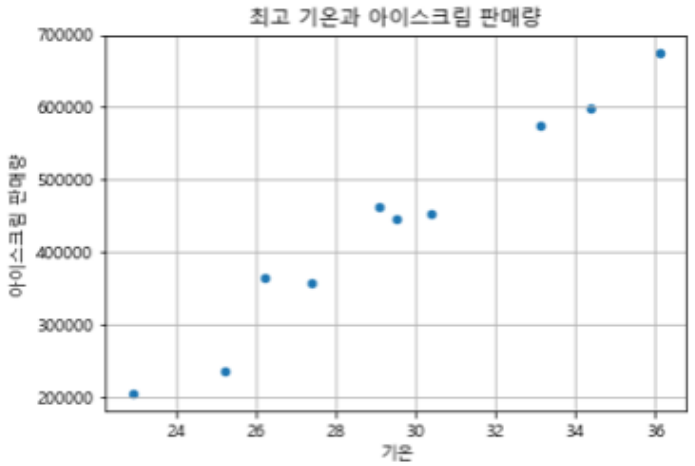
	기온	아이스크림 판매량
0	25.2	236500
1	27.4	357500
2	22.9	203500
3	26.2	365200
4	29.5	446600
5	33.1	574200
6	30.4	453200
7	36.1	675400
8	34.4	598400
9	29.1	463100

In [81]:
```
temperature = [25.2, 27.4, 22.9, 26.2, 29.5, 33.1, 30.4, 36.1, 34.4, 29.1]
ice_cream_sales=[236500, 357500, 203500, 365200, 446600, 574200, 453200, 675400, 598400, 463100]

dict_data={'기온':temperature,'아이스크림 판매량':ice_cream_sales}
df_ice_cream=pd.DataFrame(dict_data,columns=['기온','아이스크림 판매량'])

df_ice_cream
```

Out[81]:

	기온	아이스크림 판매량
0	25.2	236500
1	27.4	357500
2	22.9	203500
3	26.2	365200
4	29.5	446600
5	33.1	574200
6	30.4	453200
7	36.1	675400
8	34.4	598400
9	29.1	463100

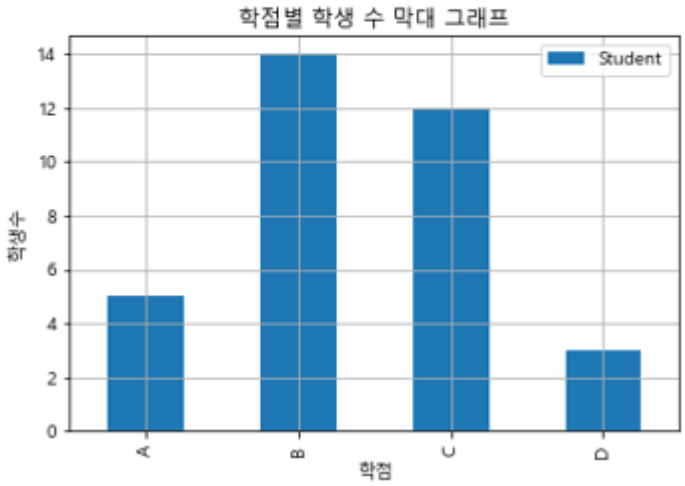
In [82]:
```
df_ice_cream.plot.scatter(x='기온',y='아이스크림 판매량',grid=True,
                          title='최고 기온과 아이스크림 판매량')
plt.show()
```



결과 그래프를 분석해보면 일일 최고 온도와 아이스크림 판매량이 비례관계

Pandas의 막대 그래프

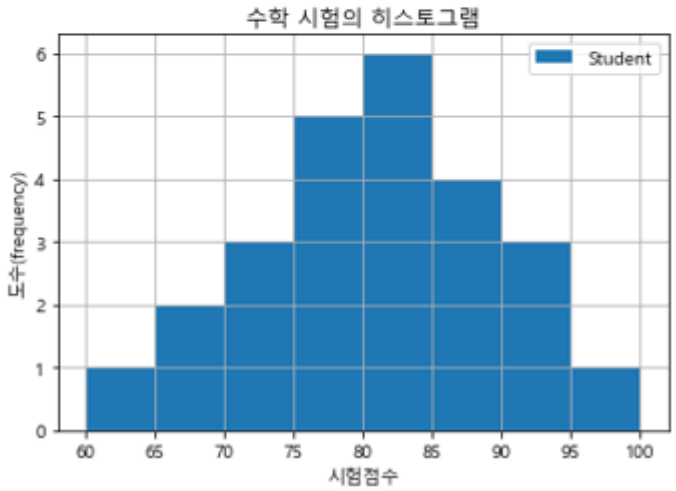
```
In [84]: grade_bar = df_grade.plot.bar(grid=True)
         grade_bar.set_xlabel("학점")
         grade_bar.set_ylabel("학생수")
         grade_bar.set_title("학점별 학생 수 막대 그래프")
         plt.show()
```



DataFrame 데이터의 Index인 [A,B,C,D]는 x축의 tick 라벨로 표시되었고, value인 [5,14,12,3]은 y축의 막대그래프로 표시
columns는 그래프의 범례로 표시

Pandas의 히스토그램

In [88]:
```
math = [76,82,84,83,90,86,85,92,72,71,100,87,81,
        76,94,78,81,60,79,69,74,87,82,68,79]
df_math = pd.DataFrame(math,columns=['Student'])

math_hist = df_math.plot.hist(bins=8, grid=True)
math_hist.set_xlabel("시험점수")
math_hist.set_ylabel("도수(frequency)")
math_hist.set_title("수학 시험의 히스토그램")

plt.show()
```



Pandas의 파이 그래프

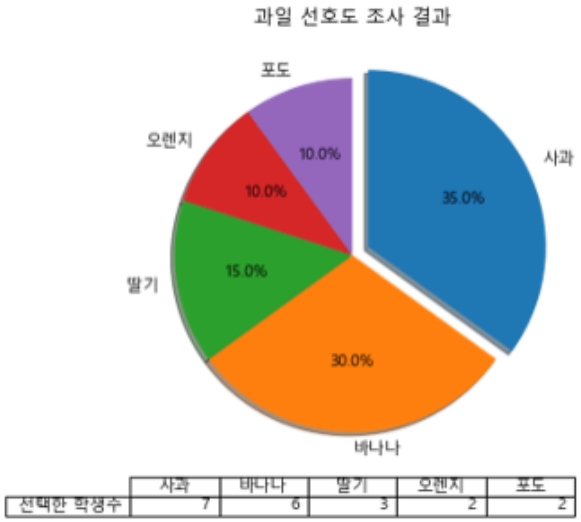
- 한 학급에서 20명의 학생이 5개의 과일 중 제일 좋아하는 과일을 선택한 것을 이용하여 분석

```
In [89]: fruit=['사과','바나나','딸기','오렌지','포도']
         result=[7, 6, 3, 2, 2]

         df_fruit = pd.Series(result, index = fruit, name='선택한 학생수')
         df_fruit
```

```
Out[89]: 사과    7
         바나나   6
         딸기    3
         오렌지   2
         포도    2
         Name: 선택한 학생수, dtype: int64
```

```
In [90]: df_fruit.plot.pie()
         plt.show()
```


```
explode_value=(0.1,0,0,0,0)
fruit_pie = df_fruit.plot.pie(figsize=(5,5),autopct='%.1f%%',
                              startangle=90,counterclock=False,
                              explode=explode_value, shadow=True, table=True)
fruit_pie.set_ylabel("") #불필요한 y축 라벨 제거
fruit_pie.set_title("과일 선호도 조사 결과")

#그래프를 이미지 파일로 저장, dpi는 200으로 설정
plt.savefig('C:/Users/jyeon/Desktop/saveFigTest3.png', dpi=200)
plt.show()
```



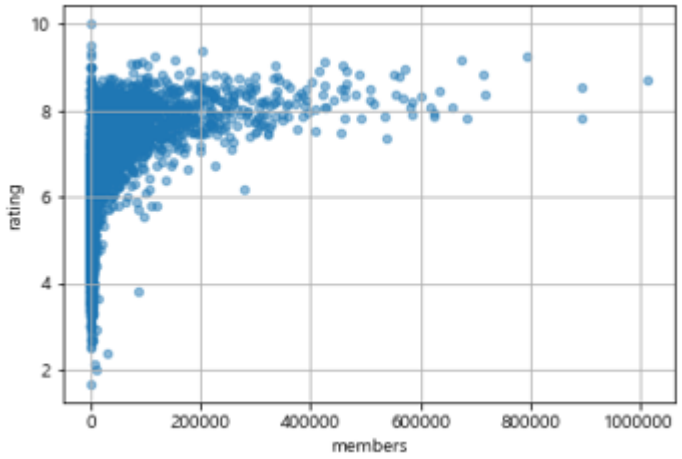
Pandas의 pie의 경우 figsize=(w,h)를 옵션 인자로 입력하여 그래프 창의 크기 조절, table = true 옵션은 데이터를 표 형식으로 출력

실습

- **anime_stock_returns.csv로 꺾은선 그래프를 출력**
  - 그래프의 크기를 가로 10inch, 세로 5inch로 지정하고, 아래와 같은 그래프 출력
  - 출력한 그래프를 saveTest.png 파일로 저장

실습

- **anime_master.csv로 산점도 그래프를 출력**
  - anime_master.csv 파일을 저장할 때 anime_id를 index로 함
  - X값으로 members를 y값으로 rating을 지정하고, 투명도는 0.5로 하여 산점도 그래프를 작성



  - 멤버 수 80만 이상인 작품을 추출하여라
  - 멤버 수 60만 이상 평점 8.5점 이상의 데이터를 추출하여라