

2021년도

ICT 이노베이션스퀘어 확산 사업

[인공지능 : 기초부터 실전까지]

▶ 11차수 ~ 13차수 강의 교안

※ 본 교안은 강의 수강 용도로만 사용 가능합니다.
상업적 이용을 일절 금함.

12



데이터 시각화

matplotlib로 그래프 그리기

- 데이터를 효과적으로 시각화하기 위한 파이썬 라이브러리
- MATLAB(과학 및 공학 연산을 위한 소프트웨어)의 시각화 기능을 모델링해서 만들어짐
- 몇 줄의 코드로 간단하게 그래프 그리기 가능
 - 2차원 선 그래프(plot), 산점도(scatter plot), 막대 그래프(bar chart), 히스토그램(histogram), 파이 그래프(pie chart) 등
- 아나콘다 배포판에 포함돼 있음
- matplotlib 홈페이지: <http://matplotlib.org/>

matplotlib로 그래프 그리기

- matplotlib 불러오기

```
In: import matplotlib.pyplot as plt
```

```
%matplotlib qt
```

```
%matplotlib inline
```

- 선 그래프
 - 기본적인 선 그래프 그리기

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

```
plt.plot(y)
```

```
plt.plot(y, fmt)
```

```
plt.plot(x, y)
```

```
plt.plot(x, y, fmt)
```

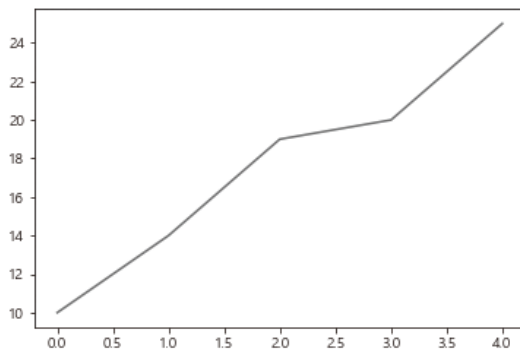
matplotlib로 그래프 그리기

– 기본적인 선 그래프 그리기

```
In: data1 = [10, 14, 19, 20, 25]
```

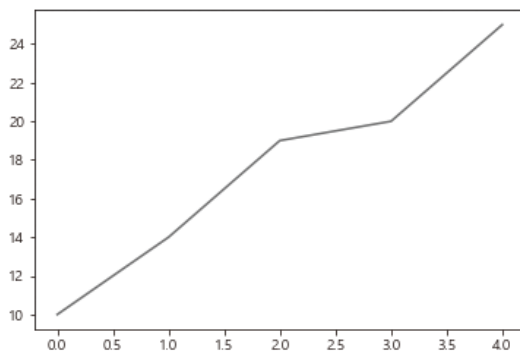
```
In: plt.plot(data1)
```

```
Out: [<matplotlib.lines.Line2D at 0x23310f30588>]
```



```
In: plt.plot(data1)
```

```
plt.show()
```



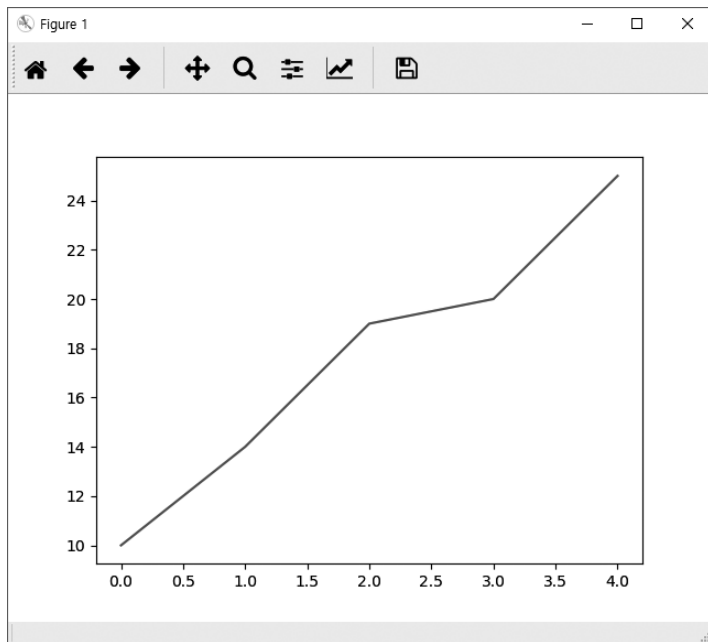
matplotlib로 그래프 그리기

– 기본적인 선 그래프 그리기

```
In: %matplotlib qt
```

```
In: plt.plot(data1)
```

```
Out: [<matplotlib.lines.Line2D at 0x23310f38be0>]
```



```
In: %matplotlib inline
```

matplotlib로 그래프 그리기

– 기본적인 선 그래프 그리기

```
In: import numpy as np
```

```
    x = np.arange(-4.5, 5, 0.5) # 배열 x 생성. 범위: [-4.5, 5), 0.5씩 증가
```

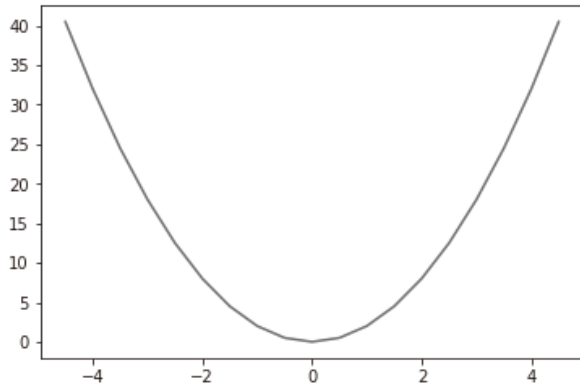
```
    y = 2*x**2 # 수식을 이용해 배열 x에 대응하는 배열 y 생성
```

```
    [x,y]
```

```
Out: [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: plt.plot(x,y)
```

```
    plt.show()
```



matplotlib로 그래프 그리기

– 여러 그래프 그리기

```
plt.plot([x1,] y1 [, fmt1])  
plt.plot([x2,] y2 [, fmt2])  
...  
plt.plot([xn,] yn [, fmtn])
```

```
plt.plot(x1, y1 [, fmt1], x2, y2 [, fmt2], ..., xn, yn [, fmtn])
```

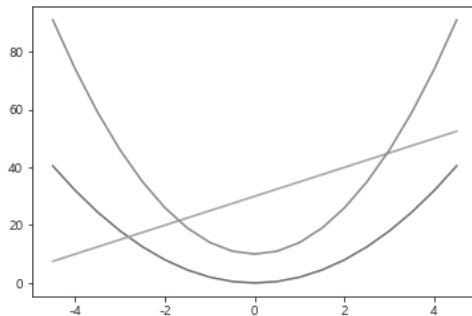
```
plt.plot(x1, y1, x2, y2, ..., xn, yn)  
plt.plot(x1, y1, fmt1, x2, y2, fmt2, ..., xn, yn, fmtn)  
plt.plot(x1, y1, x2, y2, fmt2, ..., xn, yn)  
plt.plot(x1, y1, fmt1, x2, y2, ..., xn, yn)
```

```
In: import numpy as np  
    x = np.arange(-4.5, 5, 0.5)  
    y1 = 2*x**2  
    y2 = 5*x + 30  
    y3 = 4*x**2 + 10
```

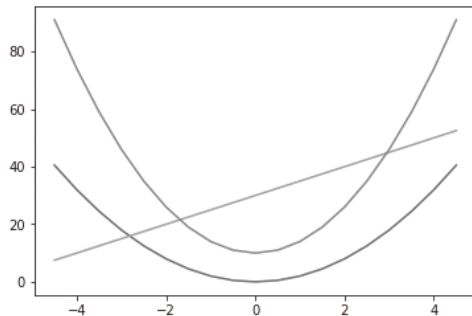

matplotlib로 그래프 그리기

– 여러 그래프 그리기

```
In: import matplotlib.pyplot as plt  
    plt.plot(x, y1)  
    plt.plot(x, y2)  
    plt.plot(x, y3)  
    plt.show()
```



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



matplotlib로 그래프 그리기

– 여러 그래프 그리기

- 새로운 그래프 창에 그래프 그리기

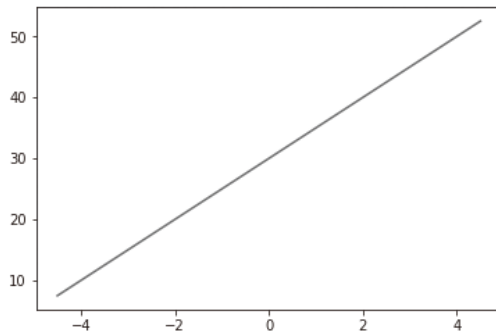
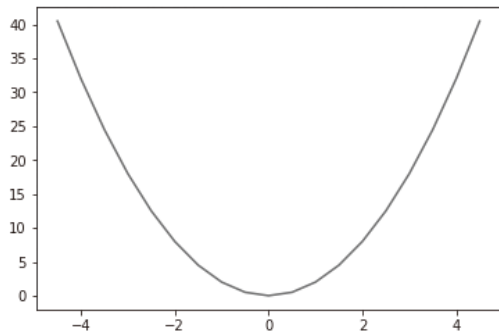
```
plt.figure()
```

```
In: plt.plot(x, y1)  # 처음 그리기 함수를 수행하면 그래프 창이 자동으로 생성됨
```

```
plt.figure()  # 새로운 그래프 창을 생성함
```

```
plt.plot(x, y2)  # 새롭게 생성된 그래프 창에 그래프를 그림
```

```
plt.show()
```



matplotlib로 그래프 그리기

– 여러 그래프 그리기

- 그래프 창의 번호를 명시적으로 지정한 후 해당 창에 그래프 그리기

```
plt.figure(n)
```

```
In: import numpy as np
```

```
# 데이터 생성
```

```
x = np.arange(-5, 5, 0.1)
```

```
y1 = x**2 - 2
```

```
y2 = 20*np.cos(x)**2 # NumPy에서 cos()는 np.cos()으로 입력
```

```
plt.figure(1) # 1번 그래프 창을 생성함
```

```
plt.plot(x, y1) # 지정된 그래프 창에 그래프를 그림
```

```
plt.figure(2) # 2번 그래프 창을 생성함
```

```
plt.plot(x, y2) # 지정된 그래프 창에 그래프를 그림
```

```
plt.figure(1) # 이미 생성된 1번 그래프 창을 지정함
```

```
plt.plot(x, y2) # 지정된 그래프 창에 그래프를 그림
```

```
plt.figure(2) # 이미 생성된 2번 그래프 창을 지정함
```

```
plt.clf() # 2번 그래프 창에 그려진 모든 그래프를 지움
```

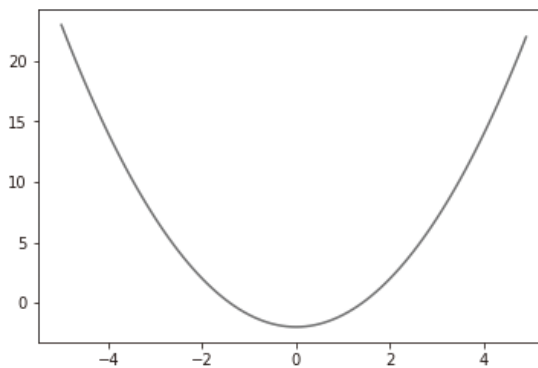
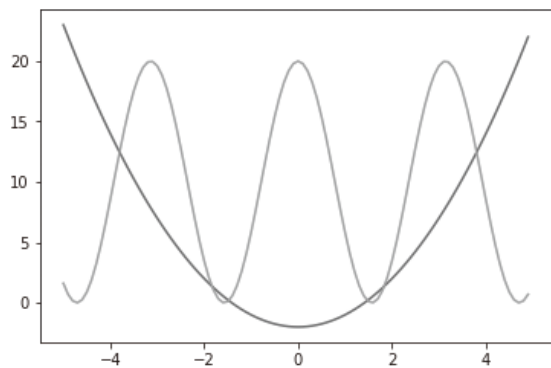
```
plt.plot(x, y1) # 지정된 그래프 창에 그래프를 그림
```

```
plt.show()
```

matplotlib로 그래프 그리기

– 여러 그래프 그리기

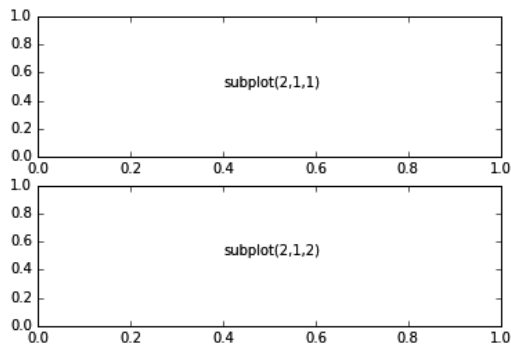
- 그래프 창의 번호를 명시적으로 지정한 후 해당 창에 그래프 그리기



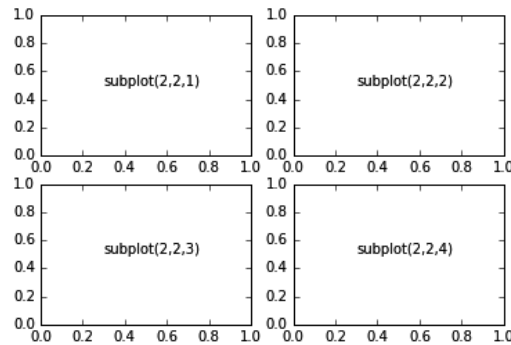
matplotlib로 그래프 그리기

- 하위 그래프 영역으로 나눈 후 각 영역에 그래프 그리기

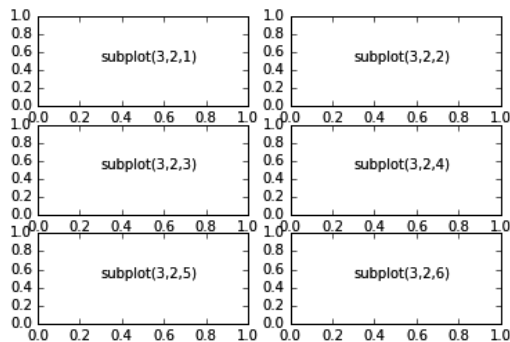
```
plt.subplot(m, n, p)
```



subplot(2, 1, p) 그래프



subplot(3, 2, p) 그래프



subplot(2, 2, p) 그래프

matplotlib로 그래프 그리기

– 하위 그래프 영역으로 나눈 후 각 영역에 그래프 그리기

```
plt.subplot(m, n, p)
```

```
In: import numpy as np
```

```
# 데이터 생성
```

```
x = np.arange(0, 10, 0.1)
```

```
y1 = 0.3*(x-5)**2 + 1
```

```
y2 = -1.5*x + 3
```

```
y3 = np.sin(x)**2 # NumPy에서 sin()은 np.sin()으로 입력
```

```
y4 = 10*np.exp(-x) + 1 # NumPy에서 exp()는 np.exp()로 입력
```

```
# 2 x 2 행렬로 이뤄진 하위 그래프에서 p에 따라 위치를 지정
```

```
plt.subplot(2,2,1) # p는 1
```

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

```
plt.subplot(2,2,2) # p는 2
```

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

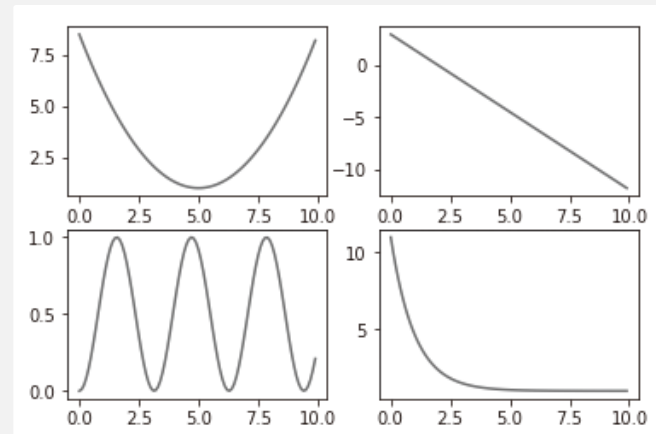
```
plt.subplot(2,2,3) # p는 3
```

```
plt.plot(x,y3)
```

```
plt.subplot(2,2,4) # p는 4
```

```
plt.plot(x,y4)
```

```
plt.show()
```



matplotlib로 그래프 그리기

– 그래프의 출력 범위 지정하기

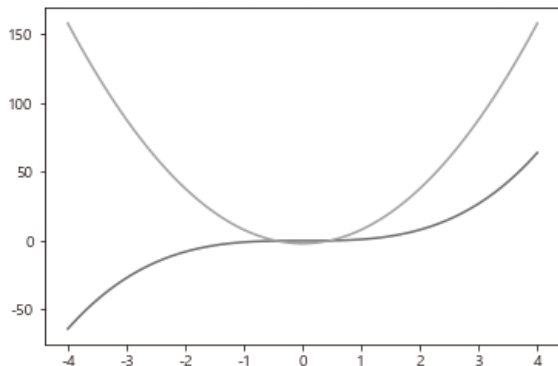
```
plt.xlim(xmin, xmax) # x축의 좌표 범위 지정(xmin ~ xmax)  
plt.ylim(ymin, ymax) # y축의 좌표 범위 지정(xmin ~ xmax)
```

```
[xmin, xmax] = plt.xlim() # x축의 좌표 범위 가져오기  
[ymin, ymax] = plt.ylim() # y축의 좌표 범위 가져오기
```

```
In: import numpy as np
```

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

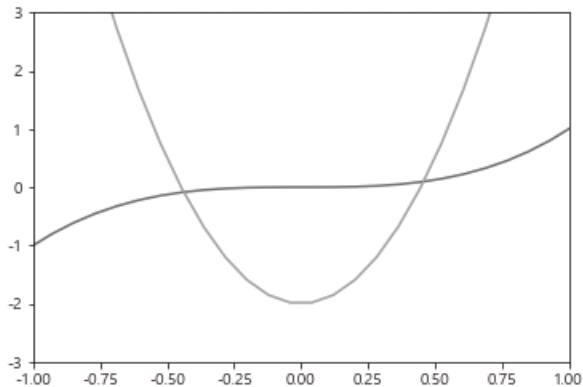
```
plt.plot(x, y1, x, y2)  
plt.show()
```



matplotlib로 그래프 그리기

– 그래프의 출력 범위 지정하기

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



matplotlib로 그래프 그리기

- 그래프 꾸미기
 - 출력 형식 지정

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

- 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)

matplotlib로 그래프 그리기

– 출력 형식 지정

- 마커 지정을 위한 약어

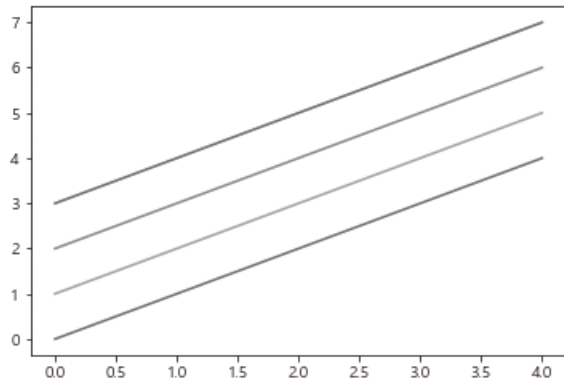
마커 약어	마커
o	원 모양 ●
^, v, <, >	삼각형 위쪽(▲), 아래쪽(▼), 왼쪽(◀), 오른쪽(▶) 방향
s	사각형(square) ■
p	오각형(pentagon)
h, H	육각형(hexagon)1, 육각형2
*	별 모양(star) ★
+	더하기(plus) +
x, X	x, 채워진 x
D, d	다이아몬드(diamond, ◆), 얇은 다이아몬드

matplotlib로 그래프 그리기

– 출력 형식 지정

```
In: import numpy as np  
    x = np.arange(0, 5, 1)  
    y1 = x  
    y2 = x + 1  
    y3 = x + 2  
    y4 = x + 3
```

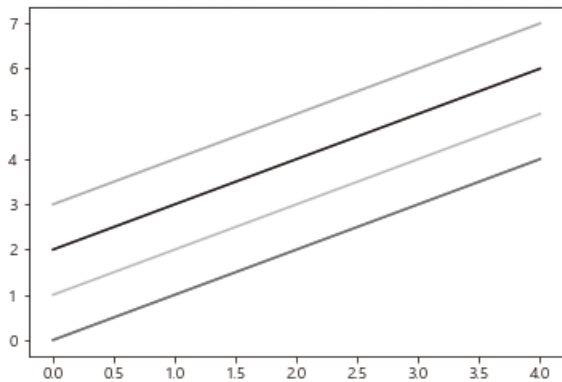
```
In: plt.plot(x, y1, x, y2, x, y3, x, y4)  
    plt.show()
```



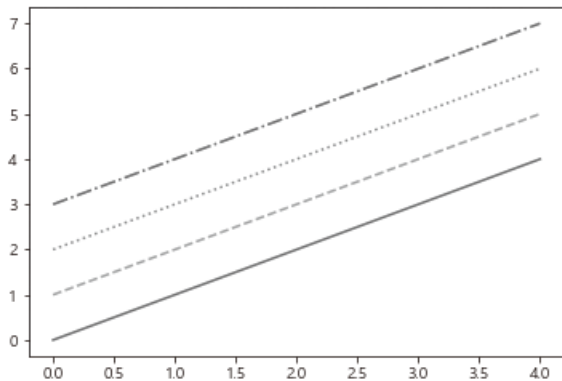
matplotlib로 그래프 그리기

– 출력 형식 지정

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



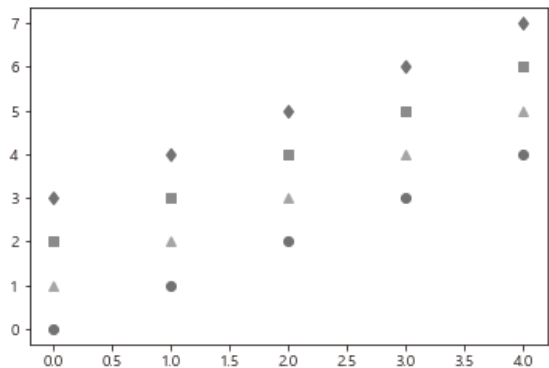
```
In: plt.plot(x, y1, '-', x, y2, '--', x, y3, ':', x, y4, '-.')  
    plt.show()
```



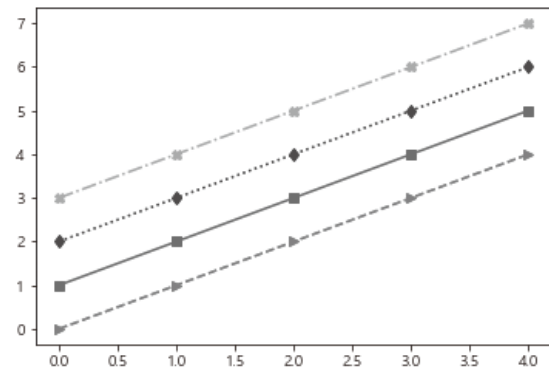
matplotlib로 그래프 그리기

– 출력 형식 지정

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



```
In: plt.plot(x, y1, '>--r', x, y2, 's-g', x, y3, 'd:b', x, y4, '-.Xc')  
    plt.show()
```



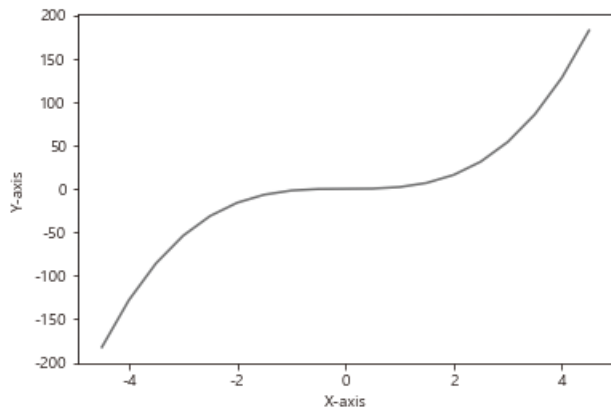
matplotlib로 그래프 그리기

– 라벨, 제목, 격자, 범례, 문자열 표시

```
In: import numpy as np
```

```
x = np.arange(-4.5, 5, 0.5)  
y = 2*x**3
```

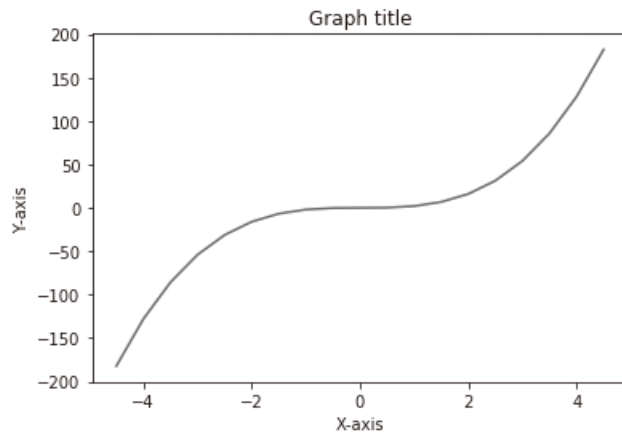
```
plt.plot(x,y)  
plt.xlabel('X-axis')  
plt.ylabel('Y-axis')  
plt.show()
```



matplotlib로 그래프 그리기

– 라벨, 제목, 격자, 범례, 문자열 표시

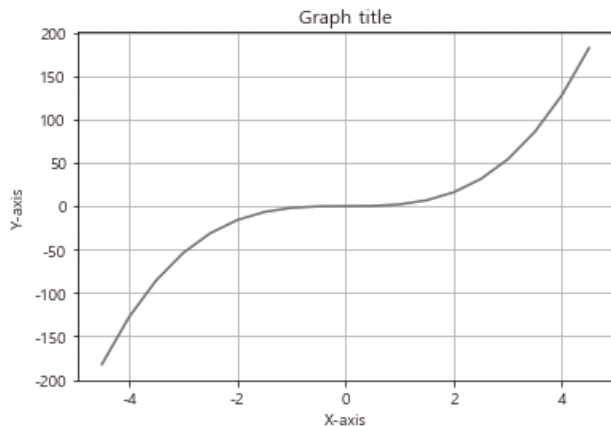
```
In: plt.plot(x,y)  
    plt.xlabel('X-axis')  
    plt.ylabel('Y-axis')  
    plt.title('Graph title')  
    plt.show()
```



matplotlib로 그래프 그리기

– 라벨, 제목, 격자, 범례, 문자열 표시

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

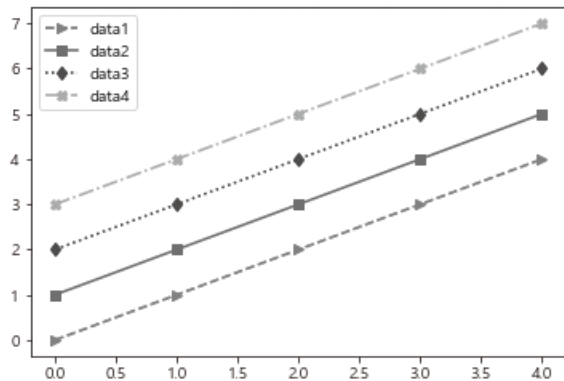


matplotlib로 그래프 그리기

– 라벨, 제목, 격자, 범례, 문자열 표시

```
In: import numpy as np
    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()
```



matplotlib로 그래프 그리기

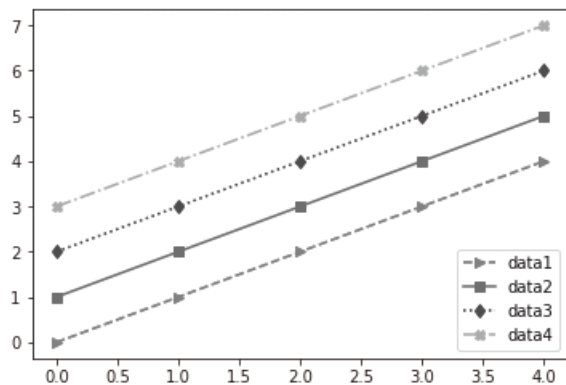
- 라벨, 제목, 격자, 범례, 문자열 표시
 - legend()에서 loc 옵션으로 범례의 위치 지정

범례 위치	위치 문자열	위치 코드
최적 위치 자동 선정	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	lower center	8
상단 중앙	upper center	9
중앙	center	10

matplotlib로 그래프 그리기

– 라벨, 제목, 격자, 범례, 문자열 표시

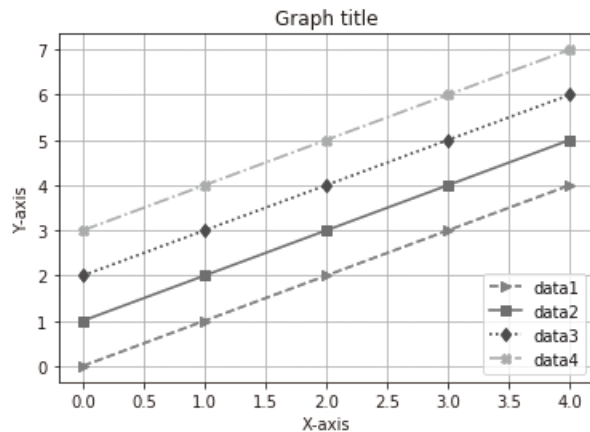
```
In: plt.plot(x, y1, '>--r', x, y2, 's-g', x, y3, 'd:b', x, y4, '-.Xc')  
    plt.legend(['data1', 'data2', 'data3', 'data4'], loc = 'lower right')  
    plt.show()
```



matplotlib로 그래프 그리기

– 라벨, 제목, 격자, 범례, 문자열 표시

```
In: 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)
```



matplotlib로 그래프 그리기

- 라벨, 제목, 격자, 범례, 문자열 표시
 - 한글 표시

```
import matplotlib
matplotlib.rcParams['font.family']
```

```
matplotlib.rcParams['font.family'] = '폰트 이름'
matplotlib.rcParams['axes.unicode_minus'] = False
```

```
import matplotlib.font_manager
```

```
font_list = matplotlib.font_manager.get_fontconfig_fonts()
font_names = [matplotlib.font_manager.FontProperties(fname=fname).get_name() for fname in
font_list]
f = open("C:\WmyPyCode\Wmy_font_list.txt", 'w')

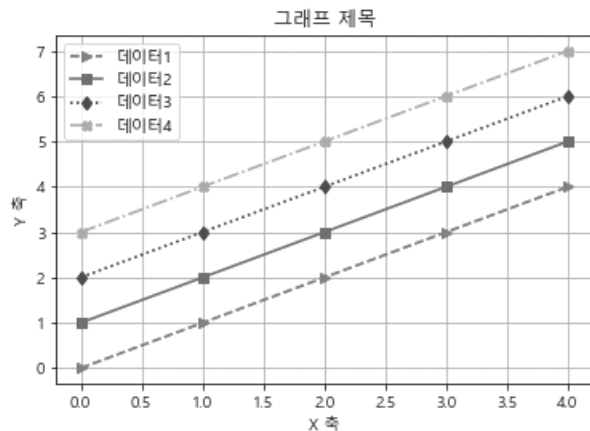
for font_name in font_names:
    f.write(font_name + "\n")
f.close()
```

```
In: import matplotlib
    matplotlib.rcParams['font.family'] = 'Malgun Gothic'      # '맑은 고딕'으로 설정
    matplotlib.rcParams['axes.unicode_minus'] = False
```

matplotlib로 그래프 그리기

- 라벨, 제목, 격자, 범례, 문자열 표시
 - 한글 표시

```
In: 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)
```

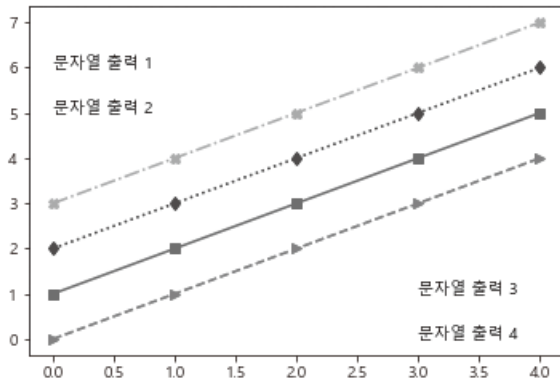


matplotlib로 그래프 그리기

- 라벨, 제목, 격자, 범례, 문자열 표시
 - 그래프 창에 좌표(x, y)를 지정해 문자열(str)을 표시

```
plt.text(x, y, str)
```

```
In: plt.plot(x, y1, '>--r', x, y2, 's-g', x, y3, 'd:b', x, y4, '-.Xc')  
    plt.text(0, 6, "문자열 출력 1")  
    plt.text(0, 5, "문자열 출력 2")  
    plt.text(3, 1, "문자열 출력 3")  
    plt.text(3, 0, "문자열 출력 4")  
    plt.show()
```

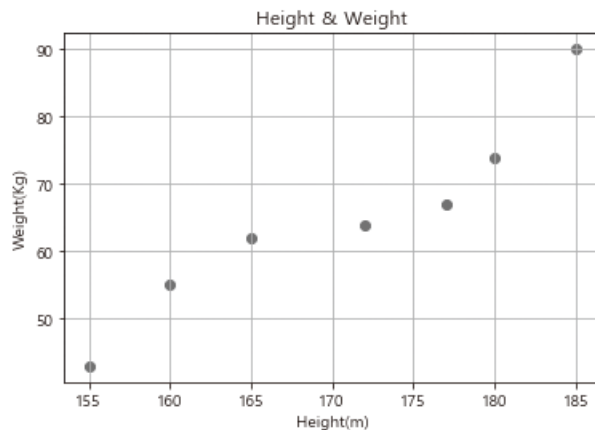


matplotlib로 그래프 그리기

– 산점도

```
plt.scatter(x, y [,s=size_n, c=colors, marker='marker_string', alpha=alpha_f])
```

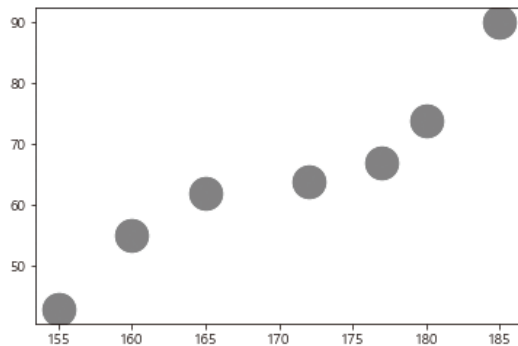
```
In: import matplotlib.pyplot as plt  
    height = [165, 177, 160, 180, 185, 155, 172] # 키 데이터  
    weight = [62, 67, 55, 74, 90, 43, 64] # 몸무게 데이터  
    plt.scatter(height, weight)  
    plt.xlabel('Height(m)')  
    plt.ylabel('Weight(Kg)')  
    plt.title('Height & Weight')  
    plt.grid(True)
```



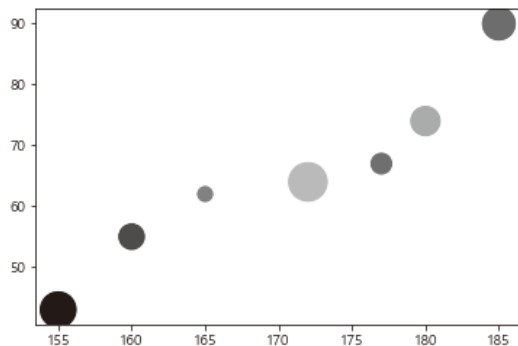
matplotlib로 그래프 그리기

– 산점도

```
In: plt.scatter(height, weight, s=500, c='r') # 마커 크기는 500, 컬러는 붉은색(red)
    plt.show()
```



```
In: size = 100 * np.arange(1,8) # 데이터별로 마커의 크기 지정
    colors = ['r', 'g', 'b', 'c', 'm', 'k', 'y'] # 데이터별로 마커의 컬러 지정
    plt.scatter(height, weight, s=size, c=colors)
    plt.show()
```



matplotlib로 그래프 그리기

– 산점도

```
In: import numpy as np
```

```
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]
```

```
# 인구 밀도(명/km^2): 2017년 통계청 자료
```

```
pop_den = [16154, 2751, 2839, 2790, 1099, 4454, 2995]
```

```
size = np.array(pop_den) * 0.2 # 마커의 크기 지정
```

```
colors = ['r', 'g', 'b', 'c', 'm', 'k', 'y'] # 마커의 컬러 지정
```

```
plt.scatter(lon, lat, s=size, c=colors, alpha=0.5)
```

```
plt.xlabel('경도(longitude)')
```

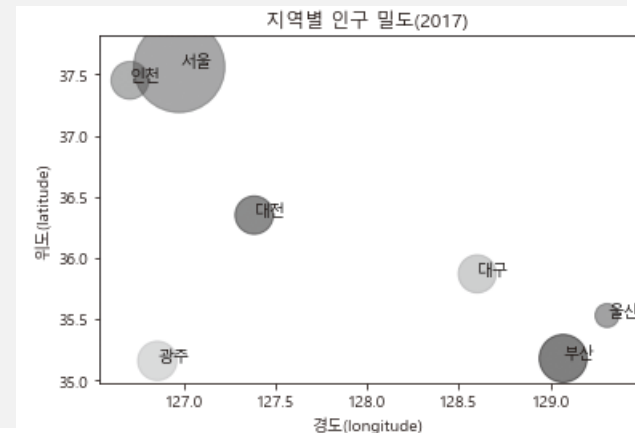
```
plt.ylabel('위도(latitude)')
```

```
plt.title('지역별 인구 밀도(2017)')
```

```
for x, y, name in zip(lon, lat, city):
```

```
    plt.text(x, y, name) # 위도 경도에 맞게 도시 이름 출력
```

```
plt.show()
```



matplotlib로 그래프 그리기

- 막대 그래프

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

- 운동 시작 전과 한 달 후의 윗몸 일으키기 횟수

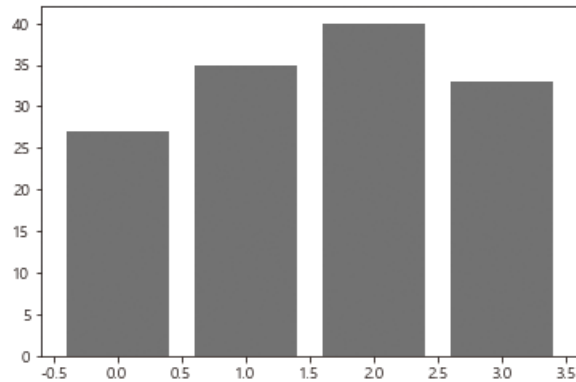
회원 ID	운동 시작 전	운동 한 달 후
m_01	27	30
m_02	35	38
m_03	40	42
m_04	33	37

```
In: member_IDs = ['m_01', 'm_02', 'm_03', 'm_04'] # 회원 ID
    before_ex = [27, 35, 40, 33] # 운동 시작 전
    after_ex = [30, 38, 42, 37] # 운동 한 달 후
```

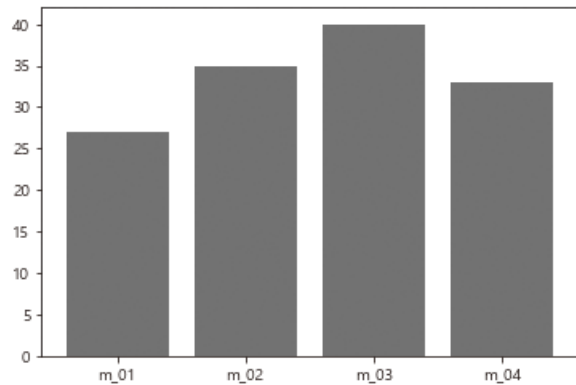
```
In: import matplotlib.pyplot as plt
    import numpy as np
    n_data = len(member_IDs) # 회원이 네 명이므로 전체 데이터 수는 4
    index = np.arange(n_data) # NumPy를 이용해 배열 생성 (0, 1, 2, 3)
    plt.bar(index, before_ex) # bar(x,y)에서 x=index, height=before_ex 로 지정
    plt.show()
```

matplotlib로 그래프 그리기

— 막대 그래프



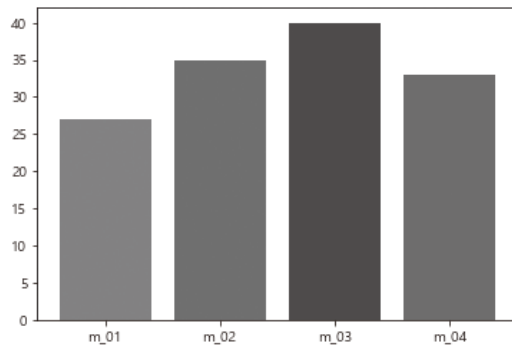
```
In: plt.bar(index, before_ex, tick_label = member_IDs)
    plt.show()
```



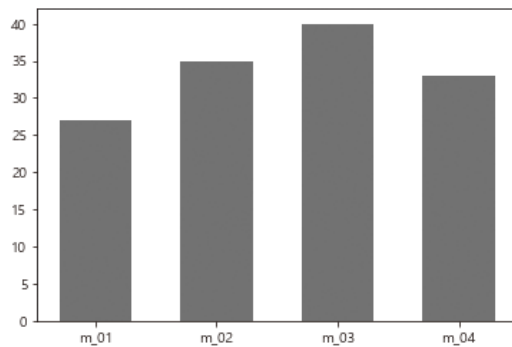
matplotlib로 그래프 그리기

- 막대 그래프

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



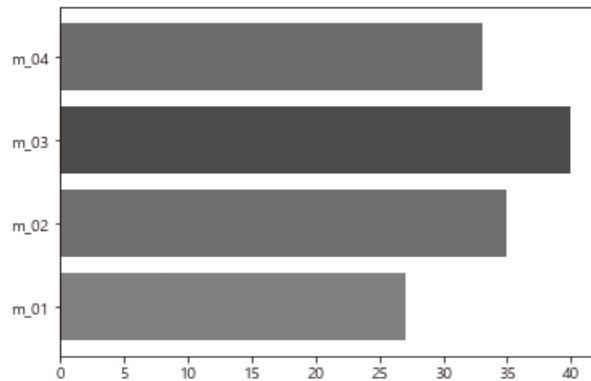
```
In: plt.bar(index, before_ex, tick_label = member_IDs, width = 0.6)  
    plt.show()
```



matplotlib로 그래프 그리기

- 막대 그래프

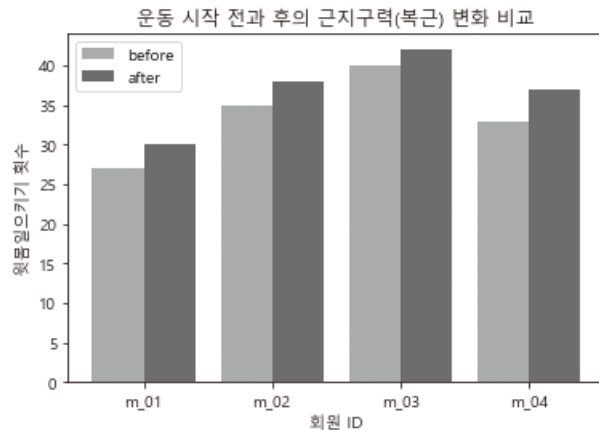
```
In: colors=['r', 'g', 'b', 'm']  
    plt.barh(index, before_ex, color = colors, tick_label = member_IDs)  
    plt.show()
```



matplotlib로 그래프 그리기

- 막대 그래프

```
In: barWidth = 0.4
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()
```



matplotlib로 그래프 그리기

- 히스토그램

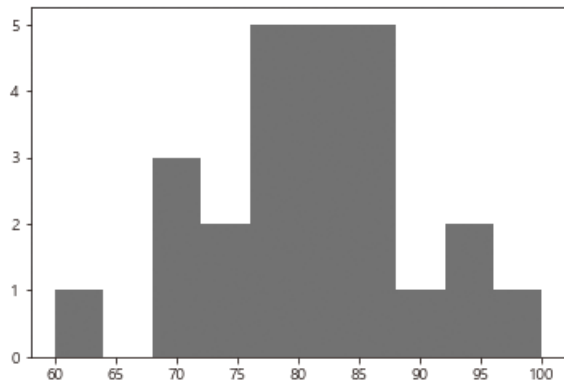
```
plt.hist(x, [,bins = bins_n 혹은 'auto'])
```

```
In: import matplotlib.pyplot as plt
```

```
    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)
```

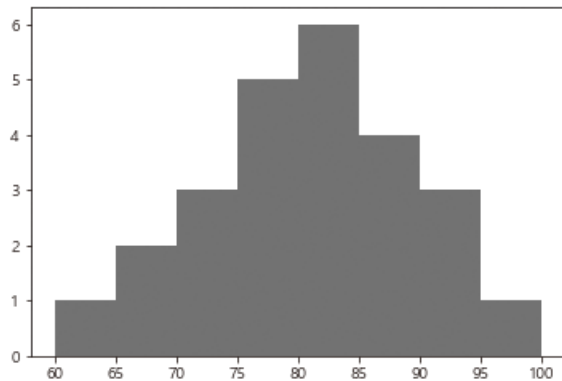
```
Out: (array([1., 0., 3., 2., 5., 5., 5., 1., 2., 1.]),  
      array([ 60.,  64.,  68.,  72.,  76.,  80.,  84.,  88.,  92.,  96., 100.]),  
      <a list of 10 Patch objects>)
```



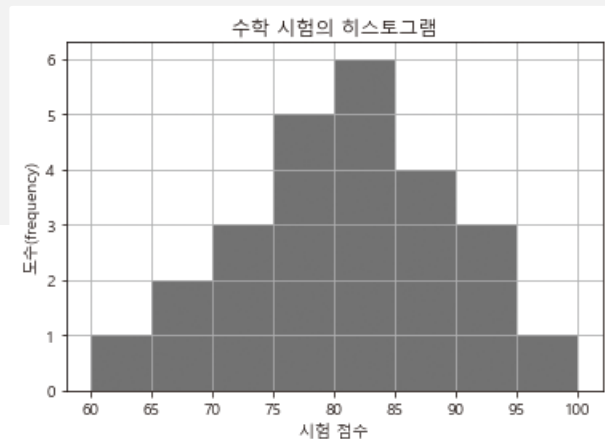
matplotlib로 그래프 그리기

– 히스토그램

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



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



matplotlib로 그래프 그리기

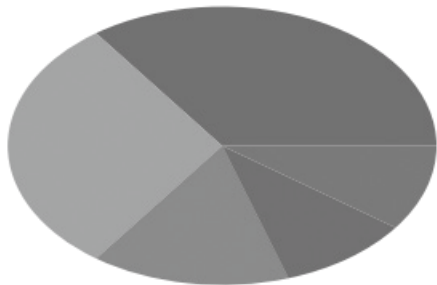
– 파이 그래프

```
plt.pie(x, [ ,labels = label_seq, autopct='비율 표시 형식(ex: %0.1f)', shadow = False(기본) 혹은 True, explode = explode_seq, counterclock = True(기본) 혹은 False, startangle = 각도 (기본은 0) ])
```

```
plt.figure(figsize = (w,h))
```

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

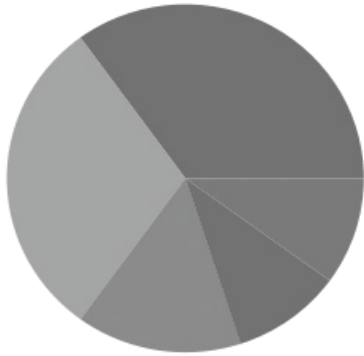
```
In: import matplotlib.pyplot as plt  
    plt.pie(result)  
    plt.show()
```



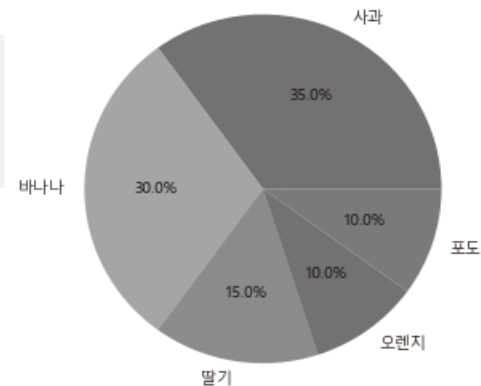
matplotlib로 그래프 그리기

- 파이 그래프

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



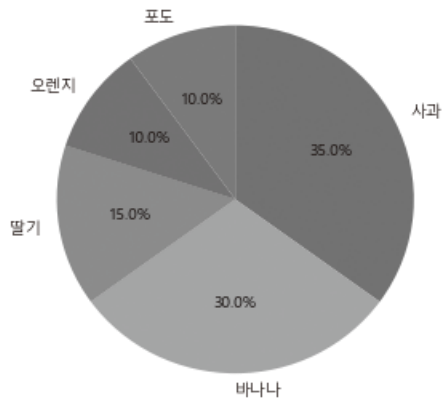
```
In: plt.figure(figsize=(5,5))  
    plt.pie(result, labels= fruit, autopct='%1f%%')  
    plt.show()
```



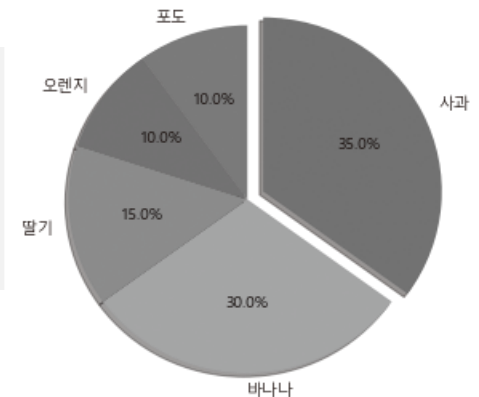
matplotlib로 그래프 그리기

– 파이 그래프

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



```
In: 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()
```



matplotlib로 그래프 그리기

- 그래프 저장하기

```
plt.savefig(file_name, [,dpi = dpi_n(기본은 72)])
```

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

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

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

```
Out: 72.0
```

matplotlib로 그래프 그리기

- 그래프 저장하기

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

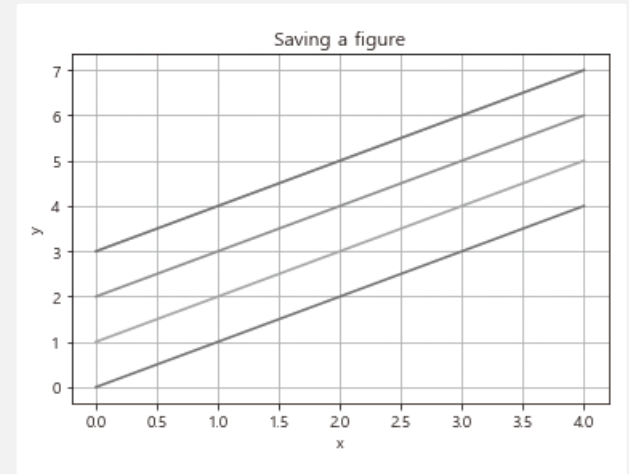
```
x = np.arange(0, 5, 1)
y1 = x
y2 = x + 1
y3 = x + 2
y4 = x + 3
```

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

```
plt.grid(True)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Saving a figure')
```

그래프를 이미지 파일로 저장. dpi는 100으로 설정

```
plt.savefig('C:/myPyCode/figures/saveFigTest1.png', dpi = 100)
plt.show()
```



matplotlib로 그래프 그리기

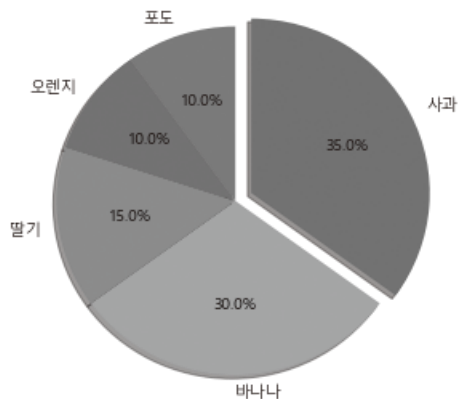
- 그래프 저장하기

```
In: import matplotlib.pyplot as plt
```

```
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)) # 그래프의 크기를 지정  
plt.pie(result, labels= fruit, autopct='%0.1f%%', startangle=90, counterclock = False,  
explode=explode_value, shadow=True)
```

```
# 그래프를 이미지 파일로 저장. dpi는 200으로 설정  
plt.savefig('C:/myPyCode/figures/saveFigTest2.png', dpi = 200)  
plt.show()
```



pandas로 그래프 그리기

- pandas의 그래프 구조

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

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

- pandas의 그래프 종류 선택

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

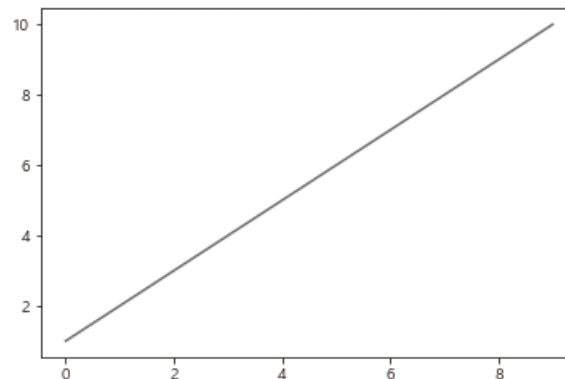
pandas로 그래프 그리기

- pandas의 선 그래프

```
In: import pandas as pd
import matplotlib.pyplot as plt
s1 = pd.Series([1,2,3,4,5,6,7,8,9,10])
s1
```

```
Out: 0    1
     1    2
     2    3
     3    4
     4    5
     5    6
     6    7
     7    8
     8    9
     9   10
     dtype: int64
```

```
In: s1.plot()
plt.show()
```



pandas로 그래프 그리기

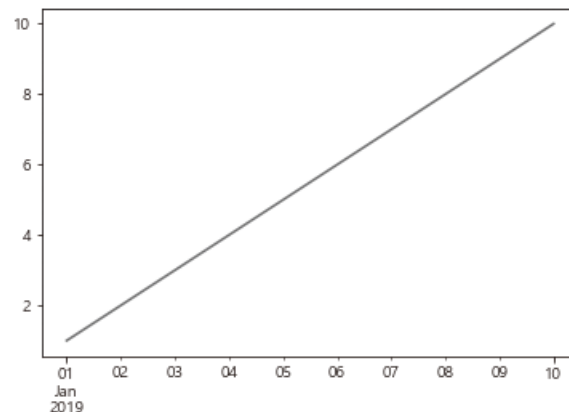
- pandas의 선 그래프

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

s2

```
Out: 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
```

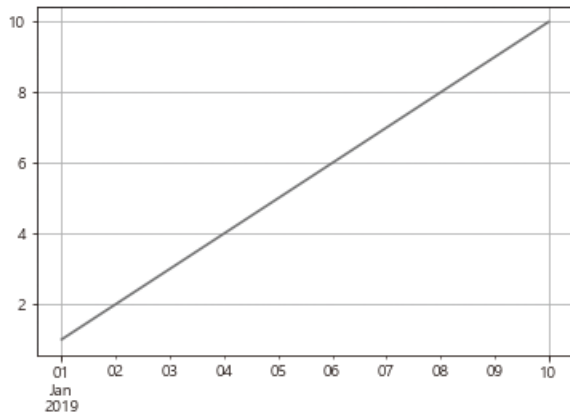
```
In: s2.plot()  
    plt.show()
```



pandas로 그래프 그리기

- pandas의 선 그래프

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



pandas로 그래프 그리기

- pandas의 선 그래프

```
In: df_rain = pd.read_csv('C:/myPyCode/data/sea_rain1.csv', index_col="연도" )
```

```
df_rain
```

Out:

	동해	남해	서해	전체
연도				
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: import matplotlib
```

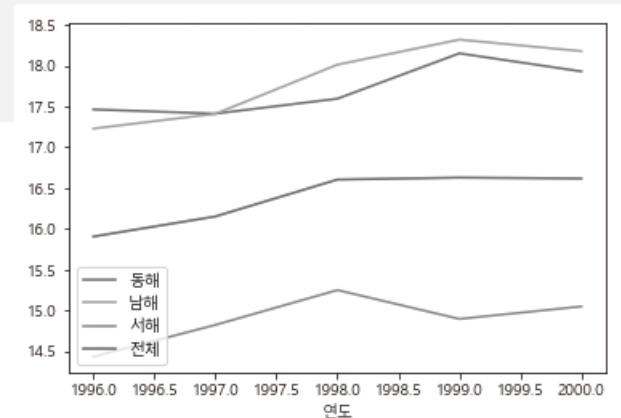
```
matplotlib.rcParams['font.family'] = 'Malgun Gothic'
```

```
matplotlib.rcParams['axes.unicode_minus'] = False
```

```
df_rain.plot()
```

```
plt.show()
```

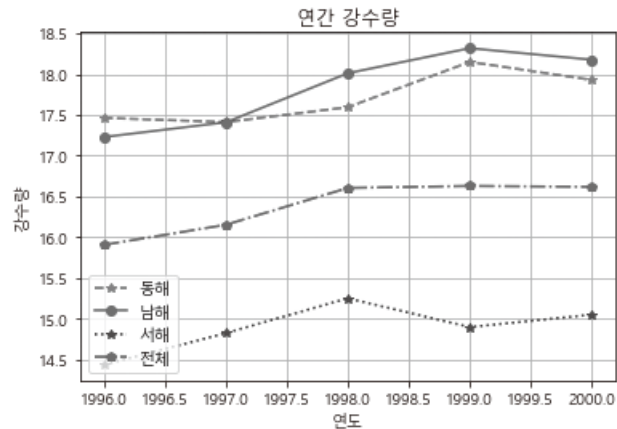
'맑은 고딕'으로 설정



pandas로 그래프 그리기

- pandas의 선 그래프

```
In: 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()
```



pandas로 그래프 그리기

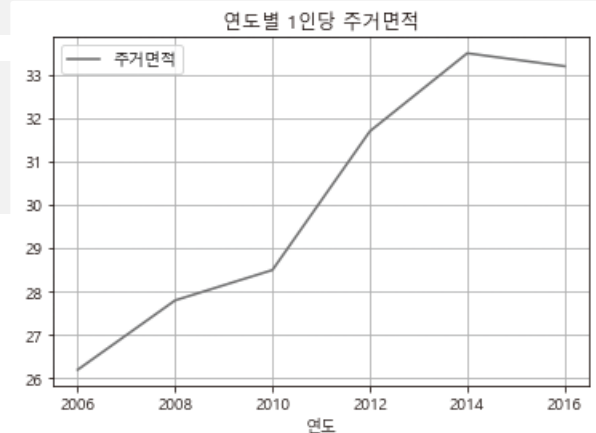
- pandas의 선 그래프

```
In: year = [2006, 2008, 2010, 2012, 2014, 2016] # 연도
    area = [26.2, 27.8, 28.5, 31.7, 33.5, 33.2] # 1인당 주거면적
    table = {'연도':year, '주거면적':area}
    df_area = pd.DataFrame(table, columns=['연도', '주거면적'])
    df_area
```

Out:

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



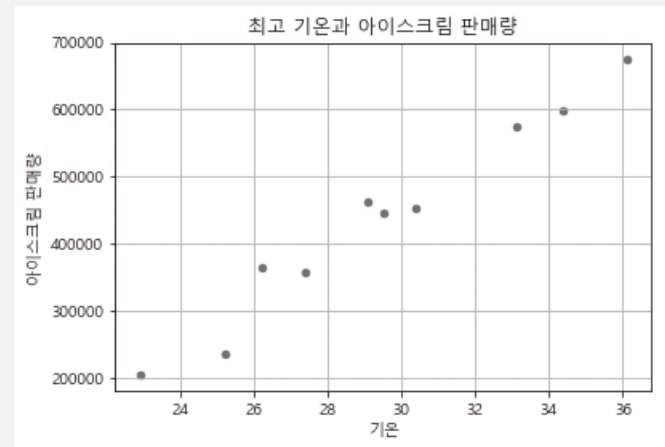
pandas로 그래프 그리기

- pandas의 산점도

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

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

pandas로 그래프 그리기

- pandas의 막대그래프

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

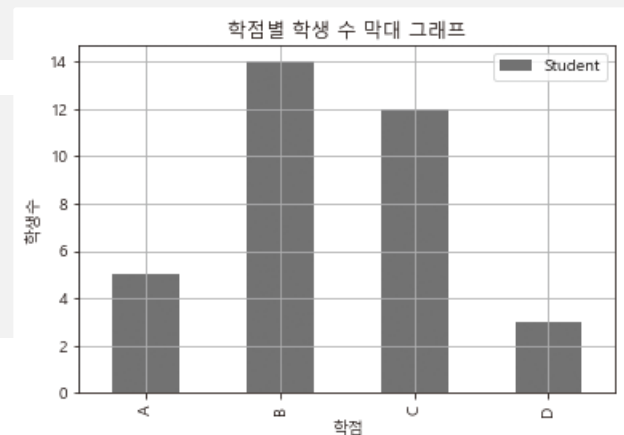
```
grade_num = [5, 14, 12, 3]  
students = ['A', 'B', 'C', 'D']
```

```
df_grade = pd.DataFrame(grade_num, index=students, columns = ['Student'])  
df_grade
```

Out:

	Student
A	5
B	14
C	12
D	3

```
In: 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()
```



pandas로 그래프 그리기

- pandas의 히스토그램

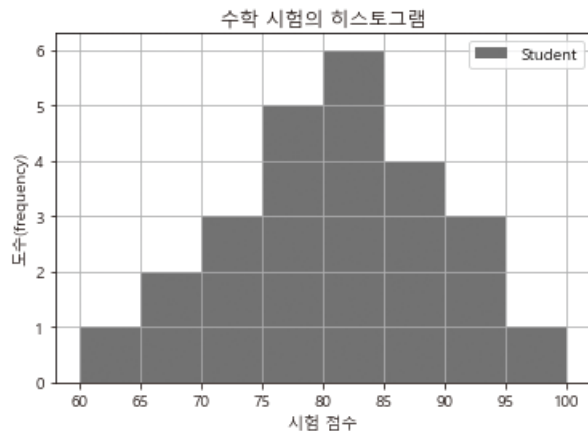
```
In: import matplotlib.pyplot as plt  
import pandas as pd
```

```
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("시험 점수")  
math_hist.set_ylabel("도수(frequency)")  
math_hist.set_title("수학 시험의 히스토그램")
```

```
plt.show()
```



pandas로 그래프 그리기

- pandas의 파이그래프

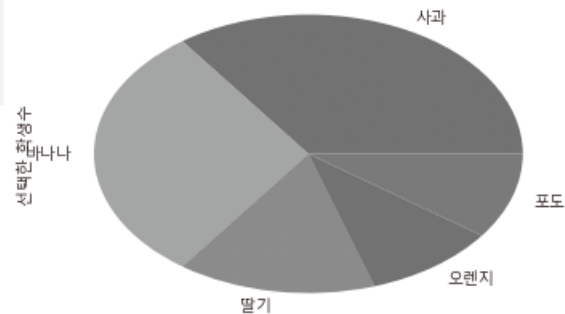
```
In: import matplotlib.pyplot as plt  
import pandas as pd
```

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

```
In: df_fruit.plot.pie()  
plt.show()
```

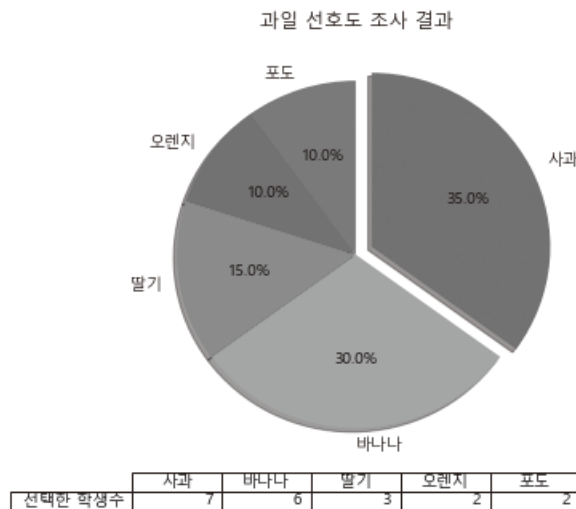


pandas로 그래프 그리기

- pandas의 파이그래프

```
In: explode_value = (0.1, 0, 0, 0, 0)
    fruit_pie = df_fruit.plot.pie(figsize=(5, 5), autopct='%0.1f%%', 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:/myPyCode/figures/saveFigTest3.png', dpi = 200)
plt.show()
```



13



엑셀 파일 다루기

엑셀 파일 읽고 쓰기

- pandas 패키지 활용
- 엑셀 파일의 데이터 읽기

```
df = pd.read_excel('excel_file.xlsx' [, sheet_name = number 혹은 '시트이름',  
index_col = number 혹은 '열이름'])
```

	A	B	C	D	E
1	학생	국어	영어	수학	평균
2	A	80	90	85	85.00
3	B	90	95	95	93.33
4	C	95	70	75	80.00
5	D	70	85	80	78.33
6	E	75	90	85	83.33

```
In: import pandas as pd  
    df = pd.read_excel('C:/myPyCode/data/학생시험성적.xlsx')  
    df
```

Out:

	학생	국어	영어	수학	평균
0	A	80	90	85	85.000000
1	B	90	95	95	93.333333
2	C	95	70	75	80.000000
3	D	70	85	80	78.333333
4	E	75	90	85	83.333333

엑셀 파일 읽고 쓰기

- 엑셀 파일의 데이터 읽기

```
In: pd.read_excel('C:/myPyCode/data/학생시험성적.xlsx', sheet_name = 1)
```

Out:

	학생	과학	사회	역사	평균
0	A	90	95	85	90.000000
1	B	85	90	80	85.000000
2	C	70	80	75	75.000000
3	D	75	90	100	88.333333
4	E	90	80	90	86.666667

```
In: pd.read_excel('C:/myPyCode/data/학생시험성적.xlsx', sheet_name = '2차시험')
```

Out:

	학생	과학	사회	역사	평균
0	A	90	95	85	90.000000
1	B	85	90	80	85.000000
2	C	70	80	75	75.000000
3	D	75	90	100	88.333333
4	E	90	80	90	86.666667

엑셀 파일 읽고 쓰기

- 엑셀 파일의 데이터 읽기

```
In: df = pd.read_excel('C:/myPyCode/data/학생시험성적.xlsx', sheet_name='2차시험', index_col=0)
df
```

Out:

	과학	사회	역사	평균
학생				
A	90	95	85	90.000000
B	85	90	80	85.000000
C	70	80	75	75.000000
D	75	90	100	88.333333
E	90	80	90	86.666667

```
In: df = pd.read_excel('C:/myPyCode/data/학생시험성적.xlsx', sheet_name='2차시험', index_col='학생')
df
```

Out:

	과학	사회	역사	평균
학생				
A	90	95	85	90.000000
B	85	90	80	85.000000
C	70	80	75	75.000000
D	75	90	100	88.333333
E	90	80	90	86.666667

엑셀 파일 읽고 쓰기

- 데이터를 엑셀 파일로 쓰기

```
# (1) pandas의 ExcelWriter 객체 생성
```

```
excel_writer = pd.ExcelWriter('excel_output.xlsx', engine='xlsxwriter')
```

```
# (2) DataFrame 데이터를 지정된 엑셀 시트(Sheet)에 쓰기
```

```
df1.to_excel(excel_writer[, index=True 혹은 False, sheet_name='시트이름1'])
```

```
df2.to_excel(excel_writer[, index=True 혹은 False, sheet_name='시트이름2'])
```

```
# (3) ExcelWriter 객체를 닫고, 지정된 엑셀 파일 생성
```

```
excel_writer.save()
```

엑셀 파일 읽고 쓰기

- 데이터를 엑셀 파일로 쓰기

```
In: import pandas as pd
    excel_exam_data1 = {'학생': ['A', 'B', 'C', 'D', 'E', 'F'],
                          '국어': [80, 90, 95, 70, 75, 85],
                          '영어': [90, 95, 70, 85, 90, 95],
                          '수학': [85, 95, 75, 80, 85, 100]}
    df1 = pd.DataFrame(excel_exam_data1, columns=['학생', '국어', '영어', '수학'])
    df1
```

Out:

	학생	국어	영어	수학
0	A	80	90	85
1	B	90	95	95
2	C	95	70	75
3	D	70	85	80
4	E	75	90	85
5	F	85	95	100

엑셀 파일 읽고 쓰기

- 데이터를 엑셀 파일로 쓰기

```
In: excel_writer = pd.ExcelWriter('C:/myPyCode/data/학생시험성적2.xlsx', engine='xlsxwriter')  
    df1.to_excel(excel_writer, index=False)  
    excel_writer.save()
```

	A	B	C	D
1	학생	국어	영어	수학
2	A	80	90	85
3	B	90	95	95
4	C	95	70	75
5	D	70	85	80
6	E	75	90	85
7	F	85	95	100

```
In: excel_writer2 = pd.ExcelWriter('C:/myPyCode/data/학생시험성적3.xlsx', engine='xlsxwriter')  
    df1.to_excel(excel_writer2, index=False, sheet_name='중간고사')  
    excel_writer2.save()
```

	A	B	C	D
1	학생	국어	영어	수학
2	A	80	90	85
3	B	90	95	95
4	C	95	70	75
5	D	70	85	80
6	E	75	90	85
7	F	85	95	100

엑셀 파일 읽고 쓰기

- 데이터를 엑셀 파일로 쓰기

```
In: import pandas as pd
    excel_exam_data2 = {'학생': ['A', 'B', 'C', 'D', 'E', 'F'],
                        '국어': [85, 95, 75, 80, 85, 100],
                        '영어': [80, 90, 95, 70, 75, 85],
                        '수학': [90, 95, 70, 85, 90, 95]}
    df2 = pd.DataFrame(excel_exam_data2, columns=['학생', '국어', '영어', '수학'])
    df2
```

Out:

	학생	국어	영어	수학
0	A	85	80	90
1	B	95	90	95
2	C	75	95	70
3	D	80	70	85
4	E	85	75	90
5	F	100	85	95

엑셀 파일 읽고 쓰기

- 데이터를 엑셀 파일로 쓰기

```
In: excel_writer3 = pd.ExcelWriter('C:/myPyCode/data/학생시험성적4.xlsx', engine='xlsxwriter')  
    df1.to_excel(excel_writer3, index=False, sheet_name='중간고사')  
    df2.to_excel(excel_writer3, index=False, sheet_name='기말고사')  
    excel_writer3.save()
```

	A	B	C	D
1	학생	국어	영어	수학
2	A	85	80	90
3	B	95	90	95
4	C	75	95	70
5	D	80	70	85
6	E	85	75	90
7	F	100	85	95
◀ ▶		중간고사	기말고사	⊕

엑셀 파일 읽고 쓰기

- 엑셀 파일 통합하기

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계						
3	구두						
4	핸드백						

제품의 분기별 판매량을 조사하기 위한 예제 파일

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계	A	가	198	123	120	137
3	구두	A	가	273	241	296	217
4	핸드백	A	가	385	316	355	331

Andy사원의 판매량('담당자별_판매량_Andy사원.xlsx')

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계	B	나	154	108	155	114
3	구두	B	나	200	223	213	202
4	핸드백	B	나	350	340	377	392

Becky사원의 판매량('담당자별_판매량_Becky사원.xlsx')

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계	C	다	168	102	149	174
3	구두	C	다	231	279	277	292
4	핸드백	C	다	365	383	308	323

Chris사원의 판매량('담당자별_판매량_Chris사원.xlsx')

엑셀 파일 읽고 쓰기

- 엑셀 파일 통합하기

```
In: excel_data_files = ['C:/myPyCode/data/담당자별_판매량_Andy사원.xlsx',  
                        'C:/myPyCode/data/담당자별_판매량_Becky사원.xlsx',  
                        'C:/myPyCode/data/담당자별_판매량_Chris사원.xlsx']
```

```
In: total_data = pd.DataFrame()
```

```
In: import pandas as pd  
    for f in excel_data_files:  
        df = pd.read_excel(f)  
        total_data = total_data.append(df)  
    total_data
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331
0	시계	B	나	154	108	155	114
1	구두	B	나	200	223	213	202
2	핸드백	B	나	350	340	377	392
0	시계	C	다	168	102	149	174
1	구두	C	다	231	279	277	292
2	핸드백	C	다	365	383	308	323

엑셀 파일 읽고 쓰기

- 엑셀 파일 통합하기

```
In: import pandas as pd
    total_data = pd.DataFrame()

    for f in excel_data_files:
        df = pd.read_excel(f)
        total_data = total_data.append(df, ignore_index=True)
    total_data
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331
3	시계	B	나	154	108	155	114
4	구두	B	나	200	223	213	202
5	핸드백	B	나	350	340	377	392
6	시계	C	다	168	102	149	174
7	구두	C	다	231	279	277	292
8	핸드백	C	다	365	383	308	323

엑셀 파일 읽고 쓰기

- 엑셀 파일 통합하기

```
In: import glob
import pandas as pd

excel_data_files1 = glob.glob("C:/myPyCode/data/담당자별_판매량_*.xlsx")
total_data1 = pd.DataFrame()

for f in excel_data_files1:
    df = pd.read_excel(f)
    total_data1 = total_data1.append(df, ignore_index=True)
```

total_data1

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331
3	시계	B	나	154	108	155	114
4	구두	B	나	200	223	213	202
5	핸드백	B	나	350	340	377	392
6	시계	C	다	168	102	149	174
7	구두	C	다	231	279	277	292
8	핸드백	C	다	365	383	308	323

엑셀 파일 읽고 쓰기

- 통합 결과를 엑셀 파일로 저장하기

```
In: import glob
import pandas as pd

excel_file_name = 'C:/myPyCode/data/담당자별_판매량_통합.xlsx'

excel_total_file_writer = pd.ExcelWriter(excel_file_name, engine='xlsxwriter')
total_data1.to_excel(excel_total_file_writer, index=False, sheet_name='담당자별_판매량_통합')
excel_total_file_writer.save()
```

```
glob.glob(excel_file_name)
```

Out: ['C:/myPyCode/data/담당자별_판매량_통합.xlsx']

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계	A	가	198	123	120	137
3	구두	A	가	273	241	296	217
4	핸드백	A	가	385	316	355	331
5	시계	B	나	154	108	155	114
6	구두	B	나	200	223	213	202
7	핸드백	B	나	350	340	377	392
8	시계	C	다	168	102	149	174
9	구두	C	다	231	279	277	292
10	핸드백	C	다	365	383	308	323

엑셀 파일로 읽어온 데이터 다루기

- 데이터를 추가하고 변경하기

```
import pandas as pd
```

```
df = pd.read_excel('excel_file.xlsx')  
df.loc[index_name, column_name] = value
```

```
In: import pandas as pd
```

```
df = pd.read_excel('C:/myPyCode/data/담당자별_판매량_Andy사원.xlsx')  
df
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331

```
In: df.loc[2, '4분기'] = 0
```

```
df
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	0

엑셀 파일로 읽어온 데이터 다루기

- 데이터를 추가하고 변경하기

```
In: df.loc[3, '제품명'] = '벨트'  
    df.loc[3, '담당자'] = 'A'  
    df.loc[3, '지역'] = '가'  
    df.loc[3, '1분기'] = 100  
    df.loc[3, '2분기'] = 150  
    df.loc[3, '3분기'] = 200  
    df.loc[3, '4분기'] = 250  
df
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198.0	123.0	120.0	137.0
1	구두	A	가	273.0	241.0	296.0	217.0
2	핸드백	A	가	385.0	316.0	355.0	0.0
3	벨트	A	가	100.0	150.0	200.0	250.0

엑셀 파일로 읽어온 데이터 다루기

- 데이터를 추가하고 변경하기

```
df[column_name] = value
```

```
In: df['담당자'] = 'Andy'
```

df

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	Andy	가	198.0	123.0	120.0	137.0
1	구두	Andy	가	273.0	241.0	296.0	217.0
2	핸드백	Andy	가	385.0	316.0	355.0	0.0
3	벨트	Andy	가	100.0	150.0	200.0	250.0

```
In: excel_file_name = 'C:/myPyCode/data/담당자별_판매량_Andy사원_new.xlsx'
```

```
new_excel_file = pd.ExcelWriter(excel_file_name, engine='xlsxwriter')  
df.to_excel(new_excel_file, index=False)  
new_excel_file.save()
```

```
glob.glob(excel_file_name)
```

Out: ['C:/myPyCode/data/담당자별_판매량_Andy사원_new.xlsx']

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계	Andy	가	198	123	120	137
3	구두	Andy	가	273	241	296	217
4	핸드백	Andy	가	385	316	355	0
5	벨트	Andy	가	100	150	200	250

엑셀 파일로 읽어온 데이터 다루기

- 여러 개의 엑셀 파일에서 데이터 수정하기

```
import re  
re.sub(pattern, repl, string)
```

```
In: import re
```

```
file_name = 'C:/myPyCode/data/담당자별_판매량_Andy사원.xlsx'
```

```
new_file_name = re.sub(".xlsx", "2.xlsx", file_name)
```

```
new_file_name
```

```
Out: 'C:/myPyCode/data/담당자별_판매량_Andy사원2.xlsx'
```

```
In: import glob
```

```
import re
```

```
import pandas as pd
```

```
# 원하는 문자열이 포함된 파일을 검색해 리스트를 할당한다.
```

```
excel_data_files1 = glob.glob("C:/myPyCode/data/담당자별_판매량_*사원.xlsx")
```

```
# 리스트에 있는 엑셀 파일만큼 반복 수행한다.
```

```
for f in excel_data_files1:
```

```
    # 엑셀 파일에서 DataFrame 형식으로 데이터를 가져온다.
```

```
    df = pd.read_excel(f)
```

엑셀 파일로 읽어온 데이터 다루기

- 여러 개의 엑셀 파일에서 데이터 수정하기

```
# 특정 열의 값을 변경한다.
```

```
if(df.loc[1, '담당자']=='A'):
```

```
    df['담당자']='Andy'
```

```
elif(df.loc[1, '담당자']=='B'):
```

```
    df['담당자']='Becky'
```

```
elif(df.loc[1, '담당자']=='C'):
```

```
    df['담당자']='Chris'
```

```
# 엑셀 파일 이름에서 지정된 문자열 패턴을 찾아서 파일명을 변경한다.
```

```
f_new = re.sub(".xlsx", "2.xlsx", f)
```

```
print(f_new)
```

```
# 수정된 데이터를 새로운 이름의 엑셀 파일로 저장한다.
```

```
new_excel_file = pd.ExcelWriter(f_new, engine='xlsxwriter')
```

```
df.to_excel(new_excel_file, index=False)
```

```
new_excel_file.save()
```

```
Out: C:/myPyCode/dataW담당자별_판매량_Andy사원2.xlsx
```

```
C:/myPyCode/dataW담당자별_판매량_Becky사원2.xlsx
```

```
C:/myPyCode/dataW담당자별_판매량_Chris사원2.xlsx
```

엑셀 파일로 읽어온 데이터 다루기

- 여러 개의 엑셀 파일에서 데이터 수정하기

```
In: glob.glob("C:/myPyCode/data/담당자별_판매량_*.xlsx")
```

```
Out: ['C:/myPyCode/dataWW담당자별_판매량_Andy사원2.xlsx',  
      'C:/myPyCode/dataWW담당자별_판매량_Becky사원2.xlsx',  
      'C:/myPyCode/dataWW담당자별_판매량_Chris사원2.xlsx']
```

- 엑셀의 필터 기능 수행하기



	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계	A	가	198	123	120	137
3	구두	A	가	273	241	296	217
4	핸드백	A	가	385	316	355	331
5	시계	B	나	154	108	155	114
6	구두	B	나	200	223	213	202
7	핸드백	B	나	350	340	377	392
8	시계	C	다	168	102	149	174
9	구두	C	다	231	279	277	292
10	핸드백	C	다	365	383	308	323

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	시계	A	가	198	123	120	137
3	구두	A	가	273	241	296	217
4	핸드백	A	가	385	316	355	331
5	시계	B	나	154	108	155	114
6	구두	B	나	200	223	213	202
7	핸드백	B	나	350	340	377	392
8	시계	C	다	168	102	149	174
9	구두	C	다	231	279	277	292
10	핸드백	C	다	365	383	308	323

엑셀에서 '제품명' 셀의 화살표 클릭

엑셀에서 셀 지정 후 [필터] 아이콘 클릭

엑셀 파일로 읽어온 데이터 다루기

- 엑셀의 필터 기능 수행하기

	A	B	C	D	E	F	G
1	제품명	담당지	지역	1분기	2분기	3분기	4분기
공↓	텍스트 오름차순 정렬(S)			198	123	120	137
하↓	텍스트 내림차순 정렬(O)			273	241	296	217
	색 기준 정렬(I)			385	316	355	331
	"제품명"에서 필터 해제(O)			154	108	155	114
	색 기준 필터(I)			200	223	213	202
	텍스트 필터(F)			350	340	377	392
	검색			168	102	149	174
	<input checked="" type="checkbox"/> (모두 선택)			231	279	277	292
	<input type="checkbox"/> 구두			365	383	308	323
	<input type="checkbox"/> 시계						
	<input checked="" type="checkbox"/> 핸드백						
	확인	취소					

여러 항목 중 특정 항목만 선택

	A	B	C	D	E	F	G
1	제품명	담당지	지역	1분기	2분기	3분기	4분기
4	핸드백	A	가	385	316	355	331
7	핸드백	B	나	350	340	377	392
10	핸드백	C	다	365	383	308	323

엑셀에서 필터 기능으로 특정 항목만 선택한 결과

엑셀 파일로 읽어온 데이터 다루기

- 엑셀의 필터 기능 수행하기

```
In: import pandas as pd
    df = pd.read_excel('C:/myPyCode/data/담당자별_판매량_통합.xlsx')
    df
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331
3	시계	B	나	154	108	155	114
4	구두	B	나	200	223	213	202
5	핸드백	B	나	350	340	377	392
6	시계	C	다	168	102	149	174
7	구두	C	다	231	279	277	292
8	핸드백	C	다	365	383	308	323

엑셀 파일로 읽어온 데이터 다루기

- 엑셀의 필터 기능 수행하기

```
In: df['제품명']
```

```
Out: 0    시계
```

```
1    구두
```

```
2  핸드백
```

```
3    시계
```

```
4    구두
```

```
5  핸드백
```

```
6    시계
```

```
7    구두
```

```
8  핸드백
```

```
Name: 제품명, dtype: object
```

```
In: df['제품명'] == '핸드백'
```

```
Out: 0    False
```

```
1    False
```

```
2     True
```

```
3    False
```

```
4    False
```

```
5     True
```

```
6    False
```

```
7    False
```

```
8     True
```

```
Name: 제품명, dtype: bool
```

엑셀 파일로 읽어온 데이터 다루기

- 엑셀의 필터 기능 수행하기

```
In: handbag = df[df['제품명'] == '핸드백']
```

```
handbag
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	핸드백	A	가	385	316	355	331
5	핸드백	B	나	350	340	377	392
8	핸드백	C	다	365	383	308	323

```
DataFrame_data.isin(values)
```

```
In: import pandas as pd
```

```
df = pd.read_excel('C:/myPyCode/data/담당자별_판매량_통합.xlsx')
```

```
handbag1 = df[df['제품명'].isin(['핸드백'])]
```

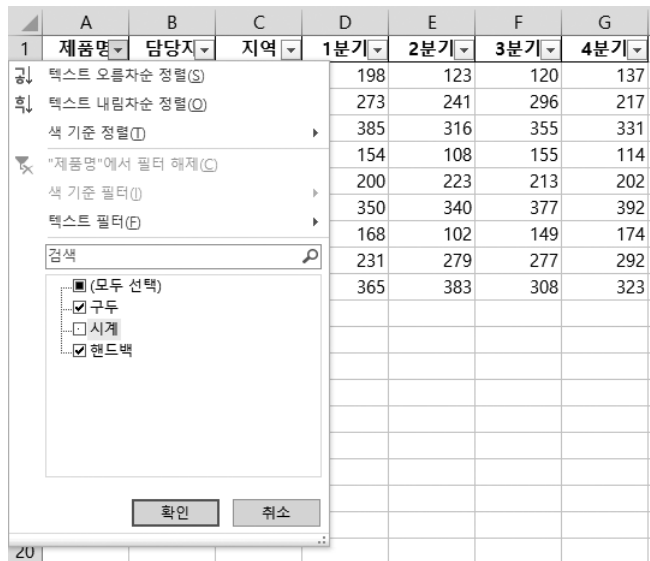
```
handbag1
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	핸드백	A	가	385	316	355	331
5	핸드백	B	나	350	340	377	392
8	핸드백	C	다	365	383	308	323

엑셀 파일로 읽어온 데이터 다루기

- 엑셀의 필터 기능 수행하기



	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	구두	오름자순 정렬(S)		198	123	120	137
3	구두	내림자순 정렬(Q)		273	241	296	217
4	구두	색 기준 정렬(I)		385	316	355	331
5	구두	"제품명"에서 필터 해제(C)		154	108	155	114
6	구두	색 기준 필터(I)		200	223	213	202
7	구두	색 기준 필터(I)		350	340	377	392
8	구두	색 기준 필터(I)		168	102	149	174
9	구두	색 기준 필터(I)		231	279	277	292
10	구두	색 기준 필터(I)		365	383	308	323

엑셀의 필터 기능으로 여러 개의 항목을 선택

	A	B	C	D	E	F	G
1	제품명	담당자	지역	1분기	2분기	3분기	4분기
3	구두	A	가	273	241	296	217
4	핸드백	A	가	385	316	355	331
6	구두	B	나	200	223	213	202
7	핸드백	B	나	350	340	377	392
9	구두	C	다	231	279	277	292
10	핸드백	C	다	365	383	308	323

엑셀의 필터 기능으로 여러 개의 항목을 선택한 결과

엑셀 파일로 읽어온 데이터 다루기

- 엑셀의 필터 기능 수행하기

```
In: df[(df['제품명']=='구두') | (df['제품명']=='핸드백')]
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331
4	구두	B	나	200	223	213	202
5	핸드백	B	나	350	340	377	392
7	구두	C	다	231	279	277	292
8	핸드백	C	다	365	383	308	323

```
In: df[df['제품명'].isin(['구두', '핸드백'])]
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331
4	구두	B	나	200	223	213	202
5	핸드백	B	나	350	340	377	392
7	구두	C	다	231	279	277	292
8	핸드백	C	다	365	383	308	323

엑셀 파일로 읽어온 데이터 다루기

- 조건을 설정해 원하는 행만 선택하기

```
In: df[(df['3분기'] >= 250)]
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331
5	핸드백	B	나	350	340	377	392
7	구두	C	다	231	279	277	292
8	핸드백	C	다	365	383	308	323

```
In: df[(df['제품명'] == '핸드백') & (df['3분기'] >= 350)]
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	핸드백	A	가	385	316	355	331
5	핸드백	B	나	350	340	377	392

엑셀 파일로 읽어온 데이터 다루기

- 원하는 열만 선택하기

	A	B	C	F	G
1	제품명	담당지	지역	분기	4분기
2	시계	A	가	120	137
3	구두	A	가	296	217
4	핸드백	A	가	355	331
5	시계	B	나	155	114
6	구두	B	나	213	202
7	핸드백	B	나	377	392
8	시계	C	다	149	174
9	구두	C	다	277	292
10	핸드백	C	다	308	323
11					
12					
13					

'숨기기' 기능으로 열 숨기기

	A	D	E	F	G
1	제품명	1분기	2분기	3분기	4분기
2	시계	198	123	120	137
3	구두	273	241	296	217
4	핸드백	385	316	355	331
5	시계	154	108	155	114
6	구두	200	223	213	202
7	핸드백	350	340	377	392
8	시계	168	102	149	174
9	구두	231	279	277	292
10	핸드백	365	383	308	323

엑셀의 '숨기기' 기능을 적용한 결과

엑셀 파일로 읽어온 데이터 다루기

- 원하는 열만 선택하기

```
In: import pandas as pd
    df = pd.read_excel('C:/myPyCode/data/담당자별_판매량_Andy사원.xlsx')
    df
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
1	구두	A	가	273	241	296	217
2	핸드백	A	가	385	316	355	331

```
In: df[['제품명', '1분기', '2분기', '3분기', '4분기']]
```

Out:

	제품명	1분기	2분기	3분기	4분기
0	시계	198	123	120	137
1	구두	273	241	296	217
2	핸드백	385	316	355	331

엑셀 파일로 읽어온 데이터 다루기

- 원하는 열만 선택하기

```
DataFrame_data.iloc[row_num, col_num]
```

```
In: df.iloc[:, [0,3,4,5,6]]
```

Out:

	제품명	1분기	2분기	3분기	4분기
0	시계	198	123	120	137
1	구두	273	241	296	217
2	핸드백	385	316	355	331

```
In: df.iloc[[0,2],:]
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
0	시계	A	가	198	123	120	137
2	핸드백	A	가	385	316	355	331

엑셀 파일로 읽어온 데이터 다루기

- 엑셀 데이터 계산하기
 - 행 데이터의 합계 구하기



	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	A	가	385	316	355	331	
7	핸드백	B	나	350	340	377	392	
10	핸드백	C	다	365	383	308	323	

	A	B	C	D	E	F	G	H	I
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량	
4	핸드백	A	가	385	316	355	331		
7	핸드백	B	나	350	340	377	392		
10	핸드백	C	다	365	383	308	323		

	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	A	가	385	316	355	331	
7	핸드백	B	나	350	340	377	392	
10	핸드백	C	다	365	383	308	323	

	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	A	가	385	316	355	331	1387
7	핸드백	B	나	350	340	377	392	
10	핸드백	C	다	365	383	308	323	

	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	A	가	385	316	355	331	1387
7	핸드백	B	나	350	340	377	392	
10	핸드백	C	다	365	383	308	323	

	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	A	가	385	316	355	331	1387
7	핸드백	B	나	350	340	377	392	
10	핸드백	C	다	365	383	308	323	

	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	A	가	385	316	355	331	1387
7	핸드백	B	나	350	340	377	392	1459
10	핸드백	C	다	365	383	308	323	1379

엑셀 파일로 읽어온 데이터 다루기

- 엑셀 데이터 계산하기
 - 행 데이터의 합계 구하기

```
In: import pandas as pd
    df = pd.read_excel('C:/myPyCode/data/담당자별_판매량_통합.xlsx')
    handbag = df[(df['제품명'] == '핸드백')]
    handbag
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기
2	핸드백	A	가	385	316	355	331
5	핸드백	B	나	350	340	377	392
8	핸드백	C	다	365	383	308	323

```
DataFrame_data.sum([axis = 0(기본) or 1])
```

```
In: handbag.sum(axis=1)
```

```
Out: 2    1387
      5    1459
      8    1379
      dtype: int64
```

엑셀 파일로 읽어온 데이터 다루기

– 행 데이터의 합계 구하기

```
In: handbag_sum = pd.DataFrame(handbag.sum(axis=1), columns = ['연간판매량'])
```

```
handbag_sum
```

Out:

연간판매량

2 1387

5 1459

8 1379

```
In: handbag_total = handbag.join(handbag_sum)
```

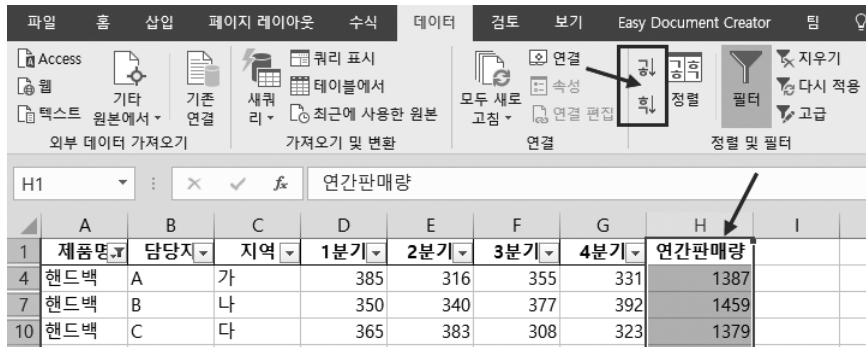
```
handbag_total
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
2	핸드백	A	가	385	316	355	331	1387
5	핸드백	B	나	350	340	377	392	1459
8	핸드백	C	다	365	383	308	323	1379

엑셀 파일로 읽어온 데이터 다루기

– 행 데이터의 합계 구하기



	A	B	C	D	E	F	G	H	I
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량	
4	핸드백	A	가	385	316	355	331	1387	
7	핸드백	B	나	350	340	377	392	1459	
10	핸드백	C	다	365	383	308	323	1379	

	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	C	다	365	383	308	323	1379
7	핸드백	A	가	385	316	355	331	1387
10	핸드백	B	나	350	340	377	392	1459

`DataFrame_data.sort_values(by [, axis=0(기본) or 1, ascending=True(기본) or False])`

In: `handbag_total.sort_values(by='연간판매량', ascending=True)`

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
8	핸드백	C	다	365	383	308	323	1379
2	핸드백	A	가	385	316	355	331	1387
5	핸드백	B	나	350	340	377	392	1459

엑셀 파일로 읽어온 데이터 다루기

– 행 데이터의 합계 구하기

In: handbag_total.sort_values(by='연간판매량', ascending=False)

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
5	핸드백	B	나	350	340	377	392	1459
2	핸드백	A	가	385	316	355	331	1387
8	핸드백	C	다	365	383	308	323	1379

– 열 데이터의 합계 구하기

SUM		X		✓		fx		=SUBTOTAL(9,D2:D10)	
	A	B	C	D	E	F	G	H	
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량	
4	핸드백	A	가	385	316	355	331	1387	
7	핸드백	B	나	350	340	377	392	1459	
10	핸드백	C	다	365	383	308	323	1379	
11				=SUBTOTAL(9,D2:D10)					
12				SUBTOTAL(function_num, ref1, [ref2], ...)					

D11		:	✕	✓	<i>f_x</i>	=SUBTOTAL(9,D2:D10)		
	A	B	C	D	E	F	G	H
1	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
4	핸드백	A	가	385	316	355	331	1387
7	핸드백	B	나	350	340	377	392	1459
10	핸드백	C	다	365	383	308	323	1379
11				1100	1039	1040	1046	4225

엑셀 파일로 읽어온 데이터 다루기

– 열 데이터의 합계 구하기

```
In: handbag_total.sum()
```

```
Out: 제품명      핸드백핸드백핸드백
```

```
    담당자      ABC
```

```
    지역      가나다
```

```
    1분기      1100
```

```
    2분기      1039
```

```
    3분기      1040
```

```
    4분기      1046
```

```
    연간판매량  4225
```

```
dtype: object
```

```
In: handbag_sum2 = pd.DataFrame(handbag_total.sum(), columns=['합계'])
```

```
    handbag_sum2
```

```
Out:
```

```
    합계
```

```
    제품명      핸드백핸드백핸드백
```

```
    담당자      ABC
```

```
    지역      가나다
```

```
    1분기      1100
```

```
    2분기      1039
```

```
    3분기      1040
```

```
    4분기      1046
```

```
    연간판매량  4225
```


엑셀 파일로 읽어온 데이터 다루기

– 열 데이터의 합계 구하기

```
In: handbag_total2 = handbag_total.append(handbag_sum2.T)
    handbag_total2
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
2	핸드백	A	가	385	316	355	331	1387
5	핸드백	B	나	350	340	377	392	1459
8	핸드백	C	다	365	383	308	323	1379
합계	핸드백	ABC	가나다	1100	1039	1040	1046	4225

```
In: handbag_total2.loc['합계', '제품명'] = '핸드백'
    handbag_total2.loc['합계', '담당자'] = '전체'
    handbag_total2.loc['합계', '지역'] = '전체'
    handbag_total2
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
2	핸드백	A	가	385	316	355	331	1387
5	핸드백	B	나	350	340	377	392	1459
8	핸드백	C	다	365	383	308	323	1379
합계	핸드백	전체	전체	1100	1039	1040	1046	4225

엑셀 파일로 읽어온 데이터 다루기

– 열 데이터의 합계 구하기

```
In: import pandas as pd
```

```
# 엑셀 파일을 pandas의 DataFrame 형식으로 읽어온다.
```

```
df = pd.read_excel('C:/myPyCode/data/담당자별_판매량_통합.xlsx')
```

```
# 제품명 열에서 핸드백이 있는 행만 선택한다.
```

```
product_name = '핸드백'
```

```
handbag = df[(df['제품명']== product_name)]
```

```
# 행별로 합계를 구하고 마지막 열 다음에 추가한다.
```

```
handbag_sum = pd.DataFrame(handbag.sum(axis=1), columns = ['연간판매량'])
```

```
handbag_total = handbag.join(handbag_sum)
```

```
# 열별로 합해 분기별 합계와 연간판매량 합계를 구하고 마지막 행 다음에 추가한다.
```

```
handbag_sum2 = pd.DataFrame(handbag_total.sum(), columns=['합계'])
```

```
handbag_total2 = handbag_total.append(handbag_sum2.T)
```

```
# 지정된 항목의 문자열을 변경한다.
```

```
handbag_total2.loc['합계', '제품명'] = product_name
```

```
handbag_total2.loc['합계', '담당자'] = '전체'
```

```
handbag_total2.loc['합계', '지역'] = '전체'
```

엑셀 파일로 읽어온 데이터 다루기

– 열 데이터의 합계 구하기

```
# 결과를 확인한다.
```

```
handbag_total2
```

Out:

	제품명	담당자	지역	1분기	2분기	3분기	4분기	연간판매량
2	핸드백	A	가	385	316	355	331	1387
5	핸드백	B	나	350	340	377	392	1459
8	핸드백	C	다	365	383	308	323	1379
합계	핸드백	전체	전체	1100	1039	1040	1046	4225

엑셀 데이터의 시각화

- 그래프를 엑셀 파일에 넣기

```
# (1) pandas의 ExcelWriter 객체 생성
```

```
excel_writer = pd.ExcelWriter('excel_output.xlsx', engine='xlsxwriter')
```

```
# (2) DataFrame 데이터를 지정된 엑셀 시트(Sheet)에 쓰기
```

```
df.to_excel(excel_writer, index=False 혹은 True, sheet_name='시트이름')
```

```
# (3) ExcelWriter 객체에서 워크시트(worksheet) 객체 생성
```

```
worksheet = excel_writer.sheets['시트이름']
```

```
# (4) 워크시트에 차트가 들어갈 위치를 지정해 이미지 넣기
```

```
worksheet.insert_image('셀위치', image_file [, {'x_scale': x_scale_num, 'y_scale': y_scale_num}])  
혹은
```

```
worksheet.insert_image(row_num, col_num, image_file [, {'x_scale': x_scale_num, 'y_scale':  
y_scale_num}])
```

```
# (5) ExcelWriter 객체를 닫고 엑셀 파일 출력
```

```
excel_writer.save()
```

엑셀 데이터의 시각화

- 그래프를 엑셀 파일에 넣기

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

```
sales = {'시간': [9, 10, 11, 12, 13, 14, 15],
        '제품1': [10, 15, 12, 11, 12, 14, 13],
        '제품2': [9, 11, 14, 12, 13, 10, 12]}
df = pd.DataFrame(sales, index = sales['시간'], columns = ['제품1', '제품2'])
df.index.name = '시간' #index 라벨 추가
```

df

Out:

	제품1	제품2
시간		
9	10	9
10	15	11
11	12	14
12	11	12
13	12	13
14	14	10
15	13	12

엑셀 데이터의 시각화

- 그래프를 엑셀 파일에 넣기

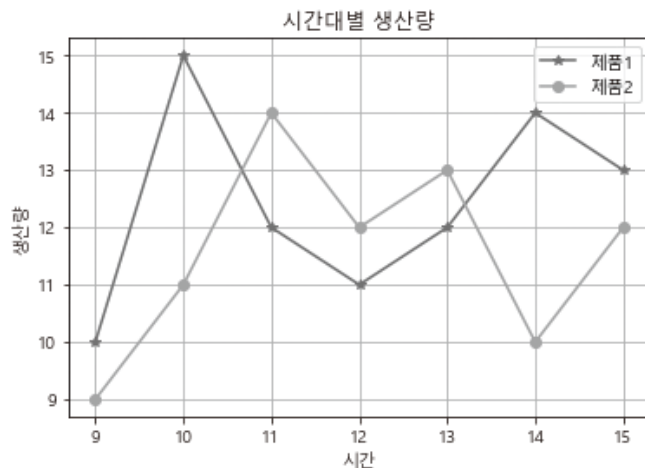
```
In: import matplotlib
import pandas as pd

matplotlib.rcParams['font.family'] = 'Malgun Gothic' # '맑은 고딕'으로 설정
matplotlib.rcParams['axes.unicode_minus'] = False

product_plot = df.plot(grid = True, style = ['-*', '-o'], title='시간대별 생산량')
product_plot.set_ylabel("생산량")

image_file = 'C:/myPyCode/figures/fig_for_excel1.png' # 이미지 파일 경로 및 이름
plt.savefig(image_file, dpi = 400) # 그래프를 이미지 파일로 저장

plt.show()
```



엑셀 데이터의 시각화

- 그래프를 엑셀 파일에 넣기

```
In: import pandas as pd
```

```
# (1) pandas의 ExcelWriter 객체 생성
```

```
excel_file = 'C:/myPyCode/data/data_image_to_excel.xlsx'  
excel_writer = pd.ExcelWriter(excel_file, engine='xlsxwriter')
```

```
# (2) DataFrame 데이터를 지정된 엑셀 시트(Sheet)에 쓰기
```

```
df.to_excel(excel_writer, index=True, sheet_name='Sheet1')
```

```
# (3) ExcelWriter 객체에서 워크시트(worksheet) 객체 생성
```

```
worksheet = excel_writer.sheets['Sheet1']
```

```
# (4) 워크시트에 차트가 들어갈 위치를 지정해 이미지 넣기
```

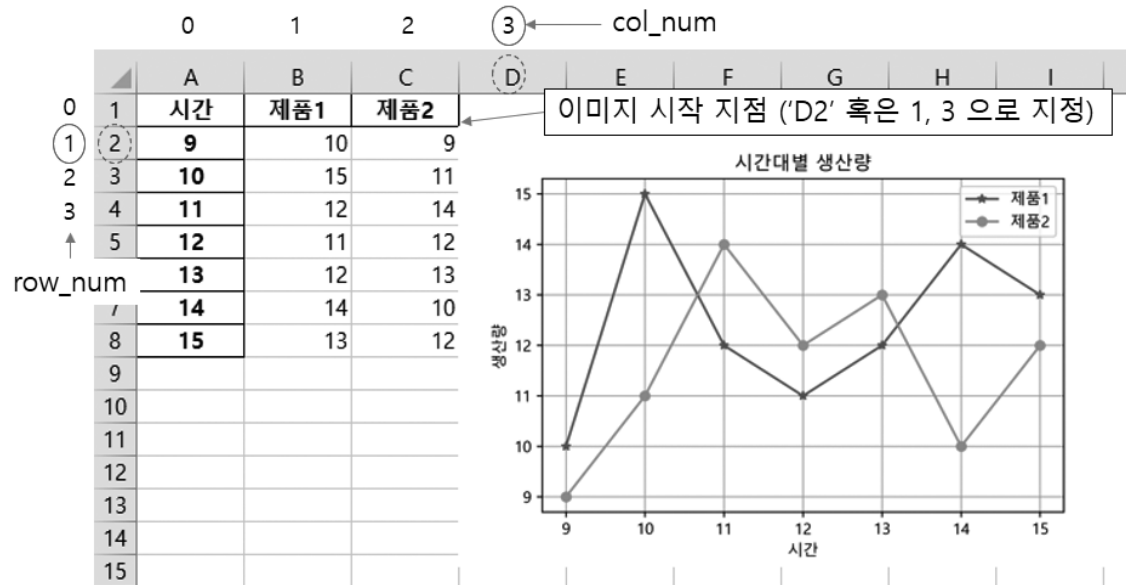
```
worksheet.insert_image('D2', image_file, {'x_scale': 0.7, 'y_scale': 0.7})  
# worksheet.insert_image(1, 3, image_file, {'x_scale': 0.7, 'y_scale': 0.7})
```

```
# (5) ExcelWriter 객체를 닫고 엑셀 파일 출력
```

```
excel_writer.save()
```

엑셀 데이터의 시각화

- 그래프를 엑셀 파일에 넣기



엑셀 데이터의 시각화

- 엑셀 차트 만들기

```
# (1) pandas의 ExcelWriter 객체 생성
excel_writer = pd.ExcelWriter('excel_output.xlsx', engine='xlsxwriter')

# (2) DataFrame 데이터를 지정된 엑셀 시트(Sheet)에 쓰기
df.to_excel(excel_writer, index=False 혹은 True, sheet_name='시트이름')

# (3) ExcelWriter 객체에서 워크북(workbook)과 워크시트(worksheet) 객체 생성
workbook = excel_writer.book
worksheet = excel_writer.sheets['시트이름']

# (4) 차트 객체 생성(원하는 차트의 종류 지정)
chart = workbook.add_chart({'type': '차트유형'})

# (5) 차트를 생성하기 위한 데이터값의 범위 지정
chart.add_series({'values': values_range})

# (6) 워크시트에 차트가 들어갈 위치 지정해 차트 넣기
worksheet.insert_chart('셀위치', chart)
혹은
worksheet.insert_chart(row_num, col_num, chart)

# (7) ExcelWriter 객체를 닫고 엑셀 파일을 출력
excel_writer.save()
```

엑셀 데이터의 시각화

- 엑셀 차트 만들기
 - 엑셀에서 그릴 수 있는 차트 유형

지정 가능한 차트 유형	엑셀 차트 유형
area	영역형 차트
bar	가로 막대형 차트
column	세로 막대형 차트
line	꺾은 선형 차트
pie	원형 차트
doughnut	도넛형 차트
scatter	분산형 차트
stock	주식형 차트
radar	방사형 차트

area	영역형 차트
bar	가로 막대형 차트
column	세로 막대형 차트
line	꺾은 선형 차트
pie	원형 차트
doughnut	도넛형 차트
scatter	분산형 차트
stock	주식형 차트
radar	방사형 차트

엑셀 데이터의 시각화

- 엑셀 차트 만들기

In: # (1) pandas의 ExcelWriter 객체 생성

```
excel_chart = pd.ExcelWriter('C:/myPyCode/data/data_chart_in_excel.xlsx', engine='xlsxwriter')
```

(2) DataFrame 데이터를 지정된 엑셀 시트(Sheet)에 쓰기

```
df.to_excel(excel_chart, index=True, sheet_name='Sheet1')
```

(3) ExcelWriter 객체에서 워크북(workbook)과 워크시트(worksheet) 객체 생성

```
workbook = excel_chart.book
```

```
worksheet = excel_chart.sheets['Sheet1']
```

(4) 차트 객체 생성(원하는 차트의 종류 지정)

```
chart = workbook.add_chart({'type': 'line'})
```

(5) 차트 생성을 위한 데이터값의 범위 지정

```
chart.add_series({'values': '=Sheet1!$B$2:$B$8'})
```

```
chart.add_series({'values': '=Sheet1!$C$2:$C$8'})
```

(6) 워크시트에 차트가 들어갈 위치를 지정해 차트 넣기

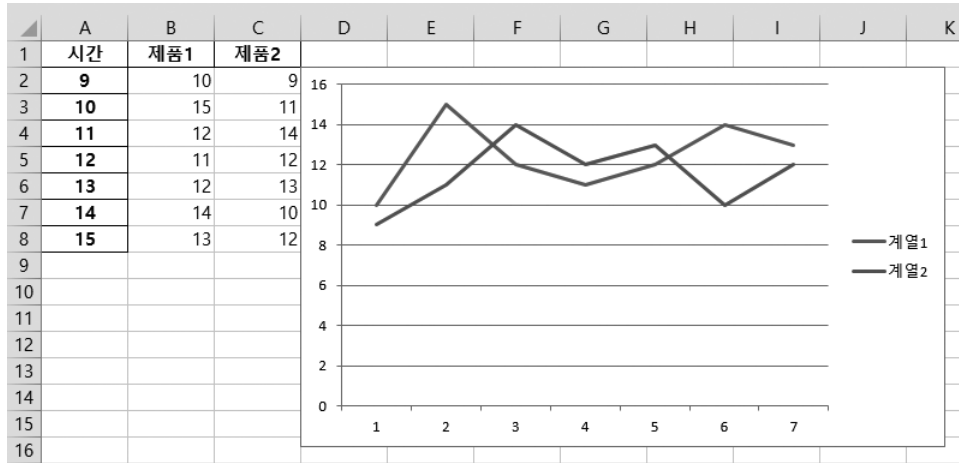
```
worksheet.insert_chart('D2', chart)
```

(7) ExcelWriter 객체를 닫고 엑셀 파일 출력

```
excel_chart.save()
```

엑셀 데이터의 시각화

- 엑셀 차트 만들기

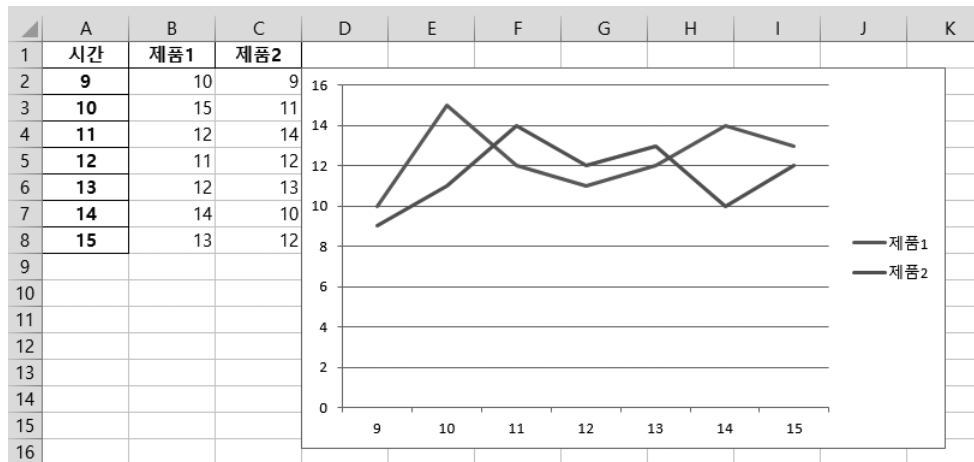


엑셀 데이터의 시각화

- 엑셀 차트 만들기

In: # (5) 차트 생성을 위한 데이터값의 범위 지정

```
chart.add_series({'values': '=Sheet1!$B$2:$B$8',  
                 'categories': '=Sheet1!$A$2:$A$8',  
                 'name': '=Sheet1!$B$1',})  
chart.add_series({'values': '=Sheet1!$C$2:$C$8',  
                 'categories': '=Sheet1!$A$2:$A$8',  
                 'name': '=Sheet1!$C$1',})
```



엑셀 데이터의 시각화

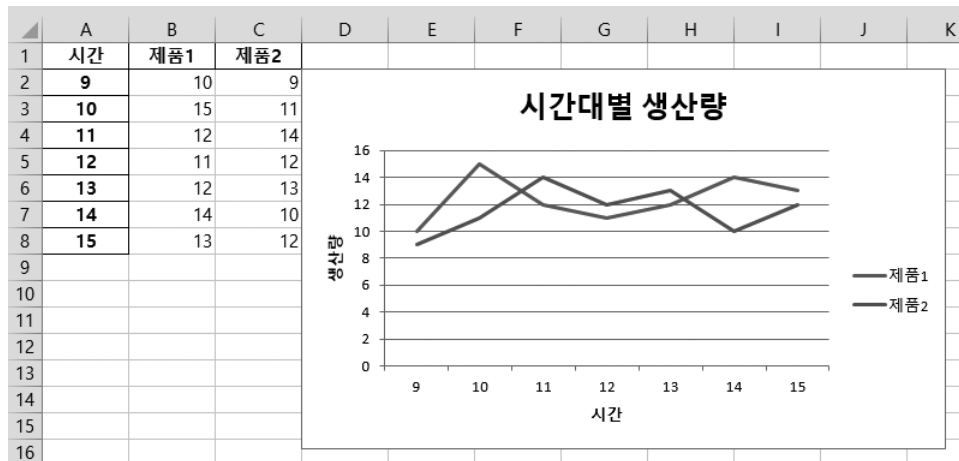
- 엑셀 차트 만들기

In: # (5-1) 엑셀 차트에 x, y축 라벨과 제목 추가

```
chart.set_title({'name': '시간대별 생산량'})
```

```
chart.set_x_axis({'name': '시간'})
```

```
chart.set_y_axis({'name': '생산량'})
```



엑셀 데이터의 시각화

- 엑셀 차트 만들기

In: # (1) pandas의 ExcelWriter 객체 생성

```
excel_chart = pd.ExcelWriter('C:/myPyCode/data/data_chart_in_excel2.xlsx',  
engine='xlsxwriter')
```

(2) DataFrame 데이터를 지정된 엑셀 시트(Sheet)에 쓰기

```
df.to_excel(excel_chart, index=True, sheet_name='Sheet1')
```

(3) ExcelWriter 객체에서 워크북(workbook)과 워크시트(worksheet) 객체 생성

```
workbook = excel_chart.book
```

```
worksheet = excel_chart.sheets['Sheet1']
```

(4) 차트 객체 생성(원하는 차트의 종류 지정)

```
chart = workbook.add_chart({'type': 'line'})
```

(5) 차트 생성을 위한 데이터값의 범위 지정

```
chart.add_series({'values': '=Sheet1!$B$2:$B$8',  
                 'categories': '=Sheet1!$A$2:$A$8',  
                 'name': '=Sheet1!$B$1'})
```

```
chart.add_series({'values': '=Sheet1!$C$2:$C$8',  
                 'categories': '=Sheet1!$A$2:$A$8',  
                 'name': '=Sheet1!$C$1'})
```

엑셀 데이터의 시각화

- 엑셀 차트 만들기

```
# (5-1) 엑셀 차트에 x, y축 라벨과 제목 추가
chart.set_title({'name': '시간대별 생산량'})
chart.set_x_axis({'name': '시간'})
chart.set_y_axis({'name': '생산량'})
```

```
# (6) 워크시트에 차트가 들어갈 위치를 지정해 차트 넣기
worksheet.insert_chart('D2', chart)
```

```
# (7) ExcelWriter 객체를 닫고 엑셀 파일 출력
excel_chart.save()
```


감사합니다.

※ 본 교안은 강의 수강 용도로만 사용 가능합니다.
상업적 이용을 일절 금함.