시각화

WordCloud

❖태그 클라우드(영어: tag cloud) 또는 워드 클라우드(word cloud)는 메타 데이터에서 얻어진 태그들을 분석하여 중요도나 인기도 등을 고려하여 시각적으로 늘어 놓아 표시하는 것

❖ 보통은 2차원의 표와 같은 형태로 태그들이 배치되며 이때 순서는 알파벳/가나다 순으로 배치되는데 시각적인 중요도를 강조하기 위해 각 태그들은 그 중요도(혹은 인기도)에 따라 글자의 색상이나 굵기를 다르게 해서 표시

WordCloud

❖파이썬에서 워드클라우드는 3개의 패키지가 설치되어 있어야만 합니다.

pytagcloud, pygame, simplejson

❖ 워드클라우드에서 한글 폰트를 사용하기 위해서는 한글 폰트가 설치되어 있어야 합니다.

- ✓ 한글 폰트 파일을 pytagcloud가 설치된 디렉토리의 fonts 디렉토리에 복사
  - 아나콘다설치디렉토리₩Lib₩site-packages₩pytagcloud₩fonts
- ✓ fonts.json 파일을 수정

```
[{
"name": “폰트이름",
"ttf":  " 폰트파일명 " ,
 " web " :  " 폰트패밀리"
},
{
"name": "Nobile",
"ttf": "nobile.ttf",
```

WordCloud

❖워드 클라우드 작업

1.단어의 list를 생성

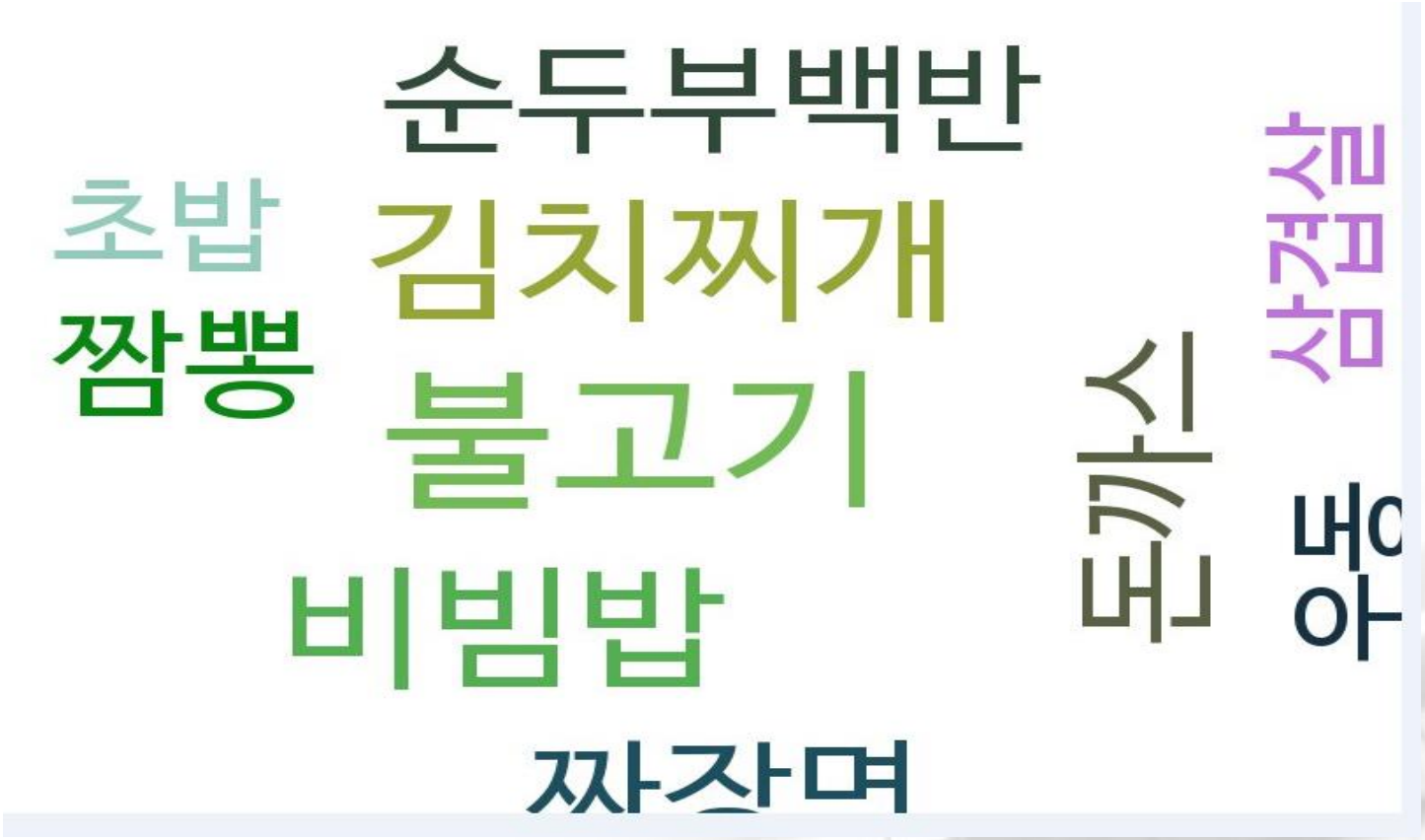
2.Collections 모듈의 Counter(단어의 list)를 이용해서 각 단어의 개수를 가지는 Counter객체를 생성

3.Counter객체를 이용해서 most_common(개수)를 호출해서 단어와 단어별 개수를 튜플로 갖는 list 객체를 생성

4. pytagcloud.make_tags(튜플의 글자수, maxsize=글자 크기의 최대 사이즈)를 호출해서 워드클라우드를 그릴 수 있는 dict의 list를 생성

5. pytagcloud.create_tag_image(dict의 list, '그림파일이름 ' , size=(너비,높이), fontname='폰트이름', rectangular=사각 출력 여부)

WordCloud



WordCloud

❖pip install 명령을 이용해서 3개의 패키지 설치

pytagcloud, pygame, simplejson

❖ 폰트 파일을 pytagcloud 디렉토리의 fonts 디렉토리에 복사하고 font.json 파일 수정

```
[{
"name": "Korean",
"ttf": "NanumBarunGothic.ttf",
"web": "http://fonts.googleapis.com/css?family=Nobile"
},
```

WordCloud

❖코드 작성

```
import pytagcloud
from collections import Counter

nouns = list()
nouns.extend(['불고기' for t in range(8)])
nouns.extend(['비빔밥' for t in range(7)])
nouns.extend(['김치찌개' for t in range(7)])
nouns.extend(['돈까스' for t in range(6)])
nouns.extend(['순두부백반' for t in range(6)])
nouns.extend(['짬뽕' for t in range(6)])
nouns.extend(['짜장면' for t in range(6)])
```

WordCloud

```
nouns.extend(['삼겹살' for t in range(5)])
nouns.extend(['초밥' for t in range(5)])
nouns.extend(['우동' for t in range(5)])

count = Counter(nouns)
tag2 = count.most_common(100)
taglist = pytagcloud.make_tags(tag2, maxsize=50)
print(taglist)
pytagcloud.create_tag_image(taglist, 'wordcloud.jpg', size=(900, 600),
fontname='Korean', rectangular=False)
```

matplotlib

❖python에서 이미지를 출력하거나 그래프를 그릴 때 가장 많이 사용하는 라이브러리가 matplotlib

❖numpy나 pandas와 함께 사용하면, MATLAB과 유사한 성능을 얻을 수 있습니다.

❖plt라는 namespace로 import하는 것이 가장 잘 알려진 방법

matplotlib

❖ **matplotlib를 이용한 이미지 출력**

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
img = mpimg.imread('wordcloud.png')
imgplot = plt.imshow(img)
plt.show()
```

WordCloud

❖동아일보 기사 검색을 이용한 워드클라우드

http://news.donga.com/search?p=16&query=%ED%96%84%EB%B2%84%EA%B1%B0&check_news=1&more=1&sorting=3&search_date=1&range=3

❖ 파라미터

- ✓ p는 데이터의 일련번호로 페이지 당 15개
- ✓ query는 검색어
- ✓ check_news: 통합검색인지 뉴스 검색인지의 여부
- ✓ more: 더보기를 누른 것인지의 여부
- ✓ sorting: 정렬
- ✓ search_date: 기간
- ✓ range: 범위

WordCloud

❖동아일보 기사 검색을 이용한 워드클라우드

http://news.donga.com/search?p=16&query=%ED%96%84%EB%B2%84%EA%B1%B0&check_news=1&more=1&sorting=3&search_date=1&range=3

✓ 위의 형식으로 검색하면 기사가 15개씩 검색됩니다.

class가 ' tit ' 인 p태그 안의 첫번째 a태그에 연결된 URL주소가 해당 기사 본문 URL이 포함된 것을 알 수 있습니다.

class='tit인 p태그를 모두 가져와서 그 안에 있는 첫번째 a태그의 'href'의 내용을 가져온다면 모든 기사의 내용을 크롤링 할 수 있습니다.

✓ 실제 기사 보기 페이지로 이동해서 살펴보면 div 태그 중에서 article_txt 라는 클래스 속성을 가진 태그가 실제 기사 내용입니다.

WordCloud

❖검색어와 파일명 및 페이지 개수를 입력 받아서 기사 내용을 텍스트 파일에 저장하기

```
from bs4 import BeautifulSoup
import urllib.request
from urllib.parse import quote

keyword = input('검색어:')
page_num = int(input('페이지 개수(페이지 당 15개):'))
output_file_name = input("저장할 파일명:")
output_file = open(output_file_name, 'w')
```

WordCloud

```
for i in range(page_num):
    current_page_num = 1 + i * 15
    target_URL = "http://news.donga.com/search?p=" + str(current_page_num) +
'&query=' + quote(
        keyword) +
'&check_news=1&more=1&sorting=3&search_date=1&range=3'
    source_code_from_URL = urllib.request.urlopen( target_URL)
    soup = BeautifulSoup(source_code_from_URL, 'lxml',
                         from_encoding='utf-8')
    for title in soup.find_all('p', 'tit'):
        title_link = title.select('a')
        article_URL = title_link[0]['href']
```

WordCloud

```
        source_code_from_url = urllib.request.urlopen(article_URL)
        soup = BeautifulSoup(source_code_from_url, 'lxml',
                             from_encoding='utf-8')
        content_of_article = soup.select('div.article_txt')
        for item in content_of_article:
            string_item = str(item.find_all(text=True))
            output_file.write(string_item)

output_file.close()
```

WordCloud

❖문자열을 가지고 워드클라우드를 만들 때 단어 단위로 분류하는 작업은 형태소 분서기를 이용합니다.

❖파이썬의 형태소 분석기 설치

- ✓ JDK 1.7 이상을 설치하고 JAVA_HOME 이라는 환경변수를 만들어서 JDK가 설치된 디렉토리를 설정
- ✓ JPype를 설치 : pip 명령으로 설치가능

pip install JPype1-0.6.2-cp35-cp35m-win_amd64.whl

http://www.lfd.uci.edu/~gohlke/pythonlibs/#jpype 사이트에서 다운로드 받아서 설치 가능: JPype1-0.6.2-cp35-cp35m-win_amd64.whl

- ✓ konlpy를 설치 : pip 명령을 이용해서 설치 가능

pip install konlpy

WordCloud

❖konlpy.tag 패키지의 Kkma 나 Twitter 모듈을 이용해서 형태소를 분리합니다.

❖nouns 메서드에 문자열을 대입하면 형태소 별로 분리된 list를 리턴합니다.

WordCloud

❖문자열을 형태소 별로 분리해서 빈도수와 함께 파일로 저장하기

```
from konlpy.tag import Twitter
from collections import Counter

noun_count = 200
result_file_name = 'count.txt'
open_text_file = open(output_file_name, 'r')
text = open_text_file.read()
spliter = Twitter()
nouns = spliter.nouns(text)
count = Counter(nouns)
```

WordCloud

```
tags = []
for n, c in count.most_common(200):
    temp = {'tag': n, 'count': c}
    tags.append(temp)
open_text_file.close()

open_output_file = open(result_file_name, 'w')
for tag in tags:
    noun = tag['tag']
    cnt = tag['count']
    open_output_file.write('{} {}₩n'.format(noun, cnt))
open_output_file.close()
```

WordCloud

❖동아일보에서 뉴스를 검색한 후 워드 클라우드 만들기

```
from bs4 import BeautifulSoup
import urllib.request
from urllib.parse import quote

keyword = input('검색어:')
page_num = int(input('페이지 개수(페이지 당 15개):'))
output_file_name = input("저장할 파일명:")
output_file = open(output_file_name, 'w')
```

WordCloud

```
for i in range(page_num):
    current_page_num = 1 + i * 15
    target_URL = "http://news.donga.com/search?p=" + str(current_page_num) +
'&query=' + quote(
        keyword) +
'&check_news=1&more=1&sorting=3&search_date=1&range=3'
    source_code_from_URL = urllib.request.urlopen( target_URL)
    soup = BeautifulSoup(source_code_from_URL, 'lxml',
                         from_encoding='utf-8')
```

WordCloud

```
for title in soup.find_all('p', 'tit'):
        title_link = title.select('a')
        article_URL = title_link[0]['href']

        source_code_from_url = urllib.request.urlopen(article_URL)
        soup = BeautifulSoup(source_code_from_url, 'lxml',
                             from_encoding='utf-8')
        content_of_article = soup.select('div.article_txt')
        for item in content_of_article:
            string_item = str(item.find_all(text=True))
            output_file.write(string_item)
output_file.close()
```

WordCloud

```
from konlpy.tag import Twitter
from collections import Counter

noun_count = 200
f = open(output_file_name, "r")
data = f.read()
nlp = Twitter()
nouns = nlp.nouns(data)
result = []
for n in nouns:
    if len(n) > 1:
        result.append(n)
```

WordCloud

```
import pytagcloud
count = Counter(result)
tag2 = count.most_common(50)
taglist = pytagcloud.make_tags(tag2, maxsize=100)
pytagcloud.create_tag_image(taglist, 'wordcloud.png', size=(900, 600),
fontname='Korean', rectangular=False)
f.close()
```

WordCloud

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
img = mpimg.imread('wordcloud.png')
imgplot = plt.imshow(img)
plt.show()
```

matplotlib

❖**import**

import matplotlib.pyplot as plt

❖**영역 객체 생성**

plt.figure()

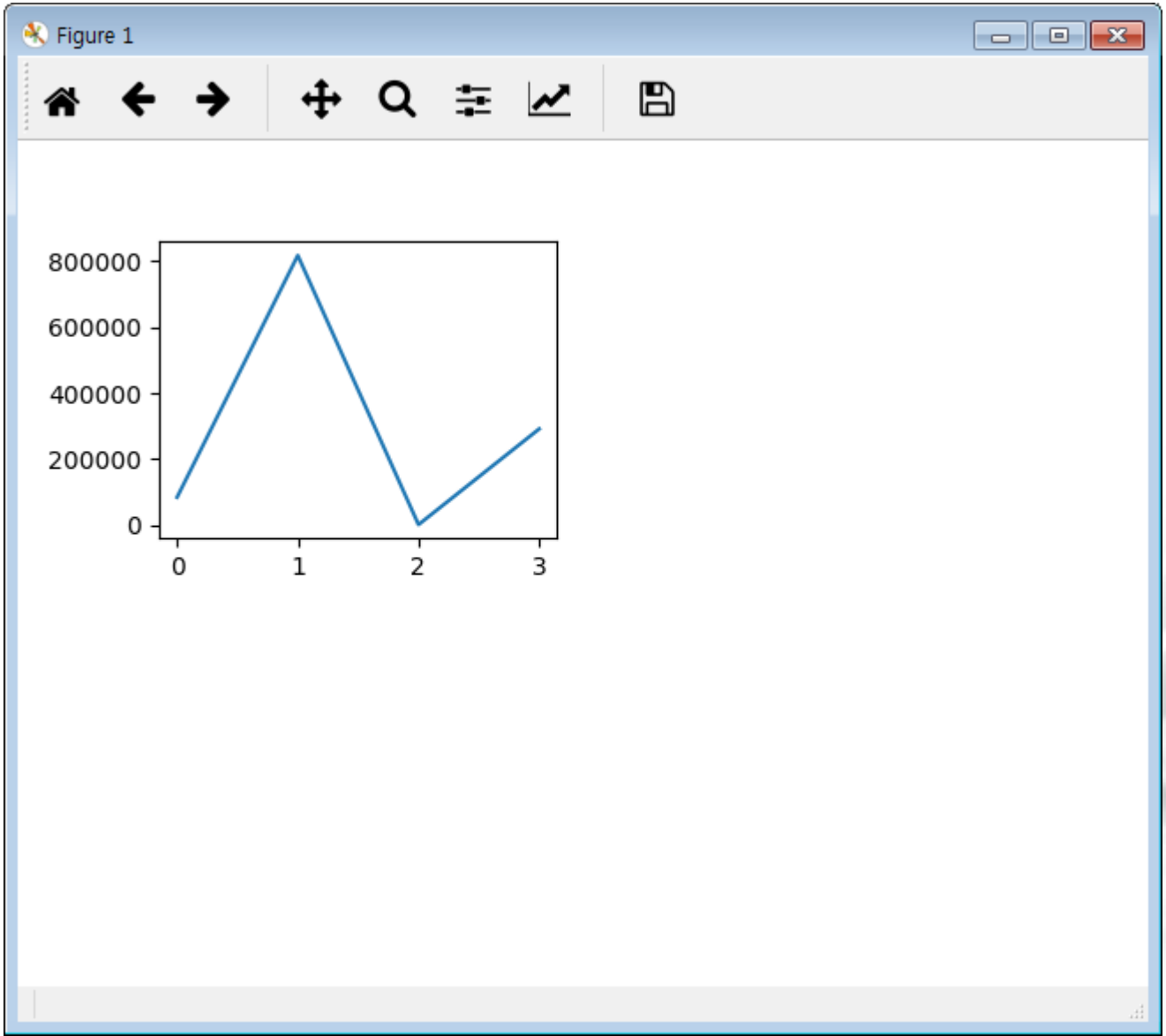
❖ **출력**

plt.show()

❖ **꺾은선 그래프**

시간의 흐름에 따라 비교할 때 꺾은선 그래프를 많이 사용

꺾은선 그래프



꺾은선 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
s = Series([84900, 818000, 1756,292000]) #Series
#s = np.arange(0, 12, 0.01) #ndarray
plt.figure()
plt.plot(s)
plt.show()
```

꺾은선 그래프

❖크기 변경

figure를 호출할 때 figsize = (가로크기, 세로크기)를 설정

단위는 inch

❖값에 그리드 적용

grid()

❖ 축의 라벨 설정

xlabel, ylabel 함수를 이용해서 축의 이름 설정

❖ 그래프의 제목 설정

title 함수

❖ 범례 설정

plot을 할 때 label 옵션에 텍스트를 입력하고 legend 함수를 호출하면 범례가 만들어집니다.

꺽은선 그래프

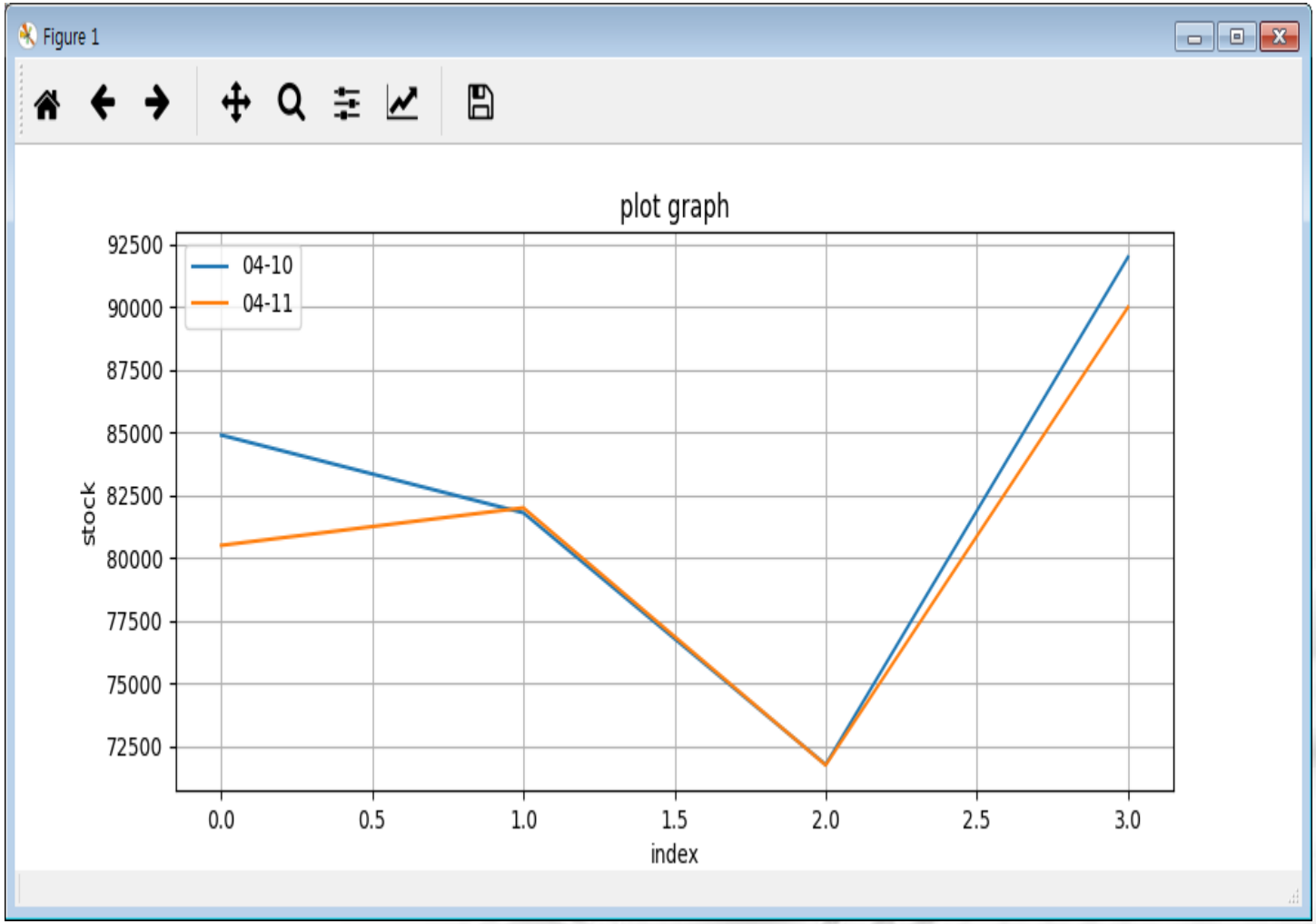


Figure 1
plot graph
04-10
04-11
92500
90000
87500
85000
82500
80000
77500
75000
72500
stock
0.0
0.5
1.0
1.5
2.0
2.5
3.0
index

꺽은선 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
s1 = Series([84900, 81800, 71756, 92000]) #Series
s2 = Series([80500, 82000, 71736, 90000]) #Series
plt.figure(figsize=(10,4))
plt.plot(s1, label='04-10')
plt.plot(s2, label='04-11')
plt.grid()
plt.xlabel('index')
plt.ylabel('stock')
plt.title('plot graph')
plt.legend()
plt.show()
```

꺽은선 그래프

❖색상 변경

plot을 할 때 color 옵션에 알파벳을 대입하거나 #RGB 값을 이용해서 설정

❖선의 두께

lw 옵션에 숫자로 설정

❖x축 이나 y축 범위 설정

xlim(최소, 최대) 또는 ylim(최소, 최대)로 설정

❖ 눈금 간격

xticks 나 yticks 함수에 범위를 range를 이용해서 만들거나 list로 대입

labels 옵션을 이용해서 텍스를 대입하고 rotation을 이용해서 텍스트 출력 방향 설정

❖ 눈금의 문자열 변경

xticklabels 나 yticklabels 함수를 이용

꺾은선 그래프

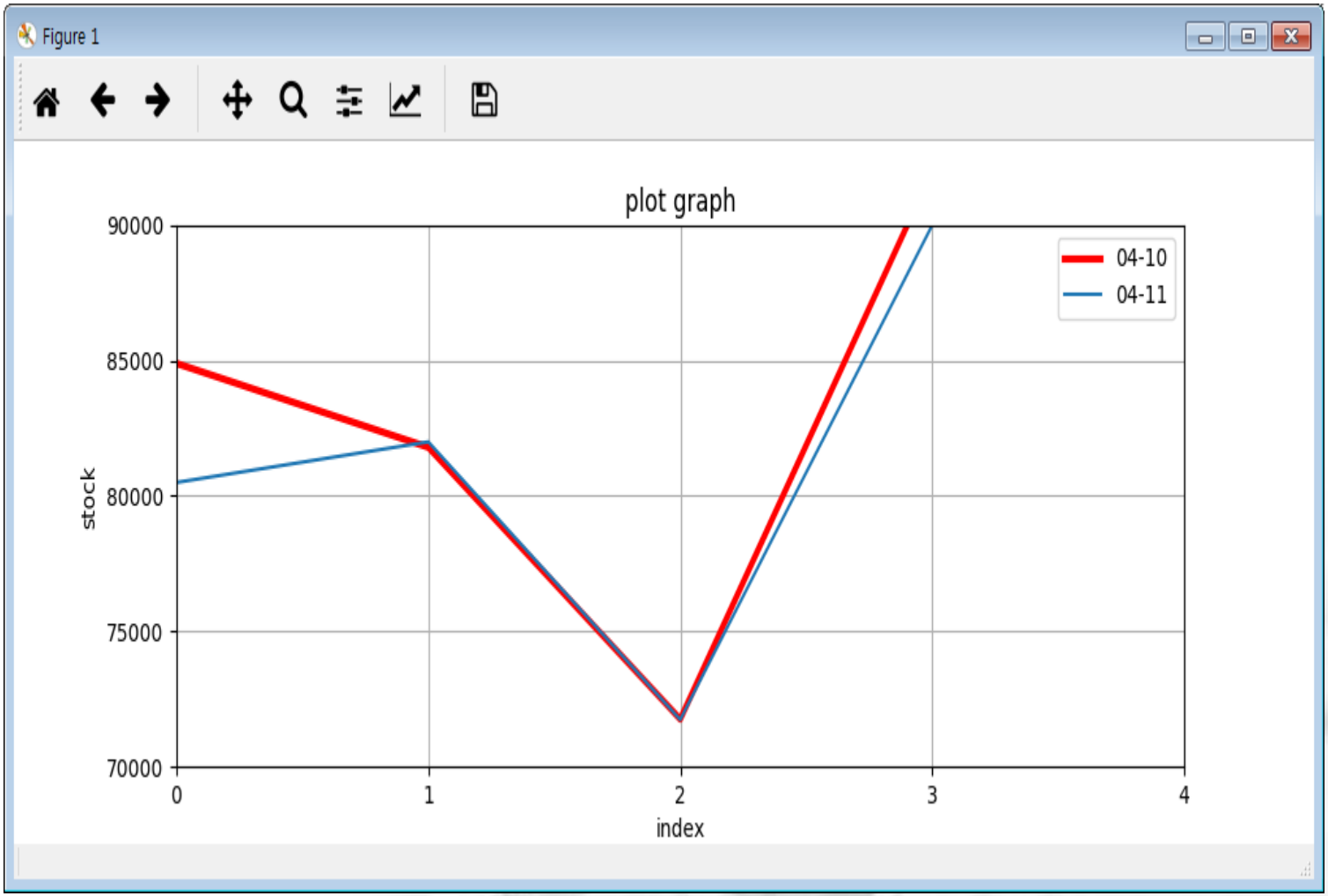


Figure 1
plot graph
90000
85000
80000
75000
70000
stock
0
1
2
3
4
index
04-10
04-11

꺽은선 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
s1 = Series([84900, 81800, 71756, 92000]) #Series
s2 = Series([80500, 82000, 71736, 90000]) #Series
plt.figure(figsize=(10,4))
plt.plot(s1, label='04-10', lw=3, color='r')
plt.plot(s2, label='04-11')
plt.xlim(0.0, 4.0)
plt.ylim(70000, 90000)
plt.xticks(range(0,5,1))
plt.yticks(range(70000,95000,5000))
```

꺾은선 그래프

```
plt.grid()
plt.xlabel('index')
plt.ylabel('stock')
plt.title('plot graph')
plt.legend()
plt.show()
```

꺽은선 그래프

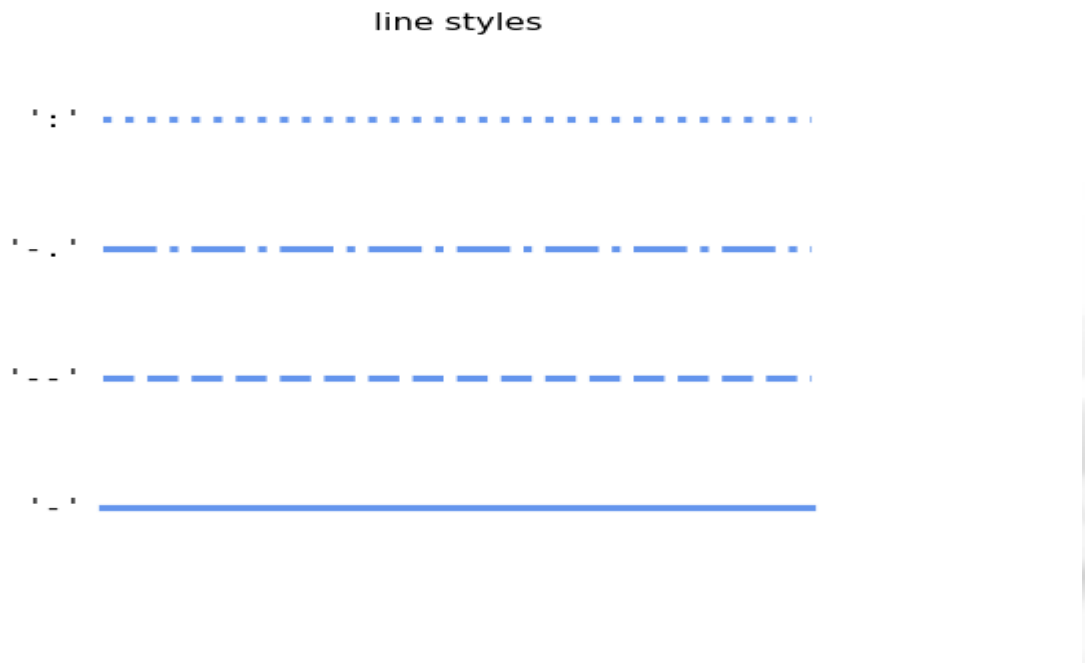
❖꺽은 선 그래프 옵션

- ✓ color
  - B: blue
  - G: green
  - R: red
  - C: cyan
  - M: magenta
  - Y: yellow
  - K: black
  - W: white
  - 숫자로 동일한 값 입력 가능
  - rgb 형식으로 입력 가능
- ✓ markerfacecolor 옵션으로 마커의 색상 설정

꺾은선 그래프

❖꺾은 선 그래프 옵션

✓ linestyle



꺽은선 그래프

❖꺽은 선 그래프 옵션

✓ marker

- "." point
- "," pixel
- "o" circle
- "v" triangle_down
- "^" triangle_up
- "<" triangle_left
- ">" triangle_right
- "1" tri_down
- "2" tri_up
- "3" tri_left
- "4" tri_right
- "8" octagon
- "s" square
- "p" pentagon
- "P" plus (filled)
- "*" star
- "h" hexagon1
- "H" hexagon2
- "+" plus
- "x" x
- "X" x (filled)
- "D" diamond
- "d" thin_diamond
- "|" vline
- "_" hline
- TICKLEFT tickleft
- TICKRIGHT tickright
- TICKUP tickup
- TICKDOWN tickdown
- CARETLEFT caretleft (centered at tip)
- CARETRIGHT caretright (centered at tip)
- CARETUP caretup (centered at tip)
- CARETDOWN caretdown (centered at tip)
- CARETLEFTBASE caretleft (centered at base)
- CARETRIGHTBASE caretright (centered at base)
- CARETUPBASE caretup (centered at base)
- "None", " " or "" nothing
- '$...$' render the string using mathtext.

✓**Markersize** 옵션으로 마커의 크기 설정

꺽은선 그래프

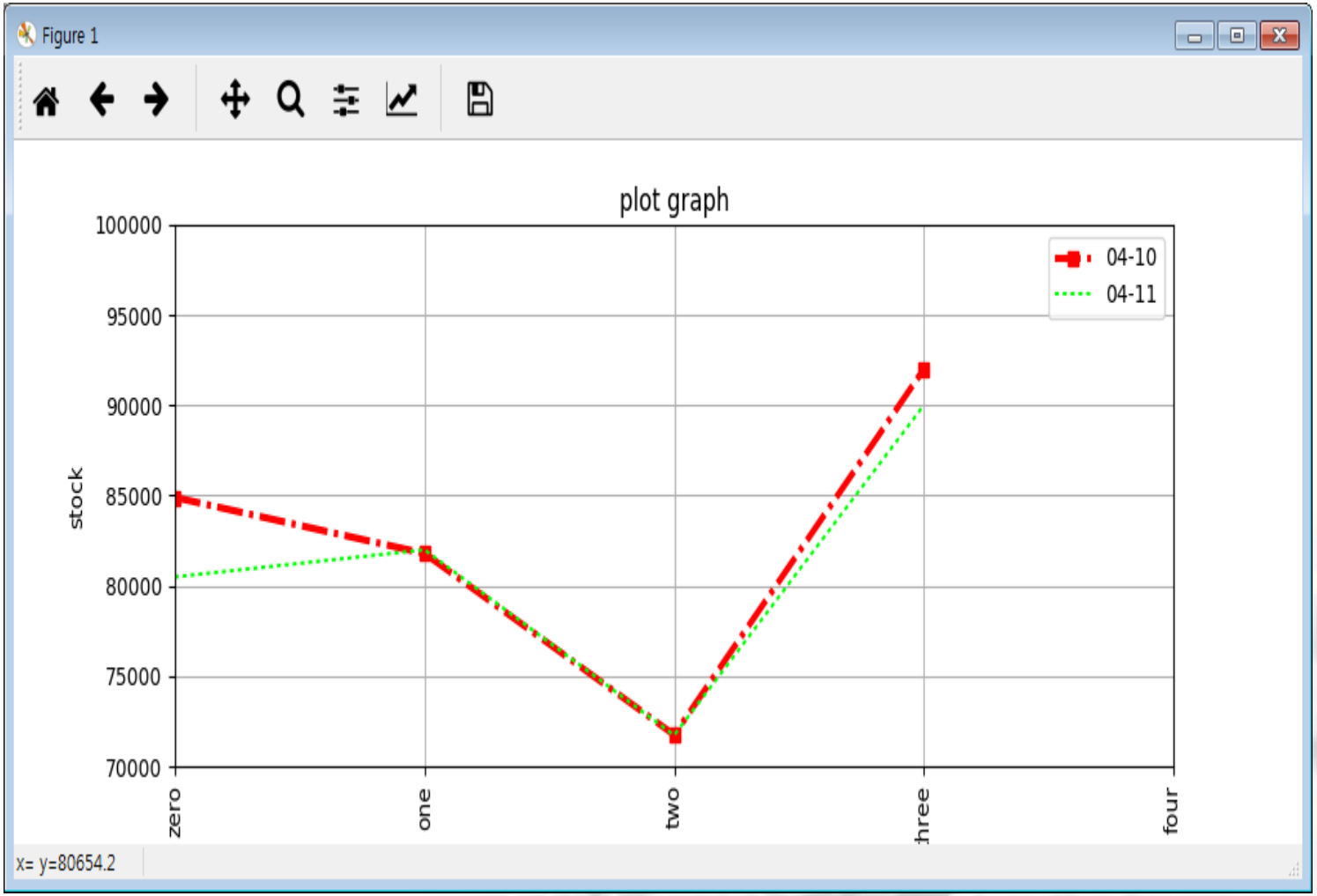


Figure 1
plot graph
100000
95000
90000
85000
80000
75000
70000
stock
04-10
04-11
zero
one
two
three
four
x= y=80654.2

꺽은선 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
s1 = Series([84900, 81800, 71756, 92000]) #Series
s2 = Series([80500, 82000, 71736, 90000]) #Series
plt.figure(figsize=(10,4))
plt.plot(s1, label='04-10', lw=3, color='r', linestyle='-.' , marker='s')
plt.plot(s2, label='04-11', linestyle=':', color='#00ff00')
plt.xlim(0.0, 4.0)
plt.ylim(70000, 90000)
```

꺽은선 그래프

```
plt.xticks(range(0,5,1),['zero', 'one', 'two', 'three', 'four'], rotation='vertical')
plt.yticks(range(70000,105000,5000))
plt.grid()
plt.xlabel('index')
plt.ylabel('stock')
plt.title('plot graph')
plt.legend()
plt.show()
```

산포도

❖산포도 그래프

- ✓ 자료의 분포를 표시할 때 많이 사용하는 그래프
- ✓ scatter를 이용해서 생성
- ✓ 색상에 변화를 줄 항목을 c에 지정해서 색상을 다르게 설정 가능
- ✓ colorbar()를 호출하면 인덱스에 해당하는 색상을 출력

❖ 차트에 표시해야 하는 문자에 한글이 있는 경우

```
from matplotlib import font_manager, rc
font_name = font_manager.FontProperties(fname="글꼴 파일 이름").get_name()
rc('font', family=font_name)
```

산포도

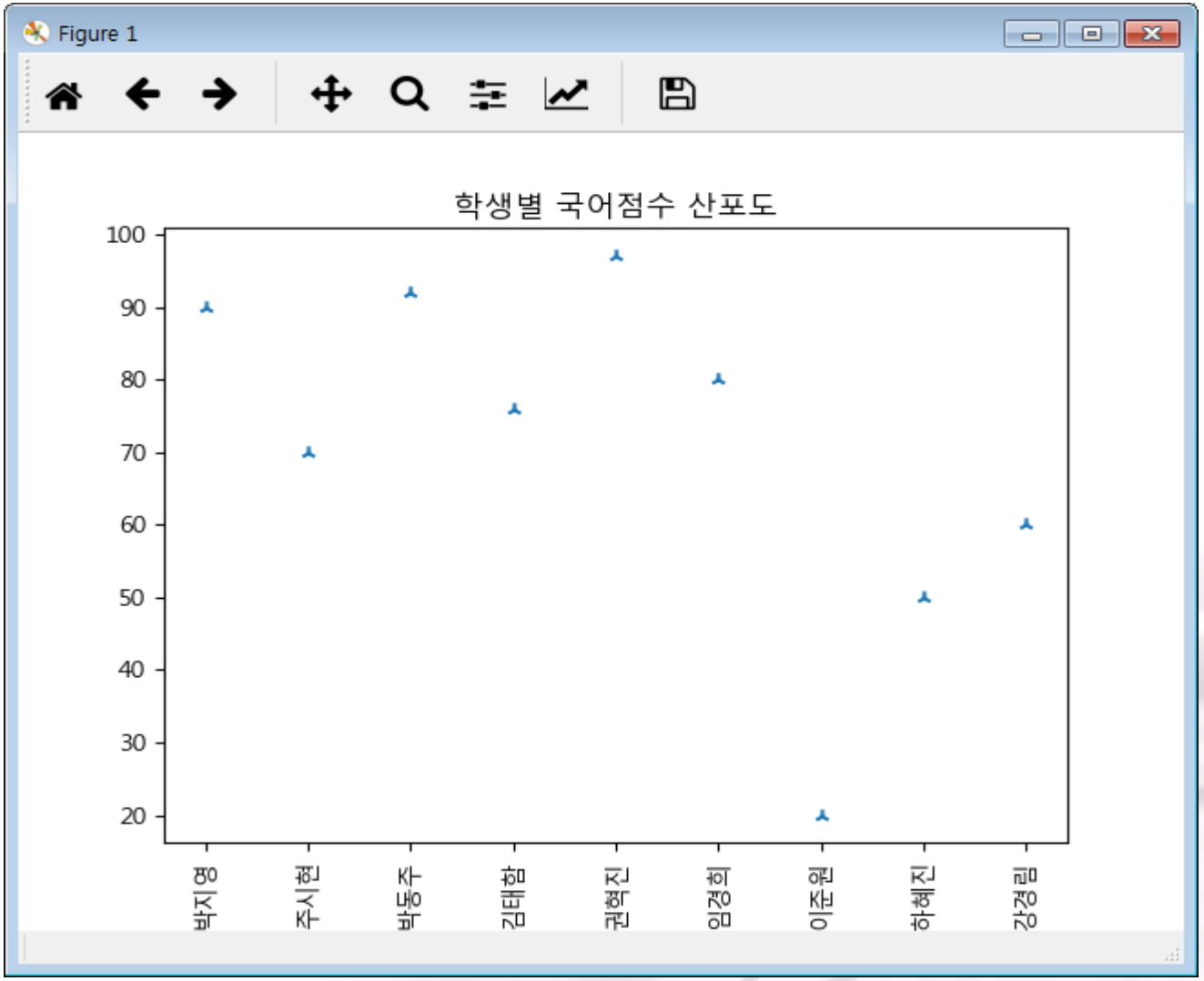


Figure 1
학생별 국어점수 산포도
100
90
80
70
60
50
40
30
20
박지영
주시현
박동주
김태함
권혁진
임경희
이준원
하혜진
강경림

산포도

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)

df = pd.read_csv('korea.csv',encoding='ms949')
plt.figure()
plt.scatter(x=df.index,y=df['점수'], marker='2')
plt.xticks(range(0,len(df['점수']),1),df['이름'], rotation='vertical')
plt.title('학생별 국어점수 산포도')
plt.show()
```

산포도

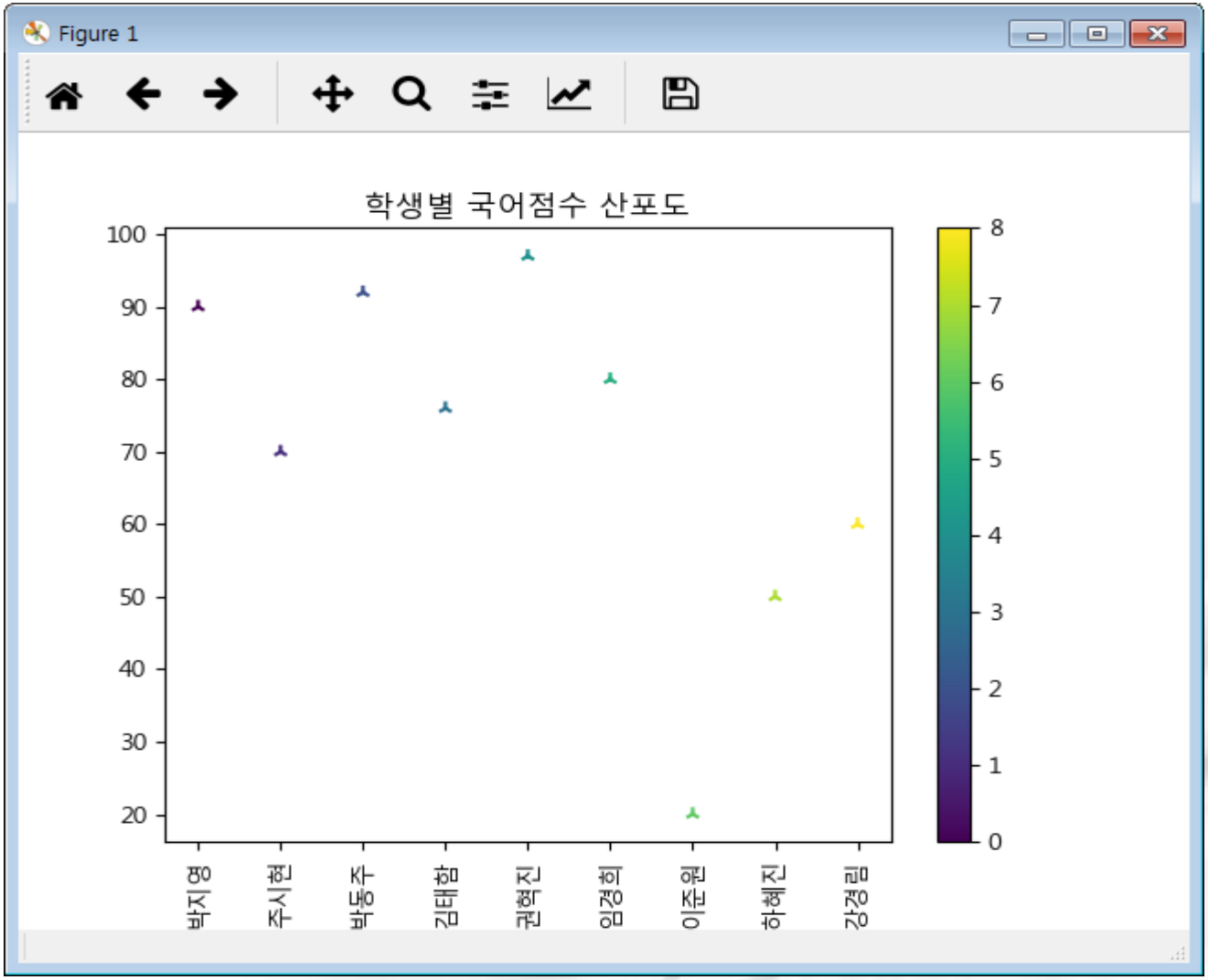


Figure 1
학생별 국어점수 산포도
100
90
80
70
60
50
40
30
20
박지영
주시현
박동주
김태함
권혁진
임경희
이준원
하혜진
강경림
8
7
6
5
4
3
2
1
0

산포도

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

산포도

```
df = pd.read_csv('korea.csv',encoding='ms949')
plt.figure()
colormap = df.index
plt.scatter(x=df.index,y=df['점수'], marker='2', c=colormap)
plt.xticks(range(0,len(df['점수']),1),df['이름'], rotation='vertical')
plt.colorbar()
plt.title('학생별 국어점수 산포도')
plt.show()
```

막대 그래프

❖막대 그래프

- ✓ 비교 대상이 있을 때나 강조해야 하는 경우에 사용하는 그래프
- ✓ bar 함수로 출력
- ✓ barh 함수로 출력하면 수평 막대 그래프
- ✓ 옵션으로 width를 이용해서 너비 변경 가능
- ✓ 2개를 출력할 때는 첫번째 바의 width를 0.5이하로 잡고 x 좌표를 이동해주면 됩니다.

막대 그래프

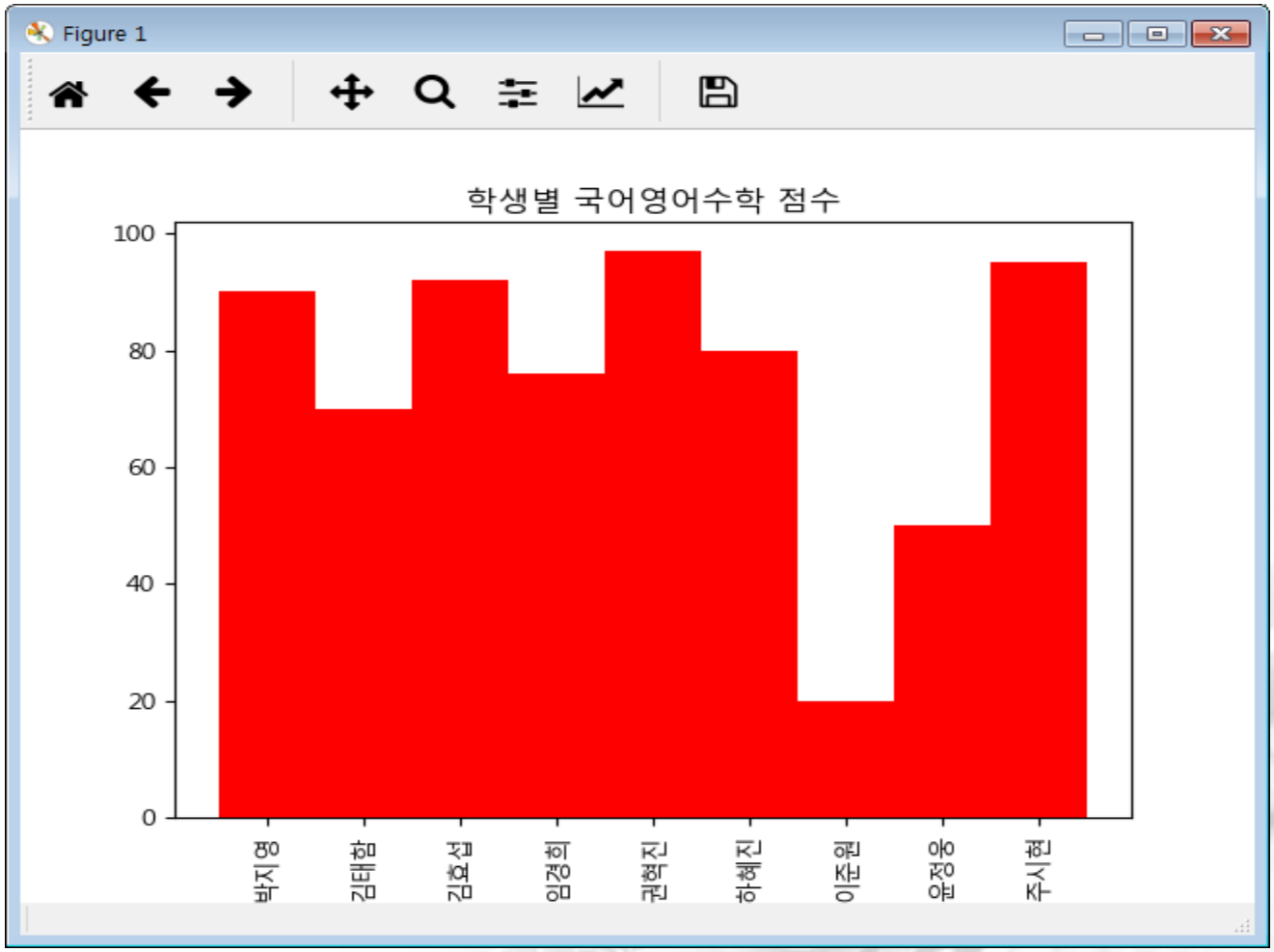


Figure 1
학생별 국어영어수학 점수
100
80
60
40
20
0
박지영
김태함
김효섭
임경희
권혁진
하혜진
이준원
엽정웅
주시현

막대 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
df = pd.read_csv('student.csv',encoding='ms949')
print(df)
plt.figure()
plt.bar(df.index, df['국어'],width=1.0, color='r')
plt.xticks(range(0,len(df.index),1),df['이름'], rotation='vertical')
plt.title('학생별 국어 점수')
plt.show()
```

막대 그래프

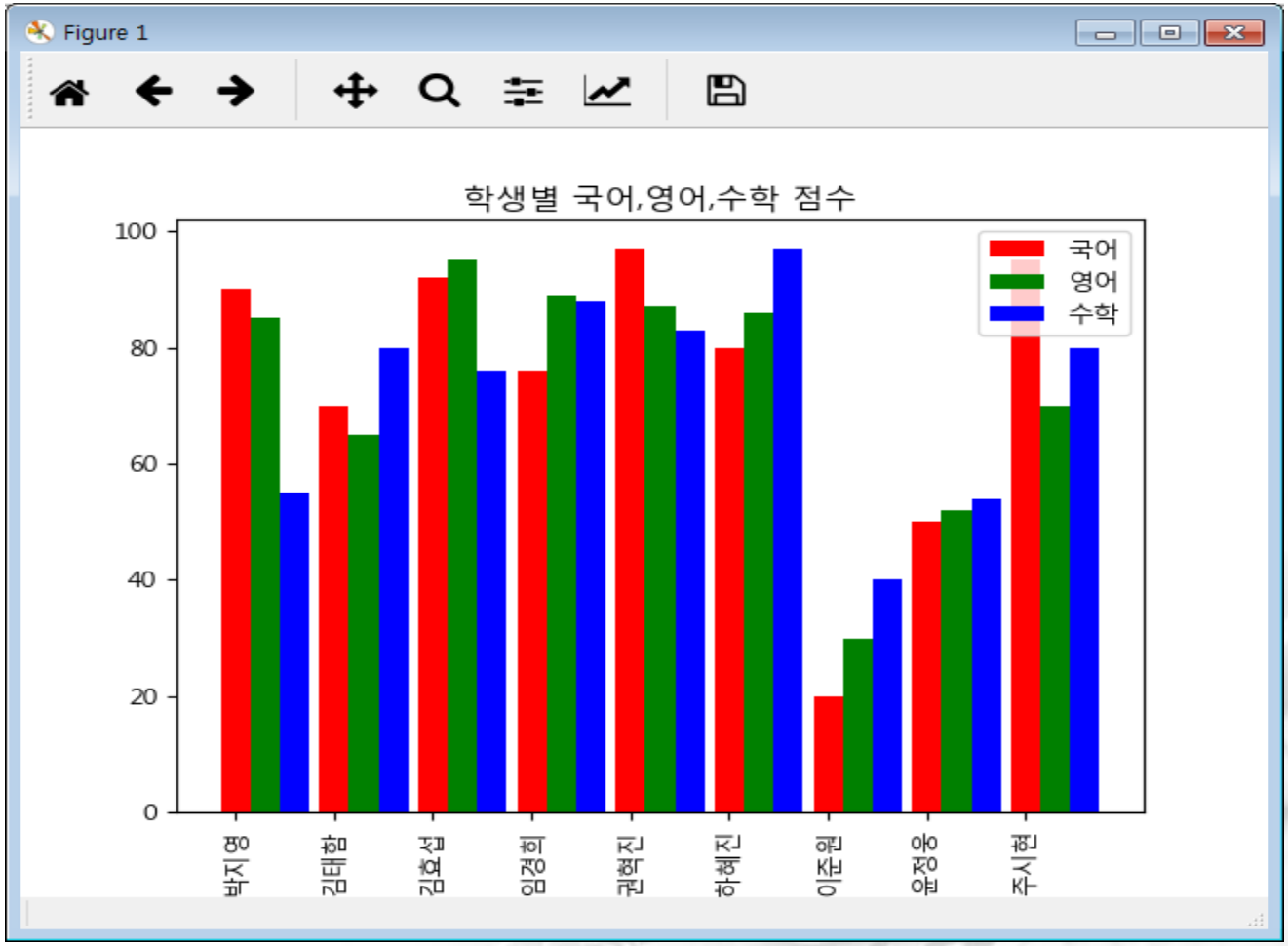


Figure 1
학생별 국어,영어,수학 점수
100
80
60
40
20
0
국어
영어
수학
박지영
김태함
김효섭
임경희
권혁진
하혜진
이준원
엄정웅
주시현

막대 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

막대 그래프

```
df = pd.read_csv('student.csv',encoding='ms949')
print(df)
plt.figure()
plt.bar(df.index, df['국어'],width=0.3, color='r', label='국어')
plt.bar(df.index+0.3, df['영어'],width=0.3, color='g', label='영어')
plt.bar(df.index+0.6, df['수학'],width=0.3, color='b', label='수학')
plt.xticks(range(0,len(df.index),1),df['이름'], rotation='vertical')
plt.title('학생별 국어,영어,수학 점수')
plt.legend()
plt.show()
```

막대 그래프

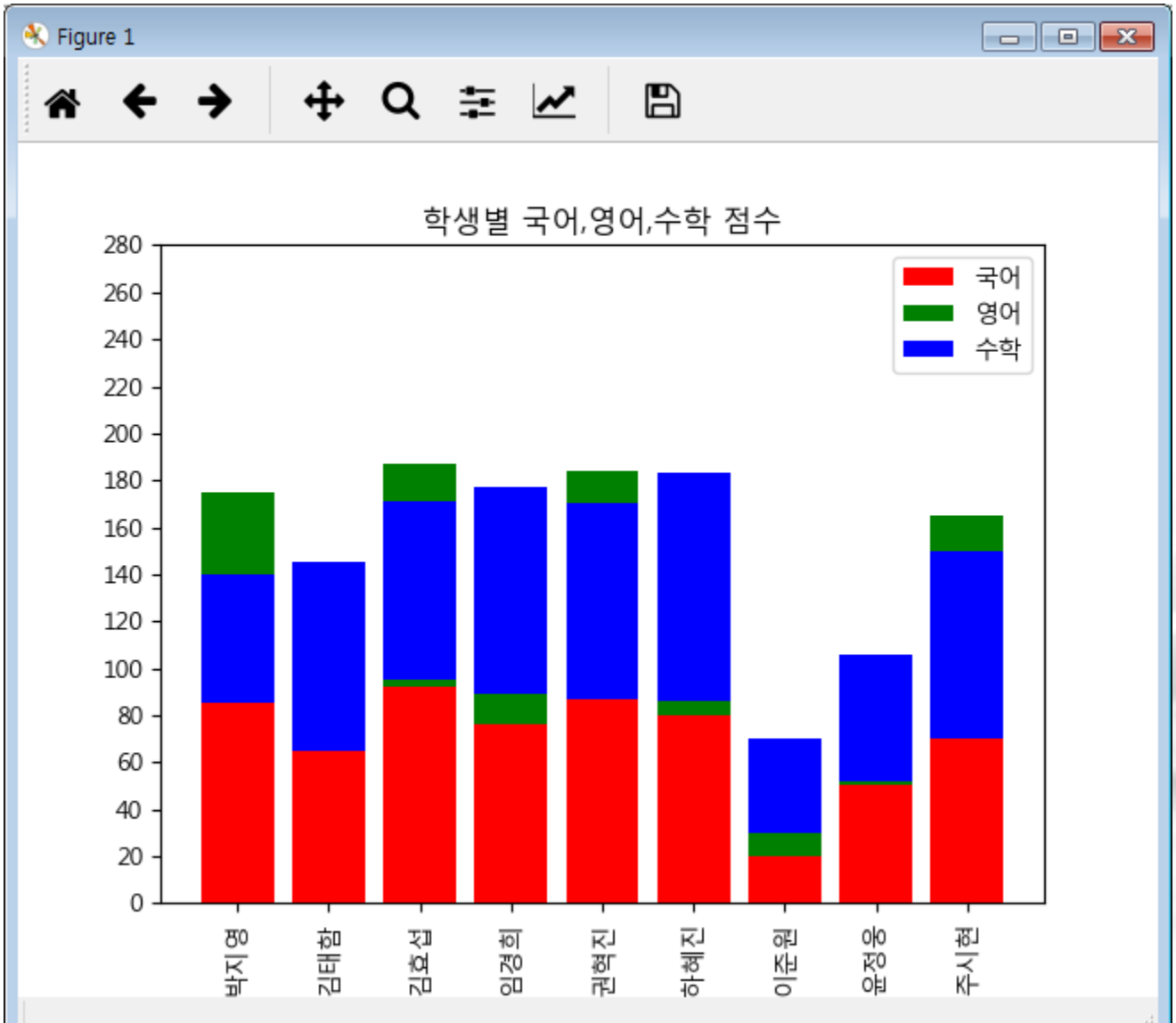


Figure 1
학생별 국어,영어,수학 점수
국어
영어
수학
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
박지영
김태함
김효섭
임경희
권혁진
하혜진
이준원
윤정웅
주시현

막대 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

막대 그래프

```
df = pd.read_csv('student.csv',encoding='ms949')
print(df)
plt.figure()
plt.bar(df.index, df['국어'], color='r', label='국어')
plt.bar(df.index, df['영어'], color='g', label='영어', bottom=df['국어'])
plt.bar(df.index, df['수학'], color='b', label='수학', bottom=df['영어'])
plt.xticks(range(0,len(df.index),1),df['이름'], rotation='vertical')
plt.yticks(range(0,300,20))
plt.title('학생별 국어,영어,수학 점수')
plt.legend()
plt.show()
```

막대 그래프


Figure 1
학생별 국어,영어 점수
국어
영어
주시현
윤정웅
이준원
하혜진
권혁진
임경희
김효섭
김태함
박지영
100
50
0
50
100

막대 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

막대 그래프

```
df = pd.read_csv('student.csv',encoding='ms949')
print(df)
plt.figure()
plt.barh(df.index, df['국어'], color='r', label='국어')
plt.barh(df.index, -df['영어'], color='g', label='영어')
plt.title('학생별 국어,영어 점수')
plt.yticks(range(0,len(df.index),1),df['이름'], rotation='horizontal')
plt.xticks([-100,-50,0,50,100],(100,50,0,50,100))
plt.legend()
plt.show()
```

파이 그래프

❖파이 그래프

- ✓ 전체에서의 기여도를 파악할 때 유용
- ✓ Pie 함수로 출력
- ✓ labels 옵션을 이용해서 각 데이터의 레이블을 출력
- ✓ colors 옵션으로 색상 설정
- ✓ explode 옵션으로 조각 분할
- ✓ autopct 옵션으로 백분율 표시

파이 그래프

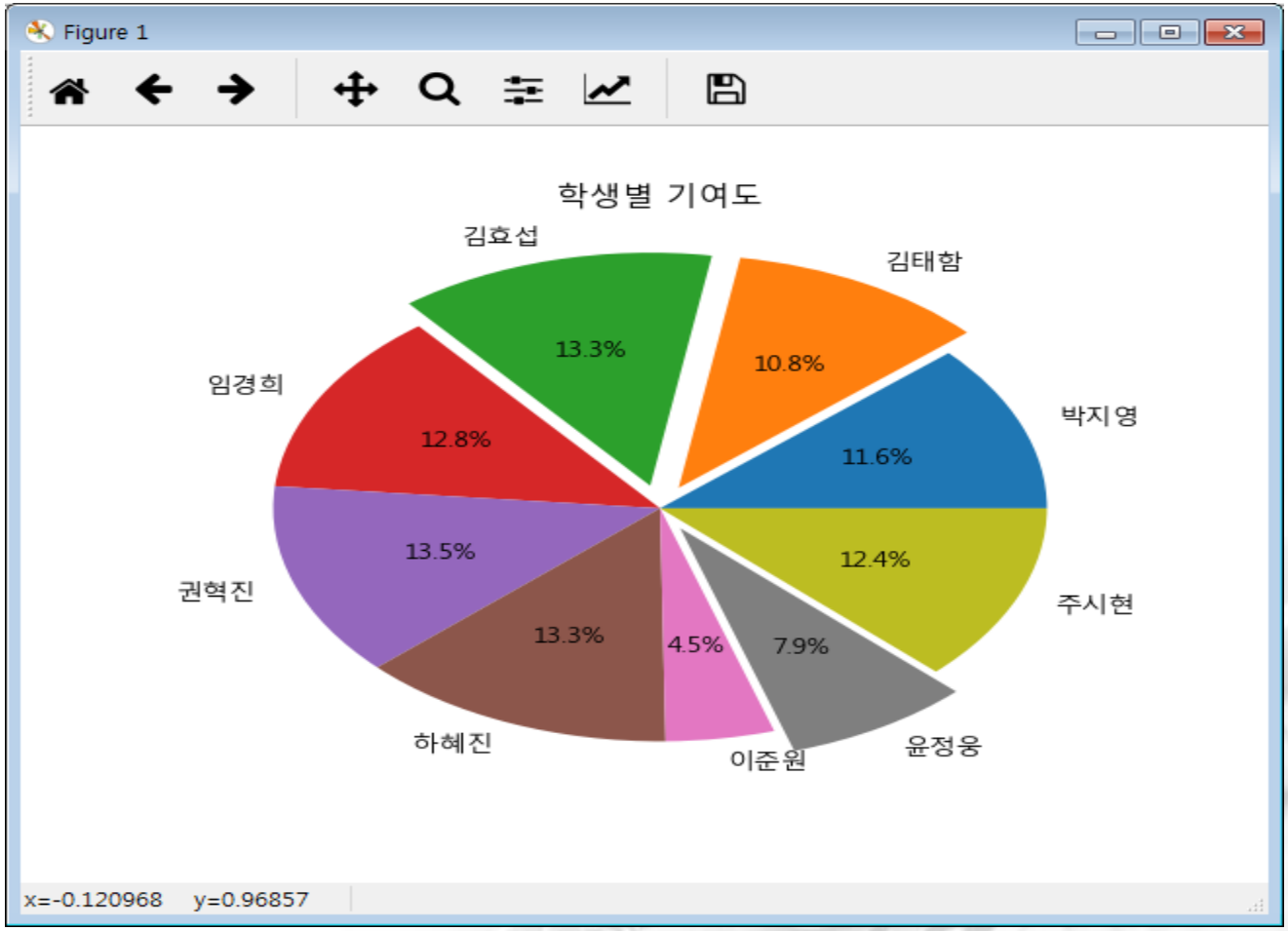


Figure 1
학생별 기여도
김효섭
13.3%
김태함
10.8%
박지영
11.6%
주시현
12.4%
윤정웅
7.9%
이준원
4.5%
하혜진
13.3%
권혁진
13.5%
임경희
12.8%
x=-0.120968 y=0.96857

파이 그래프

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

파이 그래프

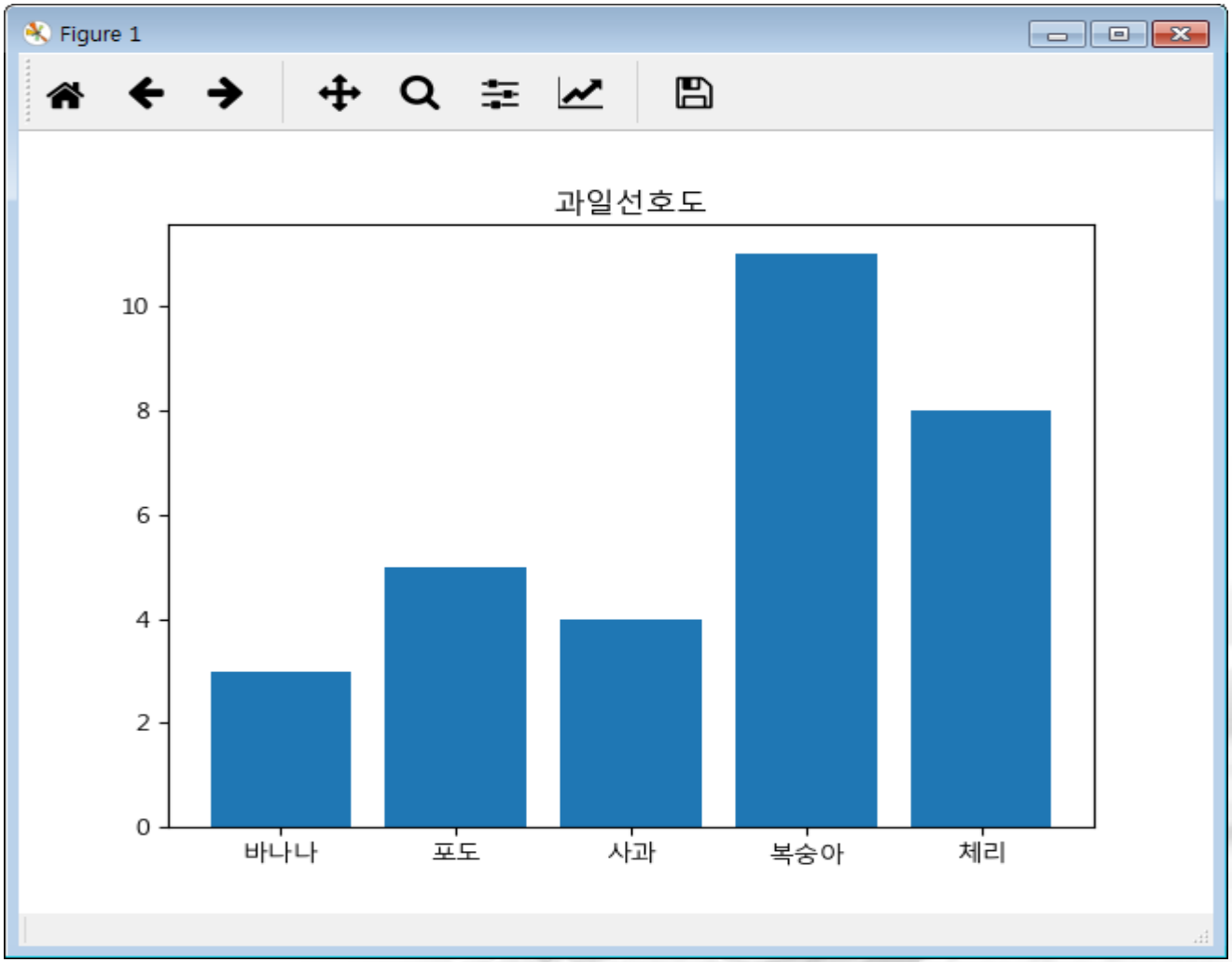
```
df = pd.read_csv('student.csv',encoding='ms949')
print(df)
plt.figure()
explode = (0, 0.1, 0.1, 0, 0, 0, 0, 0.1, 0)
plt.pie(df['국어']+df['영어']+df['수학'], labels=df['이름'],explode=explode,
autopct='%1.1f%%')
plt.title('학생별 기여도')
#plt.legend()
plt.show()
```

히스토그램

❖히스토그램

- ✓ 빈도 분석을 위해서 그리는 차트
- ✓ value_counts()로 빈도를 조사해서 막대 그래프로 그리는 것도 가능
- ✓ 숫자 데이터의 경우 hist()를 이용해서 바로 작성 가능한데 bins 옵션을 이용해서 구간의 개수를 설정

히스토그램



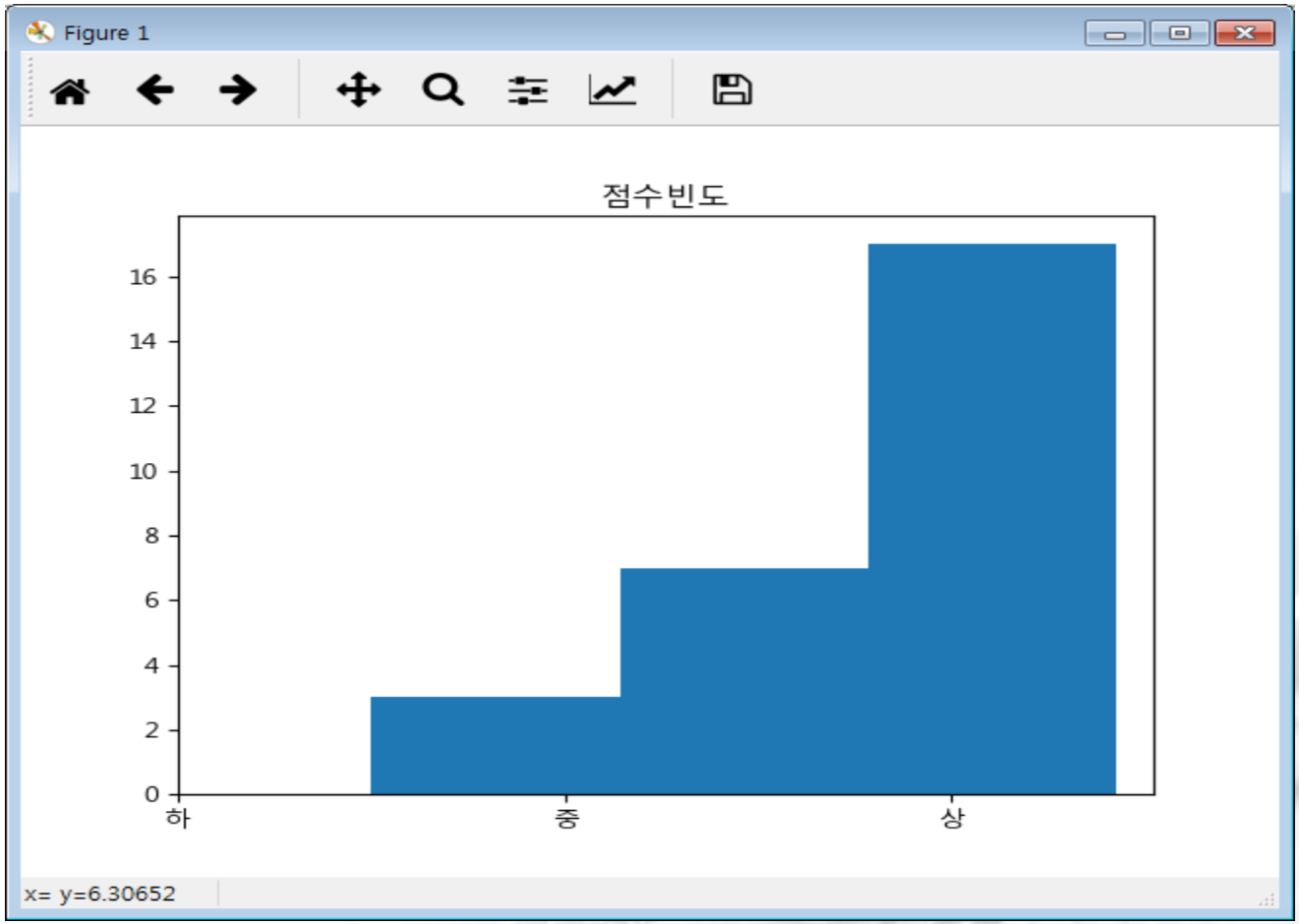
히스토그램

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

히스토그램

```
df = pd.read_csv('lovefruits.csv',encoding='ms949')
data = df['선호과일'].value_counts(sort=False)
print(data.index)
plt.bar(range(0,len(data),1), data)
plt.xticks(range(0,len(data),1),data.index)
plt.title('과일선호도')
plt.show()
```

히스토그램



히스토그램

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
df = pd.read_csv('student.csv',encoding='ms949')
data = pd.concat([df['국어'],df['영어'],df['수학']])
plt.hist(data, bins=3)
plt.xticks(range(0,100,40),['하', '중', '상'])
plt.title('점수빈도')
plt.show()
```

히스토그램

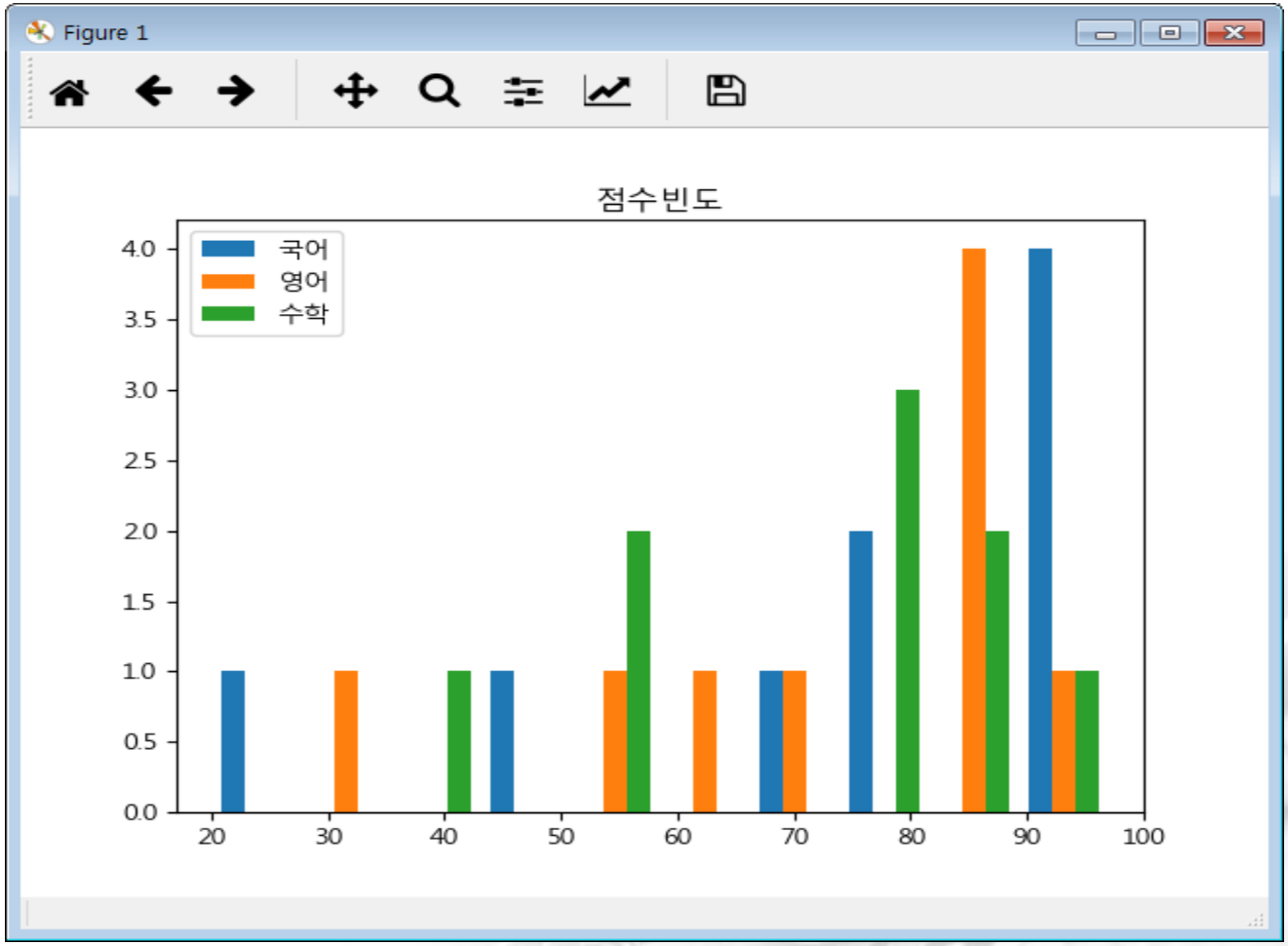


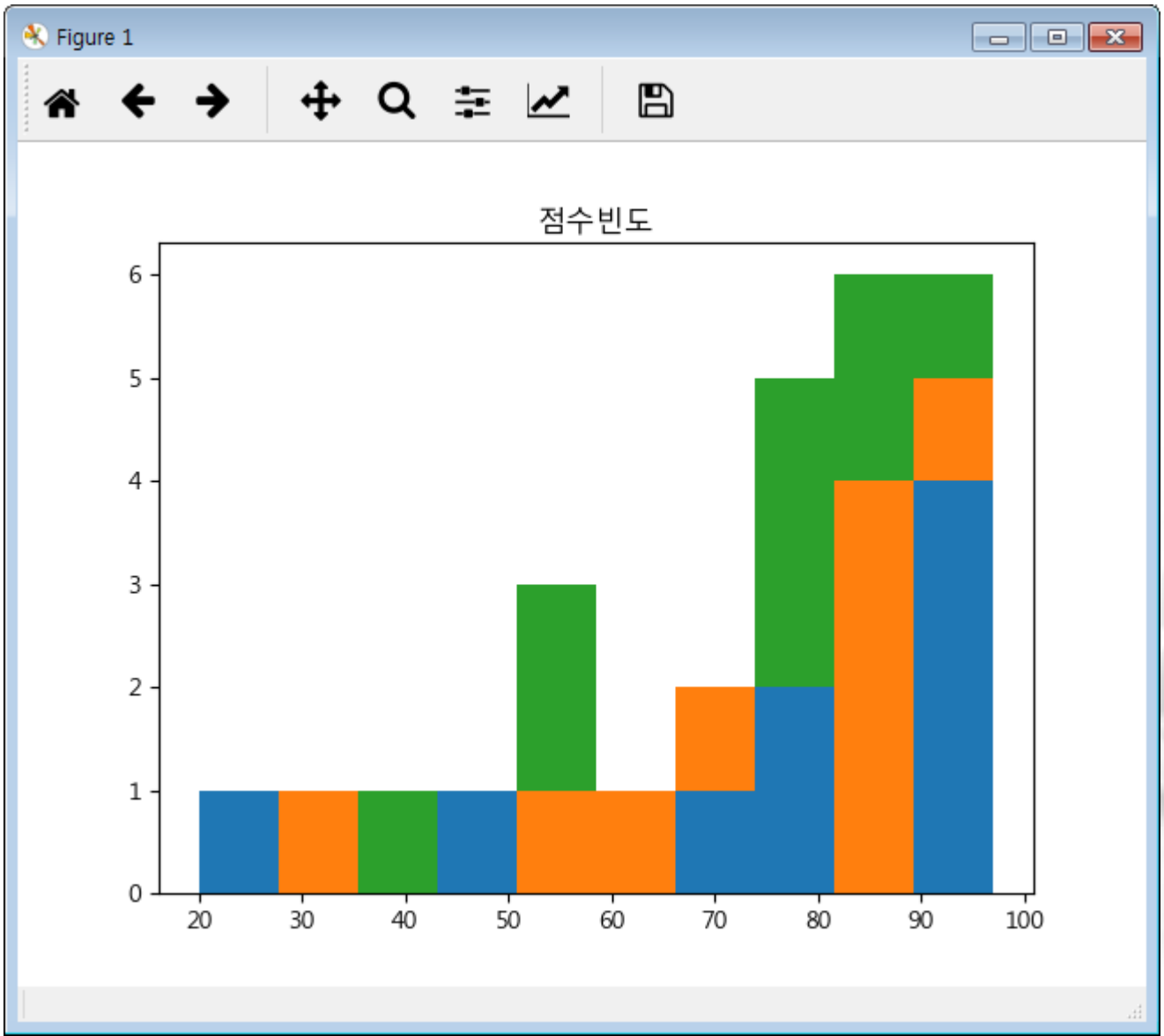
Figure 1
점수빈도
국어
영어
수학
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
20
30
40
50
60
70
80
90
100

히스토그램

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)

df = pd.read_csv('student.csv',encoding='ms949')
plt.hist((df['국어'],df['영어'],df['수학']), bins=10, label=('국어','영어','수학'))
plt.title('점수빈도')
plt.legend()
plt.show()
```

히스토그램



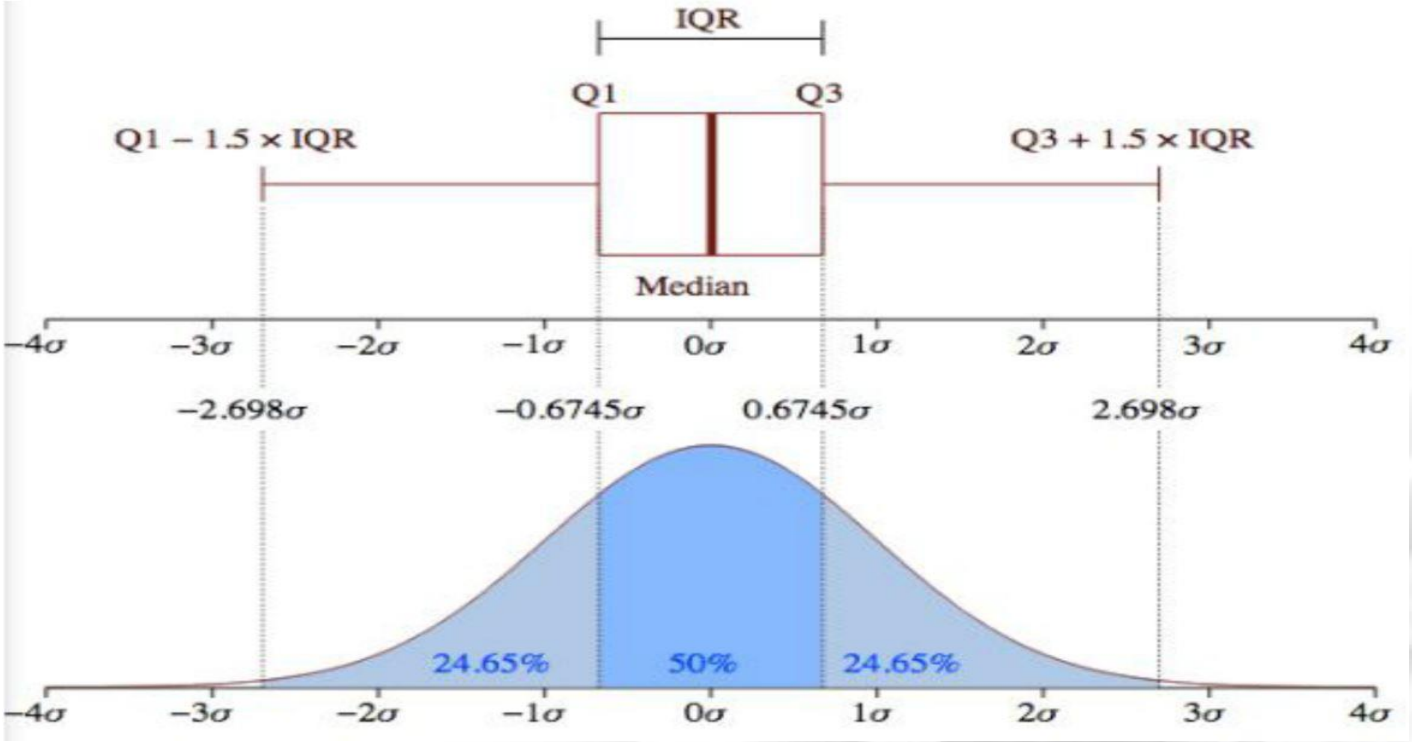
히스토그램

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
x = np.random.randn(1000, 3)
print(x)
df = pd.read_csv('student.csv',encoding='ms949')
plt.hist([df['국어'],df['영어'], df['수학']],  stacked=True)
plt.title('점수빈도')
plt.legend()
plt.show()
```

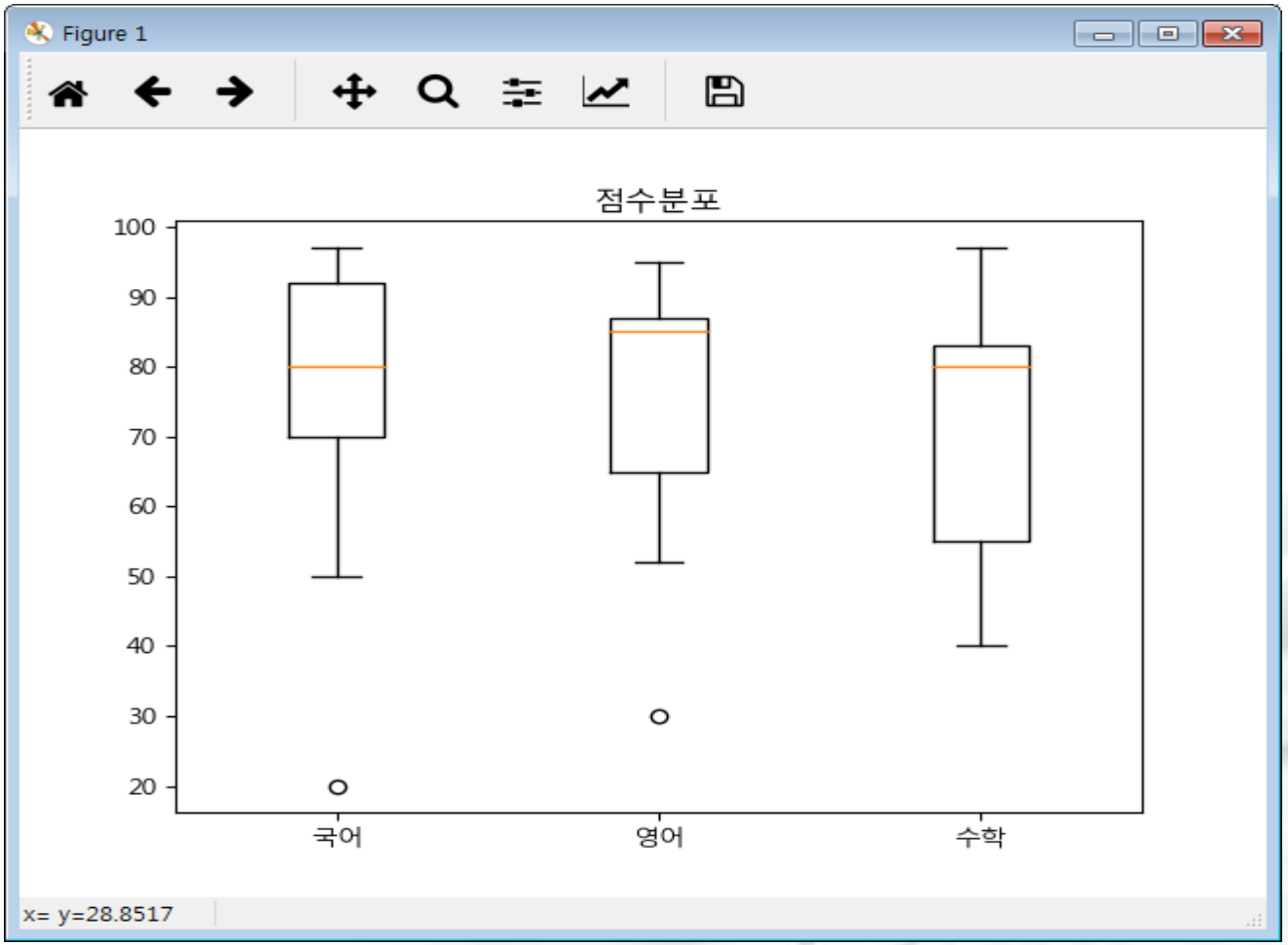
BoxPlot

❖BoxPlot

✓ 평균 값을 기준으로 50%의 데이터가 출현하는 범위를 출력하는 차트



BoxPlot



BoxPlot

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

BoxPlot

```
df = pd.read_csv('student.csv',encoding='ms949')
print(df)
plt.boxplot((df['국어'],df['영어'],df['수학']), labels=('국어','영어','수학'))
print(df['수학'].min())
print(df['수학'].mean())
print(df['수학'].median())
plt.title('점수분포')
plt.legend()
plt.show()
```