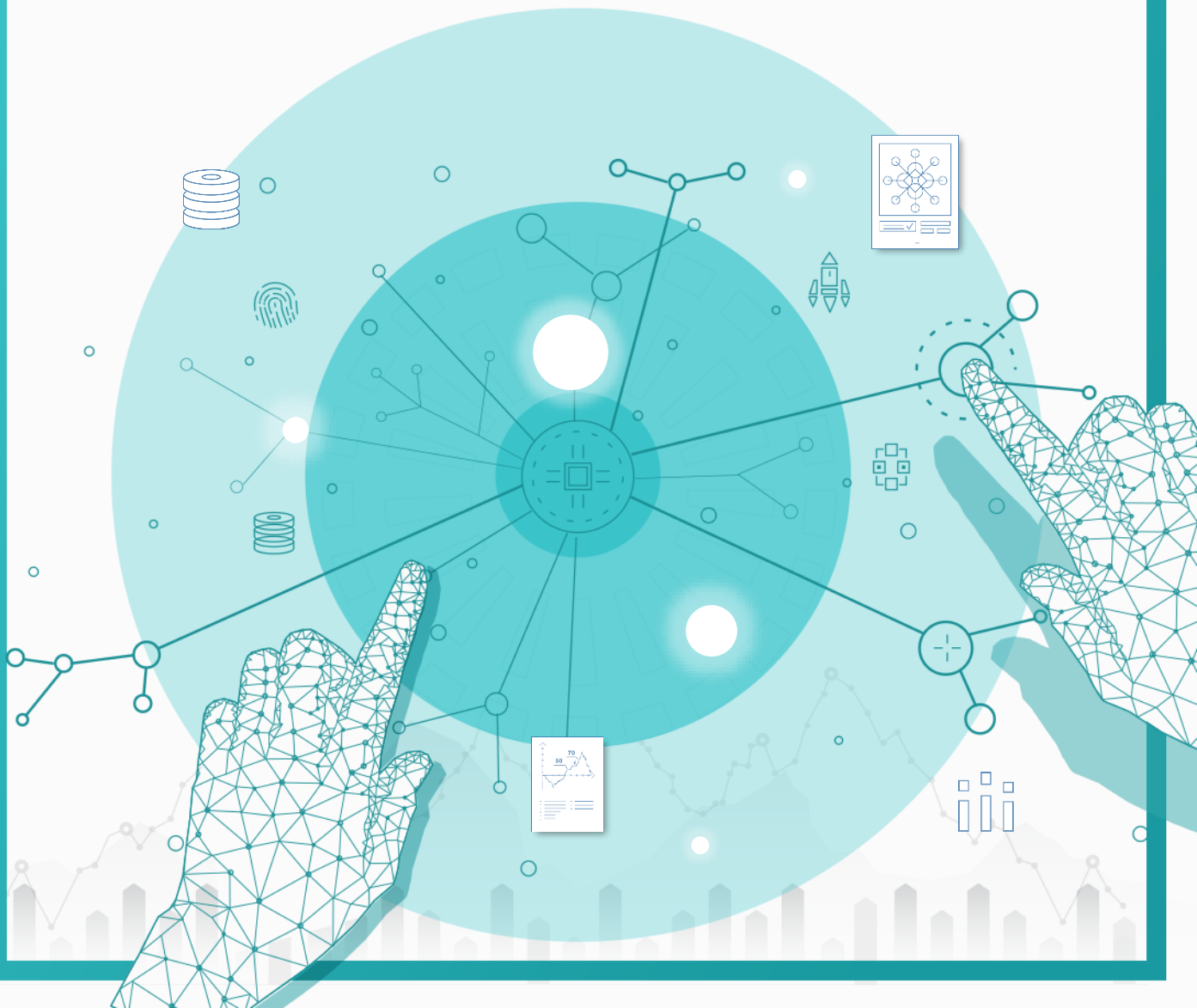
한국기술교육대학교
온라인평생교육원

파이썬 라이브러리로 하는 데이터 분석과 시각화

데이터 시각화를 위한 파이썬 모듈



데이터 시각화를 위한 파이썬 모듈

학습 목표

1. 파이썬 시각화 모듈의 종류를 설명하고, 코딩할 수 있다.
2. Matplotlib 모듈을 활용해 코딩할 수 있다.
3. Wordcloud 모듈을 활용해 코딩할 수 있다.

학습 내용

1. 파이썬 시각화 모듈의 이해
2. Matplotlib 모듈의 이해와 활용
3. Wordcloud 모듈의 이해와 활용

1. 파이썬 시각화 모듈의 이해

1) 데이터 시각화의 개념

- 데이터 분석 결과를 시각적으로 표현하고 전달하는 과정
- 디지털 시대의 학습자는 SNS, 영상, 클라우드 서비스 등 무한정의 데이터에 노출되어 있음
- 시간이 흐를수록 데이터의 표현이나 수용 방식의 변화가 필요함

➡ 정보를 효율적이고 명확하게 제공하는 데이터 시각화의 필요성이 점점 증가하는 추세

2) 데이터 시각화의 필요성

- 많은 양의 데이터를 한눈에 볼 수 있음
- 전문 지식이 없어도 누구나 쉽게 데이터를 인지하고 활용할 수 있음
- 단순한 데이터의 요약, 통계보다 정확한 데이터 분석 결과를 도출해 낼 수 있음
- 단순하게 나열된 데이터에서는 알 수 없었던 또 다른 데이터의 중요한 정보를 파악할 수 있음

3) 데이터 시각화를 위한 모듈의 종류

(1) 파이썬의 시각화 모듈

- 모듈마다 다양한 특징이 있어, 상황에 맞는 모듈 사용이 가능함
- 시각화는 주로 그래프를 쉽게 그릴 수 있도록 도와 줌
  - 다양한 형태의 그래프 지원

    (예) 3D, 원형, 막대그래프 등
- 대표적인 파이썬 시각화 모듈
  - Matplotlib, Seaborn, Plotly, Plotnine
  - Folium, Pyecharts, Wordcloud

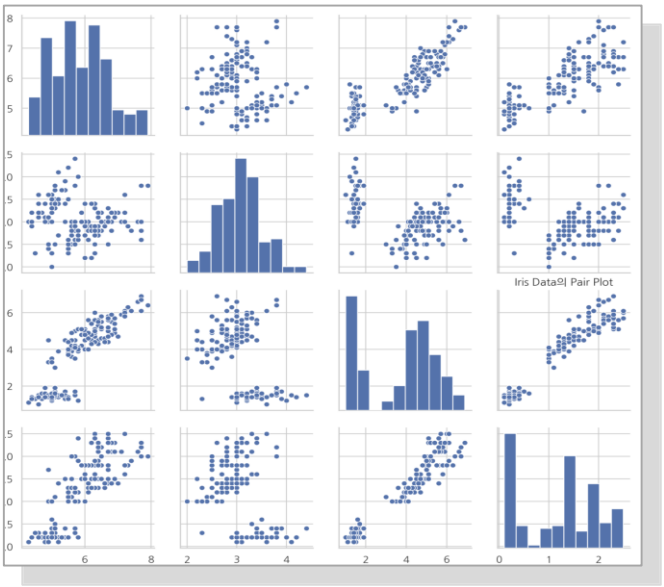
(2) Matplotlib

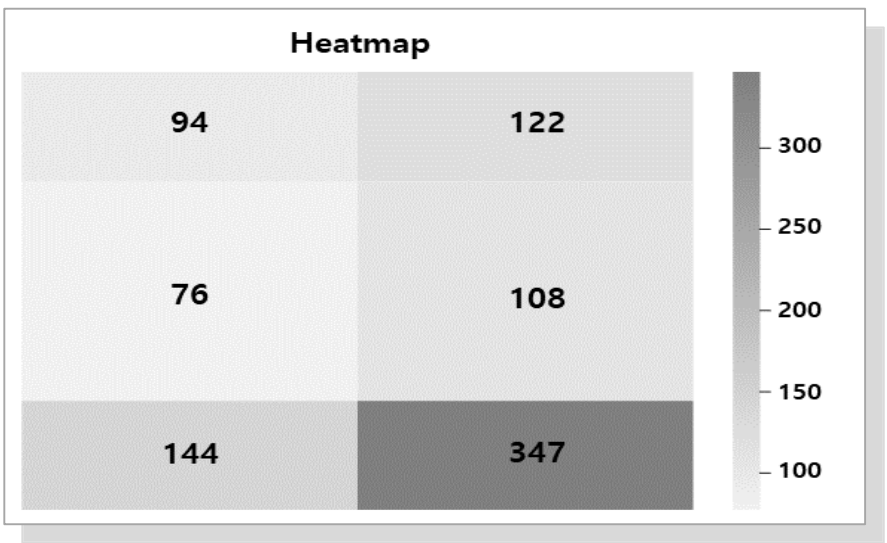
- 파이썬에서 가장 많이 사용되는 모듈 중 하나
  - MATLAB 프로그램과 유사한 인터페이스
  - 다양한 출력 형식 지원: PNG, JPG, SVG 등
  - 다양한 옵션 제공: 그래프의 종류, 축, 눈금선, 범례 등

3) 데이터 시각화를 위한 모듈의 종류

(3) Seaborn

- Matplotlib을 기반으로 통계용 기능을 추가한 시각화 패키지
  - 기본적인 시각화 기능: Matplotlib 패키지에 의존
  - 통계 기능: Statsmodels 패키지에 의존





3) 데이터 시각화를 위한 모듈의 종류

(4) Plotly

- Plotly Python: 대화형 오픈 소스 플로팅 라이브러리
  - 40개 이상의 고유한 차트 유형 지원
    (통계, 재무, 지리, 과학 및 3차원 사용 사례)

- 라이브러리 특징
  - 웹 기반 애플리케이션(Dash)에 올려 활용할 수 있음
  - 인터랙티브한 그래프를 그릴 수 있음

2. Matplotlib 모듈의 이해와 활용

1) Matplotlib 모듈의 개념

(1) 개념

- 파이썬에서 자주 활용되는 시각화 모듈

(2) 설치 방법

- pip install matplotlib 명령어를 통해 설치
  - Anaconda를 설치한 경우, 별도 설치 과정 필요 없음

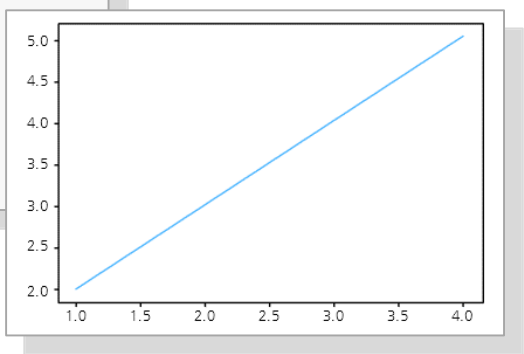
2. Matplotlib 모듈의 이해와 활용

1) Matplotlib 모듈의 개념

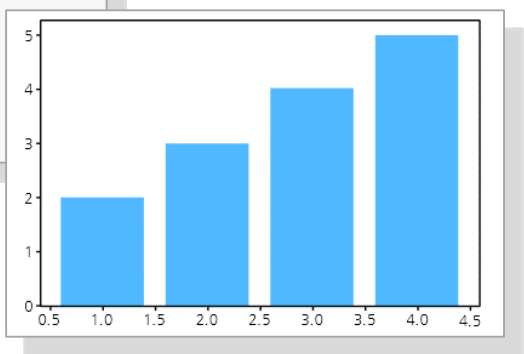
(3) 시각화 모듈 적용 예시

그래프 형태 선택 → 데이터 전달 → 데이터 출력

```
import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [2,3,4,5]
plt.plot(x,y)
plt.show()
```



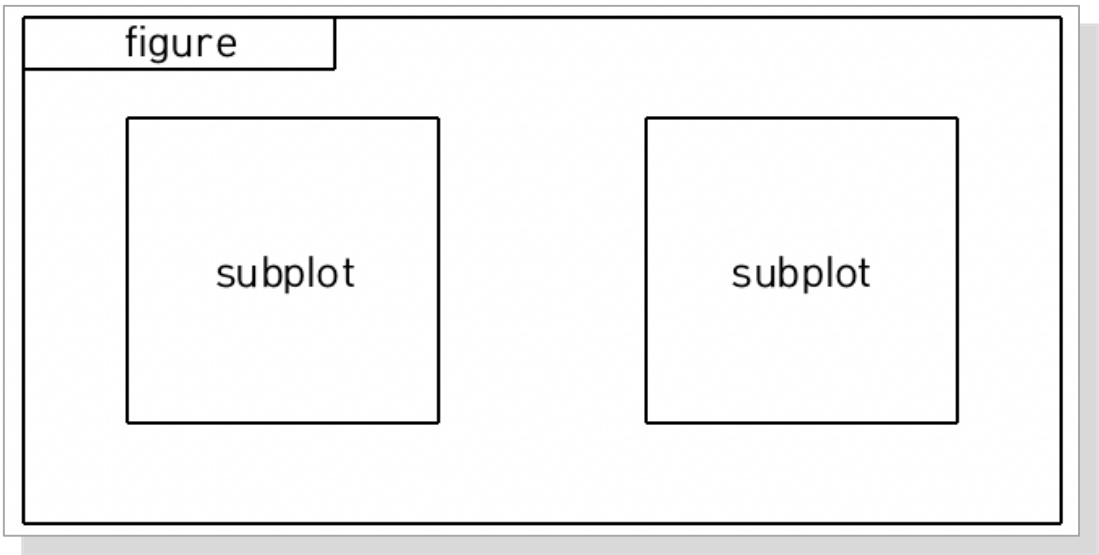

```
import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [2,3,4,5]
plt.bar(x,y)
plt.show()
```



1) Matplotlib 모듈의 개념

(4) Matplotlib 모듈을 잘 사용하려면?

- figure와 subplot의 개념 파악
  - 다양한 형태로 더욱 편리하게 활용할 수 있음
  - figure: 그래프의 영역(subplot)을 생성하기 위한 전체 틀
  - subplot: 그래프를 그리기 위한 영역

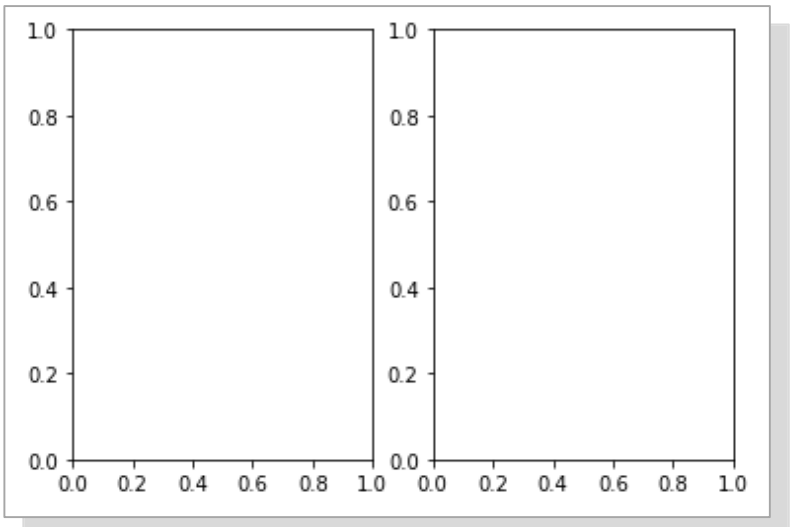


2) Matplotlib 모듈의 활용 방법

(1) add_subplot('특정 위치')

- 특정 위치에 그래프 생성
  - 특정 위치: ([총 행의 수],[총 열의 수],[subplot의 번호])를 의미함
  - 하나의 figure에 두 개의 subplot을 추가한 것
  - 121: 1행 2열에서 1열
  - 122: 1행 2열에서 2열

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes1 = figure.add_subplot(121)
axes2 = figure.add_subplot(122)

plt.show()
```



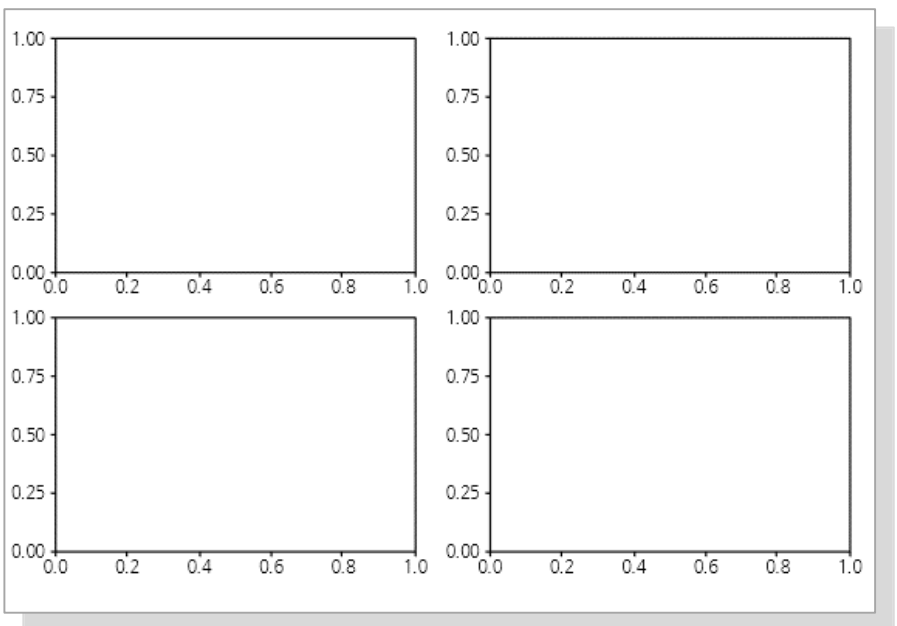
2) Matplotlib 모듈의 활용 방법

(1) add_subplot('특정 위치')

- 총 4개의 subplot
- 숫자 사이에 콤마는 넣어도 되고, 넣지 않아도 됨

  (예) 2,2,1(221): 2행 2열 1행을 의미

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes1 = figure.add_subplot(2,2,1)
axes2 = figure.add_subplot(2,2,2)
axes3 = figure.add_subplot(2,2,3)
axes4 = figure.add_subplot(2,2,4)

plt.show()
```

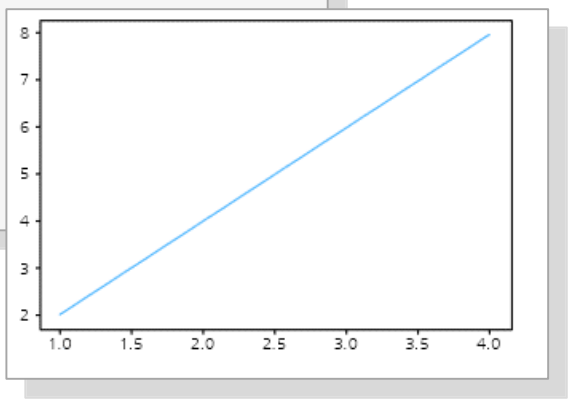


2) Matplotlib 모듈의 활용 방법

(2) plot()

- 꺾은선그래프를 그릴 때 사용
  - 시간에 따른 값의 변화량 등을 나타낼 때 적합

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot()

x = [1,2,3,4]
y = [2,4,6,8]

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



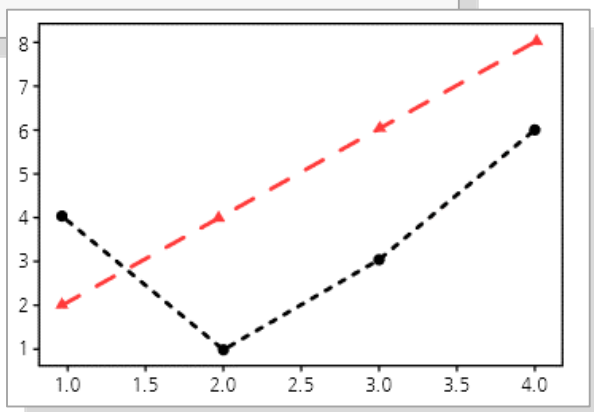
- 다양한 데이터 자료형 적용 가능
  - 파이썬의 리스트
  - Numpy의 Array
  - Pandas의 Series

2) Matplotlib 모듈의 활용 방법

(2) plot()

- 꺾은선그래프에 옵션 추가
  - linestyle: solid, dotted, dashdotted, dashed 등 선의 형태
  - color: black, red, blue, green 등 선의 색깔
  - marker: o, >, ^ 등 값에 표시하는 마커의 형태

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

figure = plt.figure()
axes = figure.add_subplot(111)

x = [1,2,3,4]
y = [2,4,6,8]

x2 = np.array([1,2,3,4])
y2 = np.array([4,1,3,6])
axes.plot(x,y, color='red',linestyle='dashed',marker='^')
axes.plot(x2,y2,color='k',linestyle='dotted',marker='o')
plt.show()
```

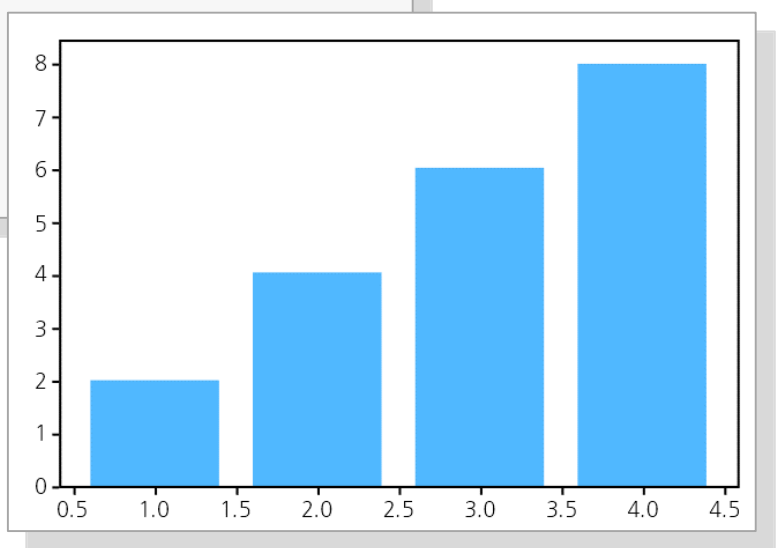


2) Matplotlib 모듈의 활용 방법

(3) bar()

- 막대그래프 생성
  - 빈도 수, 순위 등을 나타낼 때 적합

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)

x = [1,2,3,4]
y = [2,4,6,8]

axes.bar(x,y)

plt.show()
```

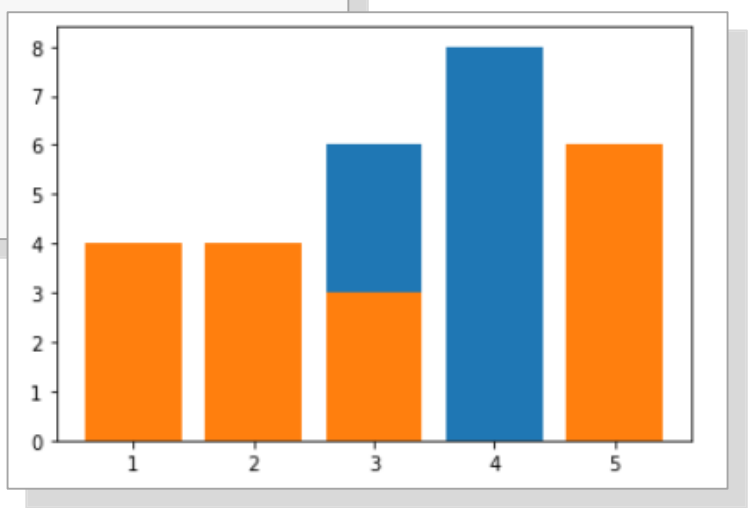


2) Matplotlib 모듈의 활용 방법

(3) bar()

- 중첩 그래프 그리기
  - 하나의 그래프 영역에 데이터를 추가하면 중첩으로 그릴 수 있음

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)

x = [1,2,3,4]
y = [2,4,6,8]

x2 = [1,2,3,5]
y2 = [4,4,3,6]
axes.bar(x,y)
axes.bar(x2,y2)
plt.show()
```



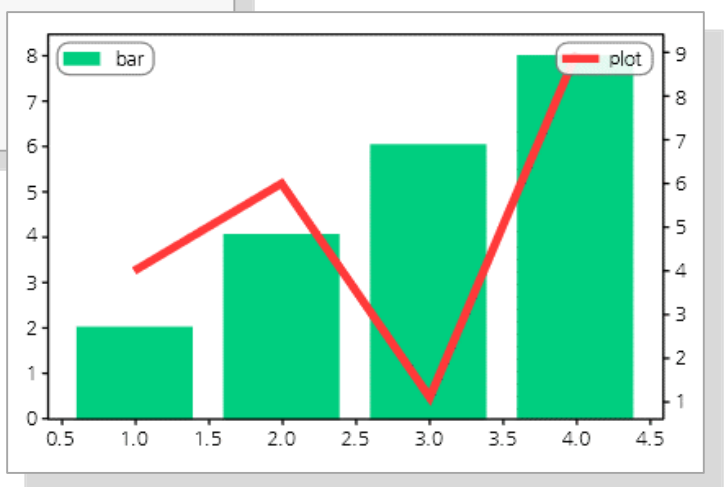
➡ 데이터 값이 중복되는 경우 합쳐져서 보이지 않게 된다는 것을 주의!

2) Matplotlib 모듈의 활용 방법

(4) twinx(), twiny()

- x축, y축을 기준으로 2개의 다른 축을 가질 경우, 그래프 표현 가능
  - twinx: x축을 기준으로 두 개의 bar와 plot 형태의 y축 그래프 생성
  - label, legend: 범례 생성

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)
axes2 = axes.twinx()

x = [1,2,3,4]
y = [2,4,6,8]
x2 = [1,2,3,4]
y2 = [4,6,1,9]

axes.bar(x,y,color='green',label='bar')
axes2.plot(x2,y2,color='r',label='plot')

axes.legend()
axes2.legend(loc=1)
plt.show()
```



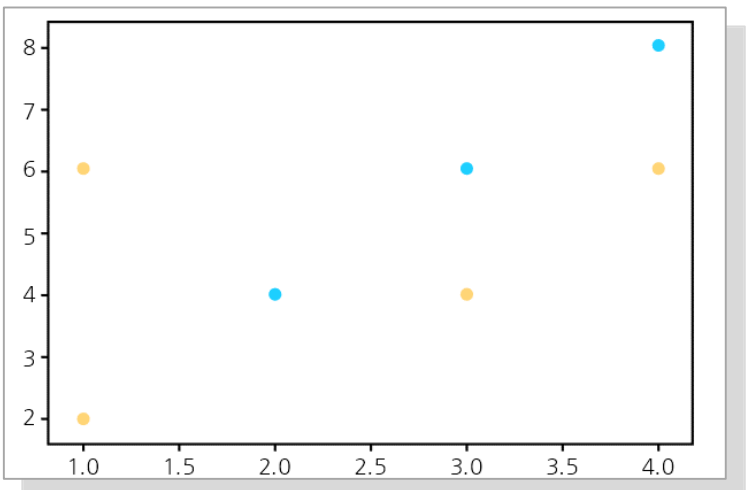
➡ 범례가 동일 위치에 생성되므로, loc 옵션을 통해 하나의 범례를 다른 위치로 이동

2) Matplotlib 모듈의 활용 방법

(5) scatter()

- 점그래프(산점도 그래프) 생성
  - 데이터의 분포를 파악하기 쉬움

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)

x = [1,2,3,4]
y = [2,4,6,8]
x2 = [1,1,3,4]
y2 = [6,2,4,6]
axes.scatter(x,y)
axes.scatter(x2,y2)
plt.show()
```

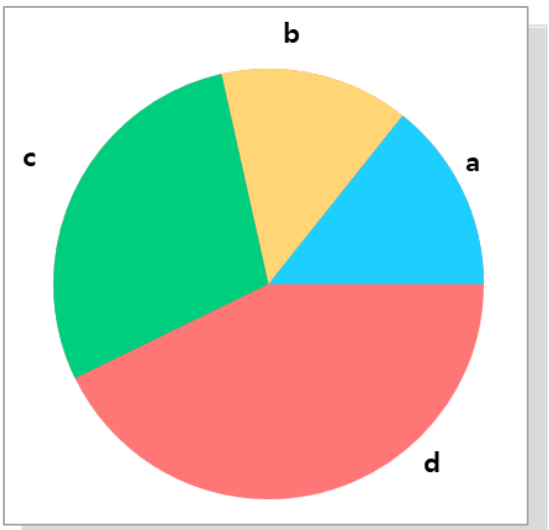


2) Matplotlib 모듈의 활용 방법

(6) pie()

- 원그래프 생성
  - 데이터 간의 비율을 파악하기 쉬움
  - label, data: 그래프를 그릴 수 있음
  - figsize 옵션: 그래프의 크기를 별도로 설정할 수 있음

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)

label = ['a','b','c','d']
data = [1,1,2,3]

axes.pie(data,labels=label)
plt.show()
```

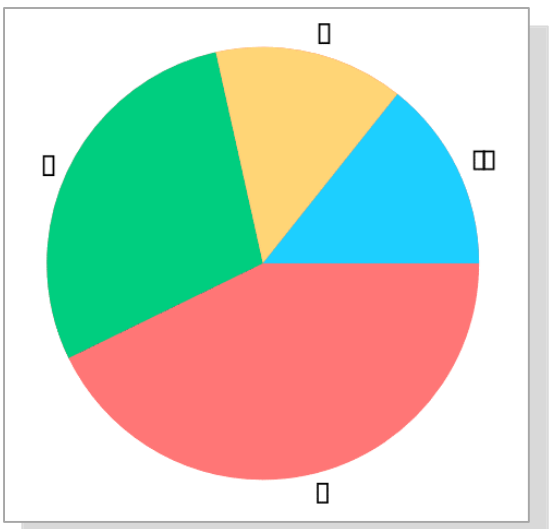


3) 한글 폰트 설정하기

(1) 한글 폰트 설정의 필요성

- Matplotlib 모듈로 출력한 그래프 라벨 값의 글자가 깨져 보임
  - 기본 폰트가 한글을 지원하지 않기 때문임

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)

label = ['하나','둘','셋','넷']
data = [1,1,2,3]

axes.pie(data,labels=label)
plt.show()
```



➡ 폰트를 변경하여 활용할 수 있음

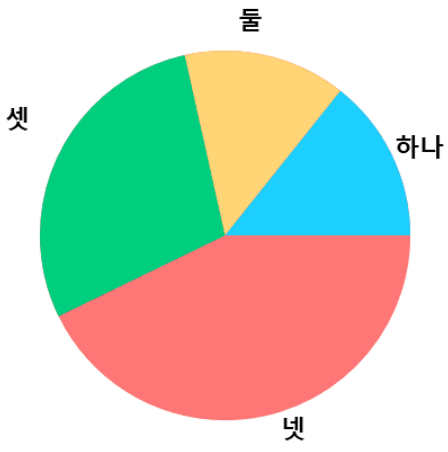
3) 한글 폰트 설정하기

(2) 한글 폰트 설정 방법

- PC에 설치된 폰트를 지정하여 활용

```
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
font_path = "/Library/Fonts/NanumGothic.otf"
font = font_manager.FontProperties(fname=font_path).get_name()
rc('font', family=font)
```

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)

label = ['하나','둘','셋','넷']
data = [1,1,2,3]

axes.pie(data,labels=label)
plt.show()
```

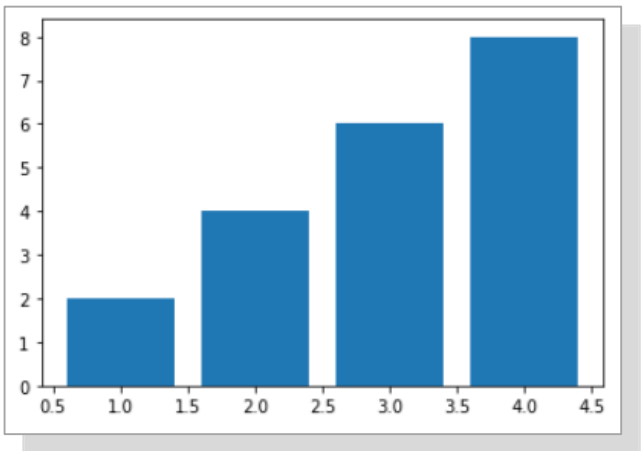


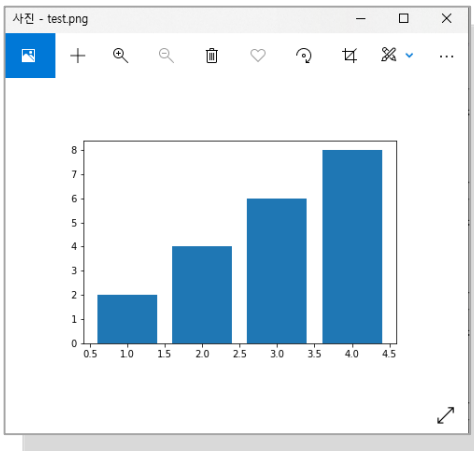
4) 시각화 결과 내보내기

(1) 시각화 결과 저장하기

- 파이썬에서 작성한 결과를 이미지 파일로 저장할 수 있음
  - savefig 함수 활용

```
import matplotlib.pyplot as plt

figure = plt.figure()
axes = figure.add_subplot(111)

x = [1,2,3,4]
y = [2,4,6,8]

axes.bar(x,y)

plt.savefig('test')
```





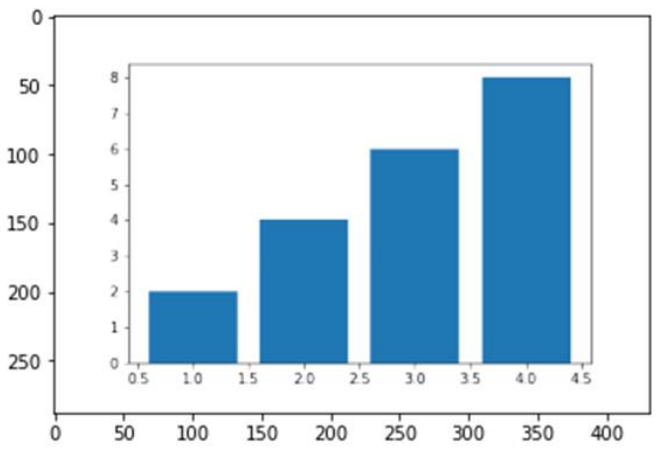
4) 시각화 결과 내보내기

(2) 이미지 파일을 불러와 출력하기

- 이미지를 파이썬으로 불러와 그래프 그리기
  - imread: 이미지를 불러옴
  - imshow: 그래프 출력

```
import matplotlib.pyplot as plt
import matplotlib.image as img

image = img.imread('test.png')
plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x1f0047a9730>
```



3. Wordcloud 모듈의 이해와 활용

1) Wordcloud 모듈의 개념

- 데이터에서 얻은 단어, 태그들을 분석하여 중요도나 인기도 등을 고려하여 시각적으로 표현하는 것

➡ 한눈에 중요 키워드 등을 파악할 수 있는 장점이 있음

- 파이썬에도 Wordcloud 모듈을 활용해 데이터를 시각화 할 수 있음
  - 설치 명령어: pip install wordcloud 또는 conda install -c conda-forge wordcloud

3. Wordcloud 모듈의 이해와 활용

2) Wordcloud 모듈의 활용 방법

(1) Wordcloud를 통한 시각화 방법

- generate_from_text 함수
  - WordCloud 객체를 생성한 뒤 generate_from_text 함수 활용

```
import wordcloud
text = "python python hello data"
wc = wordcloud.WordCloud()
wc.generate_from_text(text)
wc.to_image()
```

3. Wordcloud 모듈의 이해와 활용

2) Wordcloud 모듈의 활용 방법

(1) Wordcloud를 통한 시각화 방법

- generate_from_frequencies 함수
  - 딕셔너리 자료형의 데이터도 시각화할 수 있음

```
import wordcloud
text = {"python":10,'hello':1,'data':4}
wc = wordcloud.WordCloud()
wc.generate_from_frequencies(text)
wc.to_image()
```

3. Wordcloud 모듈의 이해와 활용

3) Wordcloud 모듈 옵션

(1) 한글 폰트 설정

- 폰트 옵션을 활용하여 직접 경로를 바꿔 사용할 수 있음

```
import wordcloud
text = "파이썬 파이썬 시각화 데이터분석"
wc = wordcloud.WordCloud()
wc.generate_from_text(text)
wc.to_image()
```

```
import wordcloud
text = "파이썬 파이썬 시각화 데이터분석"
wc = wordcloud.WordCloud(font_path='C:\\Windows\\Fonts\\맑은 고딕\\malgun.ttf')
wc.generate_from_text(text)
wc.to_image()
```

3. Wordcloud 모듈의 이해와 활용

3) Wordcloud 모듈 옵션

(2) 다양한 모듈 옵션

- background_color: 배경 색 변경
- to_image(): 이미지로 바로 출력

```
import wordcloud
text = "파이썬 파이썬 시각화 데이터분석"
wc = wordcloud.WordCloud(font_path='/Library/Fonts/NanumGothic.
otf', background_color='pink')
wc.generate_from_text(text)
wc.to_image()
```

3. Wordcloud 모듈의 이해와 활용

3) Wordcloud 모듈 옵션

(2) 다양한 모듈 옵션

- to_file(): 해당 파일로 출력

```
import wordcloud
text = "파이썬 파이썬 시각화 데이터분석"
wc = wordcloud.WordCloud(font_path='/Library/Fonts/NanumGothic.otf', background_color='pink')
wc.generate_from_text(text)
wc.to_file('test.jpg')
```

```
<wordcloud.wordcloud.WordCloud at 0x7fefe1cf7890>
```

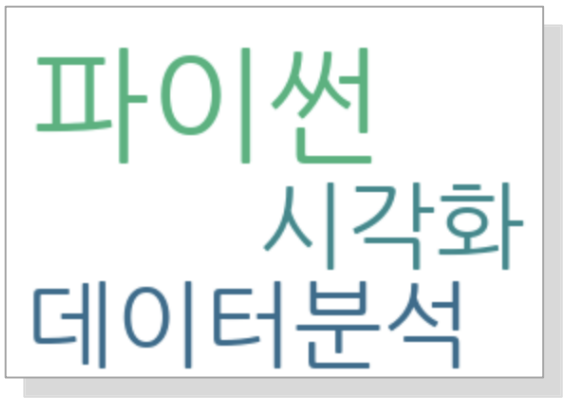
3. Wordcloud 모듈의 이해와 활용

3) Wordcloud 모듈 옵션

(2) 다양한 모듈 옵션

- Stopwords: 제외하고 싶은 단어들을 리스트 형태로 추가

```
import wordcloud
text = "파이썬 파이썬 시각화 데이터분석 오류 데이터"
wc = wordcloud.WordCloud(font_path='font/NanumGothic.ttf', back
ground_color='white',stopwords=['오류','데이터'])
wc.generate_from_text(text)
wc.to_image()
```



- 그 외 폰트 사이즈, 마스크, 최대 단어 수 등 다양한 옵션 활용 가능