THU

시각화 Data Visualization

시각화
: 데이터 나 분석 결과를 알아보기 쉽도록 해주는 작업

〈 방법 〉

1. 프로그래밍 언어를 이용

—〉 Server Side
(Java , python, C#)
시각화 결과를 만들고 클라이언트에서 출력

—〉 Client Side
Web 의 경우 Java Script
Web 을 제외한 Application
: python, R , MFC(예전 증권거래에서 사용), C#

2. 시각화 Tool 을 사용
타블로, Microsoft 의 BI

〈 시각화를 하는 이유 〉
1. 데이터 분석을 하기전 전처리 과정에서
데이터의 분포를 확인하기 위해서

2. 분석결과를 알아보기 쉽도록 하기 위해서

〈 시각화 Package 〉

matplotlib
: anaconda를 설치하면 같이 설치

pandas
: 통계 분석에 많이 사용되는 package
seaborn(몇 개의 샘플 데이터 와 조금더 화려한 기능 제공)
matplotlib 기반의 시각화 기능을 제공하는 package

folium
: 지도 출력

통계 나 AI(Machine Learning 〈- Deep Learning) package등에서도 제공

* python 이나 R은 영문기반 Linux 시스템에서
개발이 진행된 Open Source 개념을 가진다

즉, 한글은 기본적으로 바로 출력 되지 않는다.

〈 시각화 작업을 위한 준비 〉

데이터를 이해하고 , 추출하는
비지니스 Insight 도출에 가장 알맞는 시각화 방법을 찾아내는 것

web Programming 에서도 데이터 추출 시
table 보다는 데이터를 보고 알맞은 시각화 방법을 제시하는 것이 중요

——〉 Productive Web , Responsive Web 등의 출현 계기

mataplotlib

python 시각화에서 가장 많이 사용하는 package 중 하나

anaconda를 이용해서 python을 설치하면 기본적으로 제공(python 기본에는 존재하지 않음)

〈 Import 〉
—〉 import matplotlib.pyplot as plt

〈 작업 순서 〉
——〉 1. import

——〉 2. 영역 생성
: plt.figure()

——〉 3. 시각화 작업

——〉 4. 출력
: plt.show()

```
# 시각화 package

# 1. import
import matplotlib.pyplot as plt


# 꺽은선 그래프

# 2. 그래프를 그릴 영역
plt.figure()

# 3. 옵션을 설정
plt.plot([100, 1000, 300])

# 4. 출력
plt.show()
```

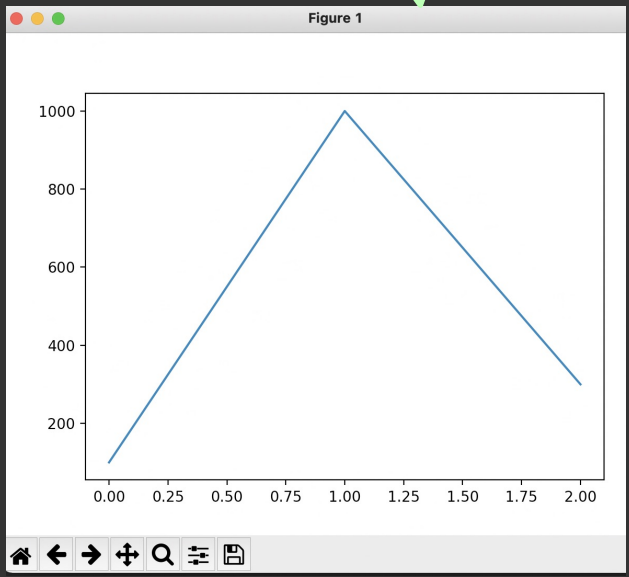
〈 plot 〉
: 꺽은선 그래프나 산점도 형태를 만들어주는 함수

특징
1. 꺽은선 그래프는 시계열 데이터에서 데이터의 추세를 나타낼 때 많이 사용
* 시계열 데이터 : 연속적인 데이터

2. 산점도 그래프는 데이터의 분포를 나타내려 할 때 주로 사용

3. 데이터는 1개의 그룹이나 2개의 그룹을 제공해서 시각화

4. 1개의 그룹을 설정하면 x 축은 index,
   2개의 그룹을 제공하면 x 축과 y 축으로 설정



〈 Option 〉

1. 크기변경

figure 함수의 figsize=(가로, 세로)

* 단위는 inch

2. 눈금표현

grid
——〉 grid()

3. 축의 lable 설정

——〉 xlabel, ylabel 함수 사용

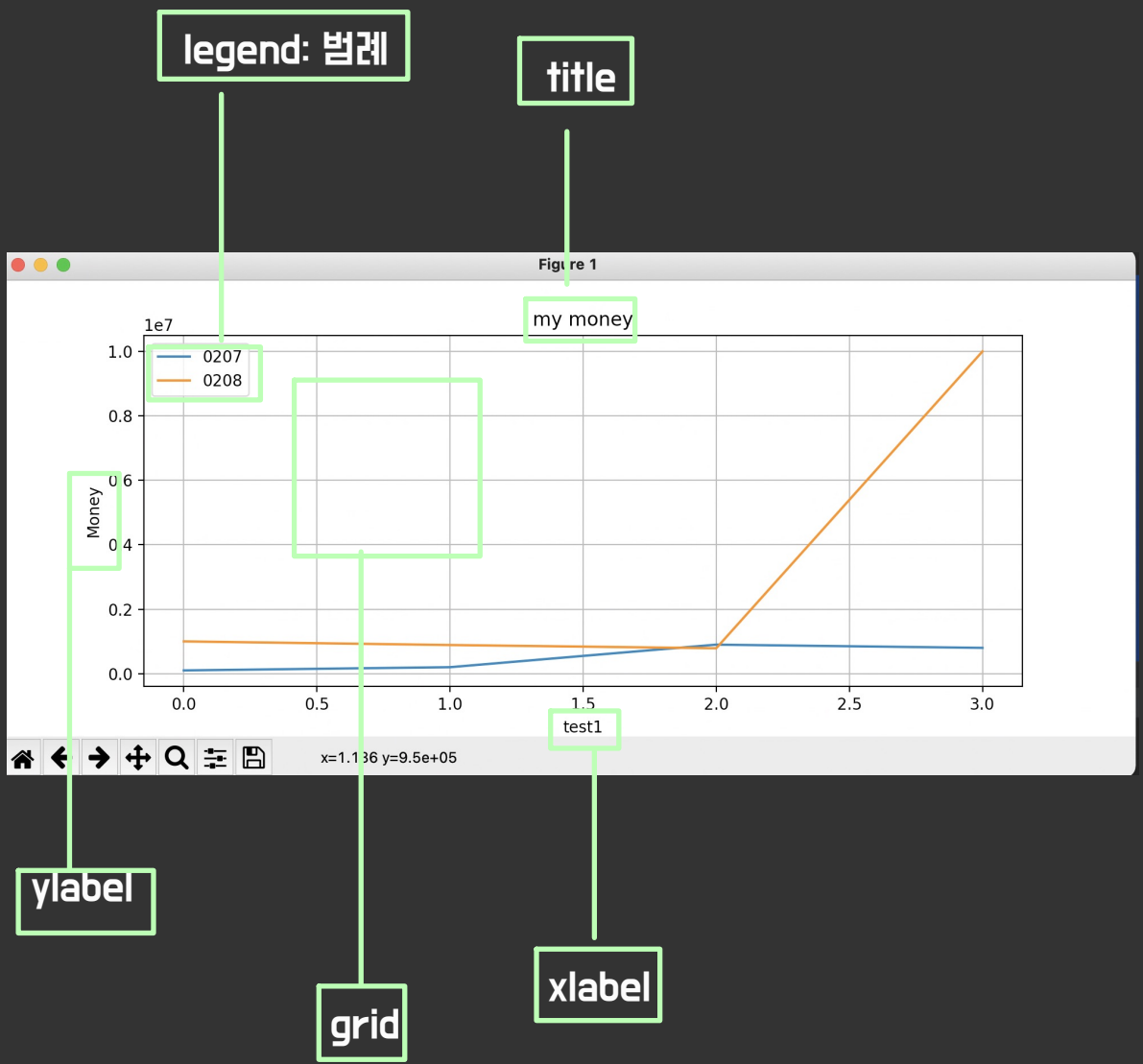
4. 그래프 제목

——〉 title 함수

5. 범례

: lable 옵션에 text 를 설정
legend 함수를 호출하면 범례 생성

```
# 옵션 변경

# 데이터 생성
s1 = [100000, 200002, 900000, 800000]
s2 = [999999, 888288, 787878, 10000000]

# 그래프 크기 설정
plt.figure(figsize=(10, 4))

# 그래프를 그리기
plt.plot(s1, label="0207")
plt.plot(s2, label="0208")
plt.grid()

plt.xlabel('test1')
plt.ylabel('Money')

plt.title('my money')

plt.legend()

# 그래프를 화면에 출력
plt.show()
```



〈 꺾은선 graph의 옵션 〉

1. color
: 색상 변경 가능
  # RGB 형태로 대입하거나
  # 알파벳을 대입

2. lw
: 선 두께 변경

3. xlim, ylim
: 축의 범위를 설정하는데 최소값과 최대값 설정

4. xticks, yticks
: 축에 출력되는 범위를 range를 이용해서 만들거나
    list로 대입

5. labels
: 축의 제목을 설정할 수 있다 (text)
: 출력 방향을 설정할 수 있다 (rotation)

6. xticklabels, yticklabels
: 눈금의 문자열을 설정

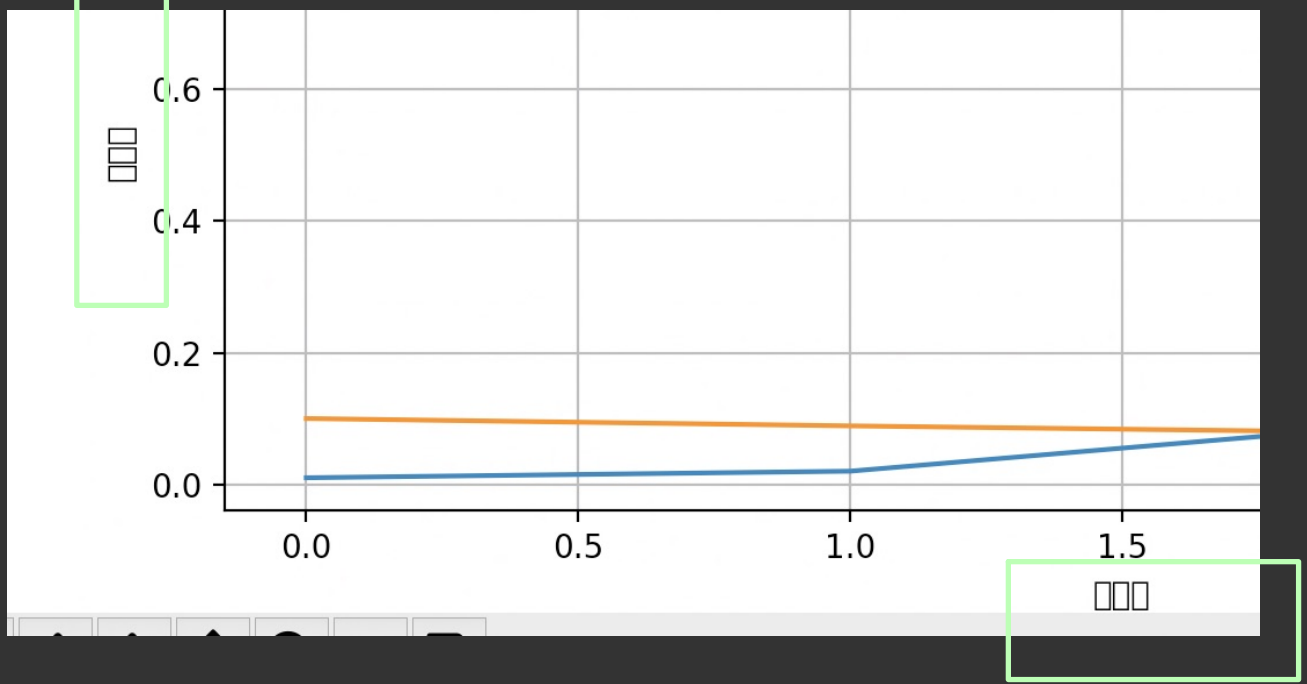
7. linestyle
: 선의 모양을 변경

〈 한글 이나 음수 출력〉

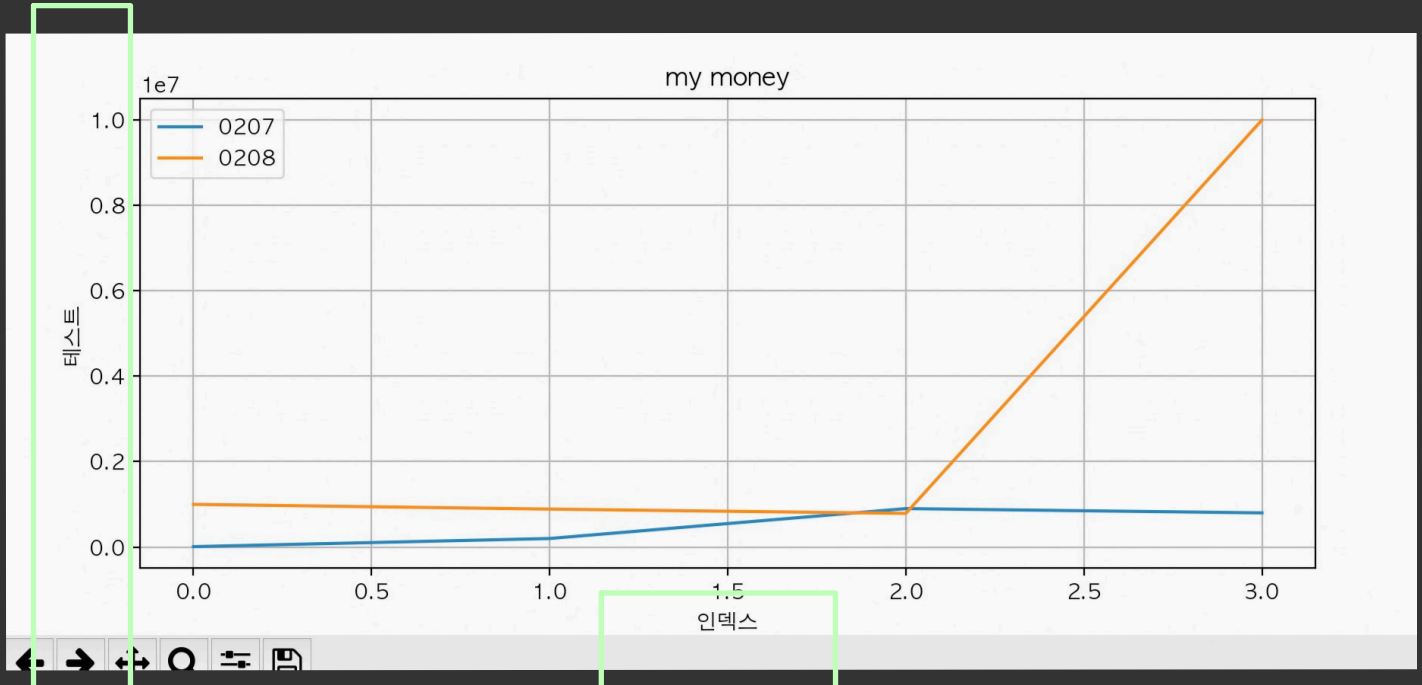
기본 설정 그대로
——〉 한글과 음수는 네모 형태로 보여진다

이유?
: maplotlib 의 기본 font 는 한글과 음수를 지원하지 않음

```
# 데이터 생성
s1 = [100000, 200002, 900000, 800000]
s2 = [999999, 888288, 787878, 10000000]

# 그래프 크기 설정
plt.figure(figsize=(10, 4))

# 그래프를 그리기
plt.plot(s1, label="0207")
plt.plot(s2, label="0208")
plt.grid()

plt.xlabel('인덱스')
plt.ylabel('테스트')

plt.title('my money')

plt.legend()

# 그래프를 화면에 출력
plt.show()
```



```python
# 1. import
import matplotlib.pyplot as plt
from matplotlib import rc
rc('font', family='AppleGothic')
plt.rcParams['axes.unicode_minus'] = False

# 음수 출력
plt.rcParams['axes.unicode_minus'] = False

# #매킨토시의 경우
# rc('font', family='AppleGothic')
#
# plt.rcParams['axes.unicode_minus'] = False
# #윈도우의 경우
# elif platform.system() == 'Windows':
#     font_name = font_manager.FontProperties(fname="폰트파일경로").get_name()
#     rc('font', family=font_name)
# #우분투 리눅스의 경우
# else:
#     font_name = '/usr/share/fonts/truetype/nanum/NanumMyeongjo.ttf'
#     rc('font', family=font_name)



# 데이터 생성
s1 = [10000, 200002, 900000, 800000]
s2 = [999999, 888288, 787878, 10000000]

# 그래프 크기 설정
plt.figure(figsize=(10, 4))

# 그래프를 그리기
plt.plot(s1, label="0207")
plt.plot(s2, label="0208")
plt.grid()

plt.xlabel('인덱스')
plt.ylabel('테스트')

plt.title('my money')

plt.legend()

# 그래프를 화면에 출력
plt.show()
```



CSV 파일을 읽어서 꺾은선 그래프로 출력하기

데이터 수집

데이터를 분석에 맞도록 전처리

— CSV 파일 읽을 때 확인 내용
1. 한글 포함 여부
: encoding 설정

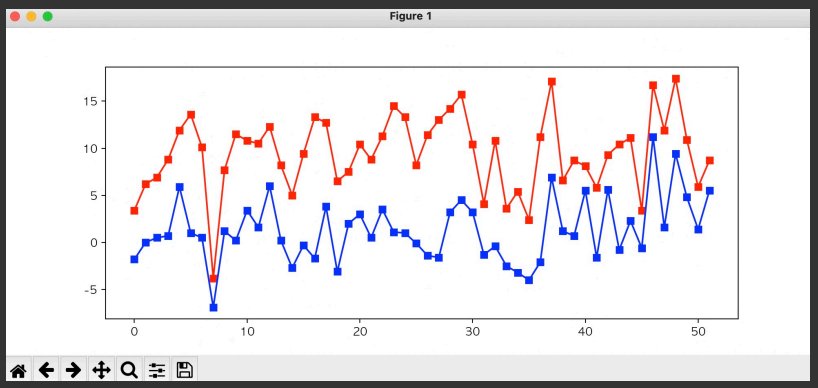
2. 구분자
: comma 가 아니면 delimiter를 설정해야 할 수 있다

3. header 포함여부
: 각 열의 제목이 존재하는지 확인 , 실제 데이터가 몇행에서 시작하는지 확인

4. 파일의 크기
: 크기가 너무 크면 한번에 읽지 못해서 분할해서 작업

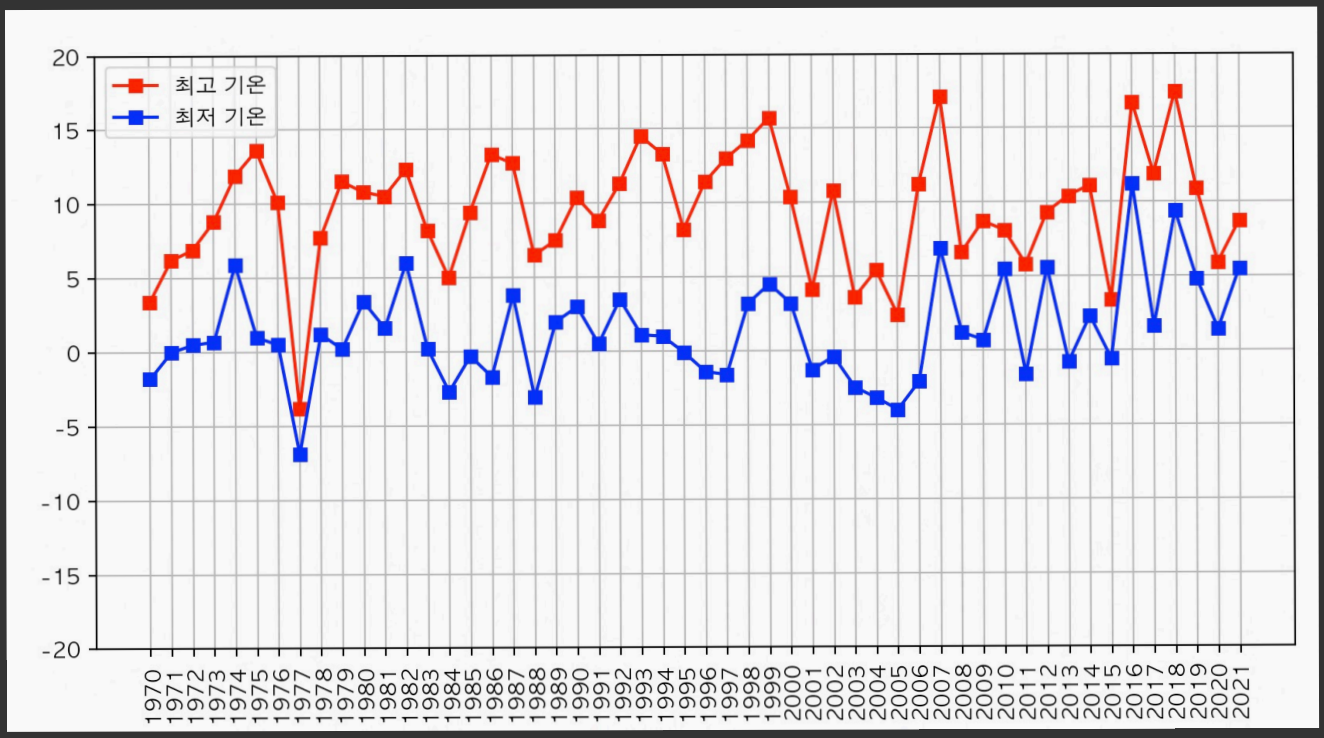
1. CSV 파일 읽기
: csv package 의 csv. reader 함수를 이용

```python
import matplotlib.pyplot as plt
from matplotlib import rc
rc('font', family='AppleGothic')
plt.rcParams['axes.unicode_minus'] = False

# 음수 출력
plt.rcParams['axes.unicode_minus'] = False

import csv

# 데이터 읽기
f = open('./mokpo.csv', encoding='ms949')
data = csv.reader(f)

# 한글 나오는지 확인용
# for row in data:
#     print(row)


# 첫번째 줄은 열의 제목이므로 제외
next(data)

print("================================================")

# 필요한 데이터 추출


# 매년 3월 4일 마다 최고 최저 기온을 저장

# 년도와 최고 기온과 최저기온을 저장할 list
year = []
high = []
low = []

for row in data:
    # 첫번째 열에 해당하는 데이터 가져오기 - 날짜
    data = row[0]
    # .을 기준으로 분할 x[0] 는 연도, x[1]은 월 , x[2]은 일
    x = data.split('.')
    if x[1] == '3' and x[2] == '4':
        # 년도를 저장
        year.append(int(x[0]))

        # 최고 기온을 저장
        high.append(float(row[-1]))

        # 최저 기온을 저장
        low.append(float(row[-2]))

# 제대로 저장되었는지 샘플을 출력
for idx in range(0, 20, 1):
    print(year[idx], ":", high[idx], ":", low[idx])


# 그래프 영역 설정
plt.figure(figsize=(10, 4))

# 그래프 생성
plt.plot(high, label="최고 기온",  color="r", linestyle='-', marker='s')
plt.plot(low, label="최저 기온",  color="b", linestyle='-', marker='s')

# 그래프 출력
plt.show()
```



————> 1970 년부터 지금까지 목포의 3월 4일 출력

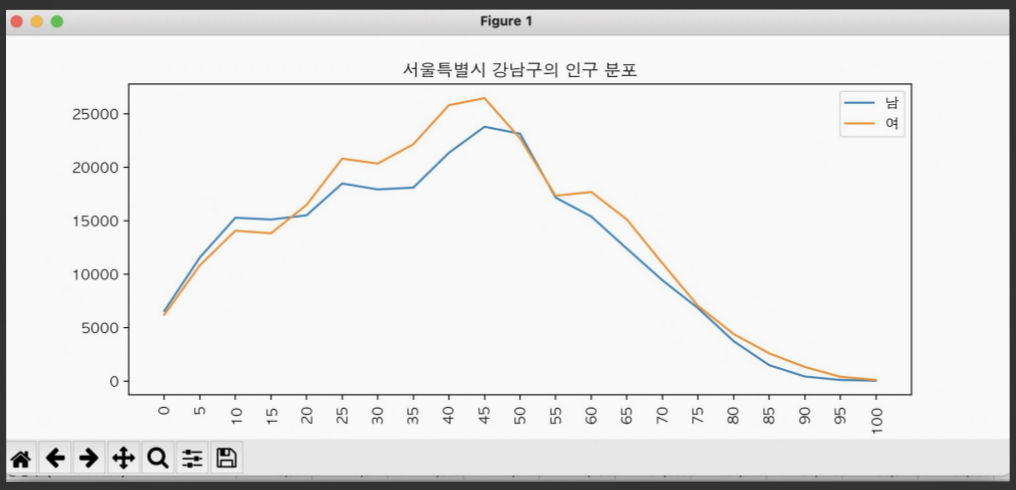
```python
# 그래프 옵션 설정

# 숫자의 범위 조절
plt.ylim(-20, 20)

# 격자 생성
plt.grid()

# 범례 설정
plt.legend()

# x 축 눈금 설정
plt.xticks(range(0, len(year), 1), year, rotation="vertical")

# 그래프 출력
plt.show()
```

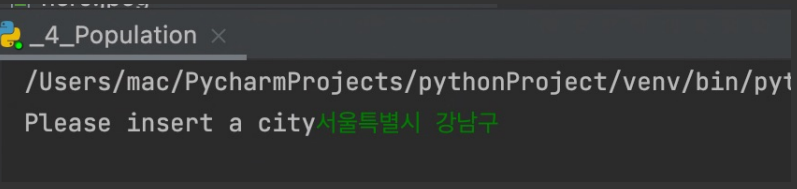


https://jumin.mois.go.kr/ageStatMonth.do#none

〈 연령별 인구 데이터 시각화 〉

다운로드 받은 파일을 편집해서 Directory 에 저장






```python
import matplotlib.pyplot as plt
import csv
from matplotlib import rc
rc('font', family='AppleGothic')
plt.rcParams['axes.unicode_minus'] = False

# 입력받은 도시의 성별, 나이별 인구 분포를 확인
f = open('./pop.csv', encoding='ms949')
data = csv.reader(f)

# 도시 이름 입력받기
city = input("Please insert a city")

# 남성 여성 따로 저장
male = []
female = []

for row in data:
    if city in row[0]:
        for i in range(24, 45, 1):
            # 남자 데이터만 정수로 변환해서 저장
            male.append(int(row[i].replace(',', '')))
            female.append(int(row[i+23].replace(',', '')))

plt.figure(figsize=(10, 4))
plt.plot(male, label="남")
plt.plot(female, label="여")
plt.xticks(range(0, 21), range(0, 105, 5), rotation="vertical")
plt.title(f"{city}의 인구 분포")
plt.legend()
plt.show()
```

산점도 , 산포도

: 점의 형태로 데이터를 시각화 하는 것을 말함

데이터의 분포 나 상관관계 파악을 위해서 사용
scatter 함수를 사용

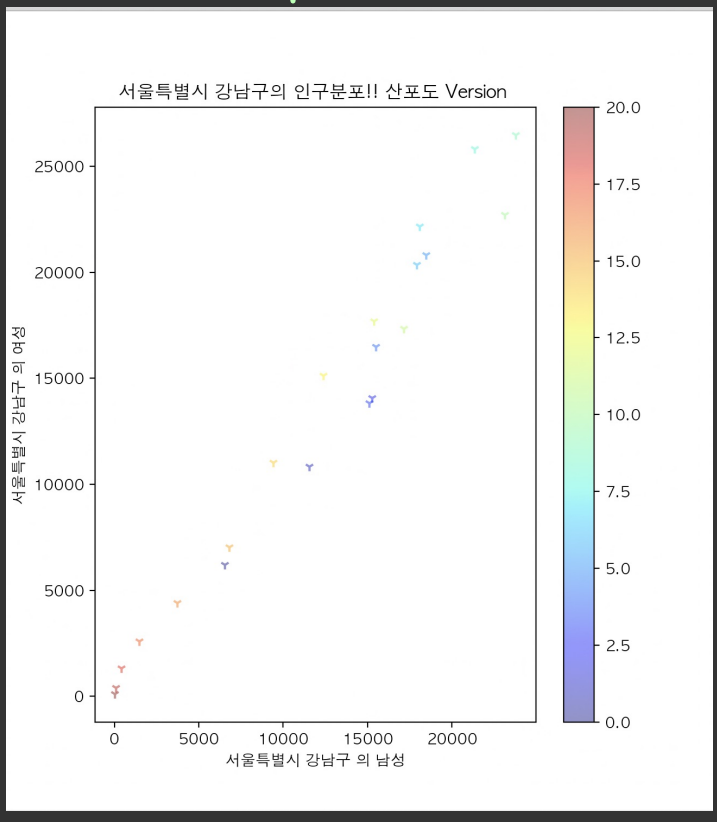
* 상관관계 파악
: 하나가 증가할 때, 다른 하나가 증가하는지 또는 감소하는지 파악

2개의 데이터를 비교하는 용도로 사용하는데
항목이 많아지면 산점도에 다른차트 추가하거나
여러 개의 삼전도를 같이 그리기도 하고 레이더 차트를 만들기도 함

—〉 colorbar 함수를 이용해서 색상 바 출력 가능

```
import matplotlib.pyplot as plt
import csv
from matplotlib import rc
rc('font', family='AppleGothic')
plt.rcParams['axes.unicode_minus'] = False

# 입력받은 도시의 성별, 나이별 인구 분포를 확인
f = open('./pop.csv', encoding='ms949')
data = csv.reader(f)

# 도시 이름 입력받기
city = input("Please insert a city")

# 남성 여성 따로 저장
male = []
female = []

for row in data:
    if city in row[0]:
        for i in range(24, 45, 1):
            # 남자 데이터만 정수로 변환해서 저장
            male.append(int(row[i].replace(',', '')))
            female.append(int(row[i+23].replace(',', '')))


# 산포 그리기
#  색상은 20가지를 쓸것이다
# marker 는 점 모양
plt.figure(figsize=(5, 12))
plt.scatter(x=male, y=female, marker='1', c=range(0, 21, 1),
            alpha=0.5, cmap='jet')

plt.colorbar()
plt.title(f"{city}의 인구분포!! 산포도 Version ")
plt.xlabel(f'{city} 의 남성')
plt.ylabel(f'{city} 의 여성')


plt.show()
```

```
_5_Scatter
/Users/mac/PycharmProjects/pythonProject/venv/bin/python /Users/mac/Pycha
Please insert a city서울특별시 강남구
```



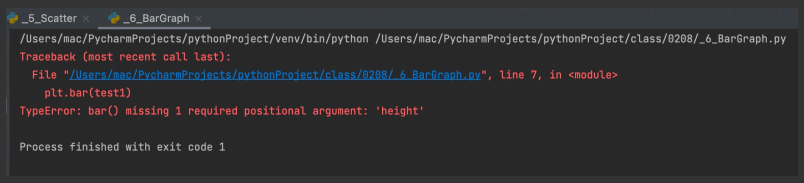
막대 그래프

〈 사용 용도 〉
다른 대상 과의 크기 비교나 강조를 해야 하는 경우 사용
자료를 구성하는 요소의 차이를 한 눈에 알아볼 수 있도록

〈 생성 방법 〉
bar 함수를 이용해서 수직 막대 그래프를 그리고
barh 함수를 이용해서 수평 막대 그래프를 그림

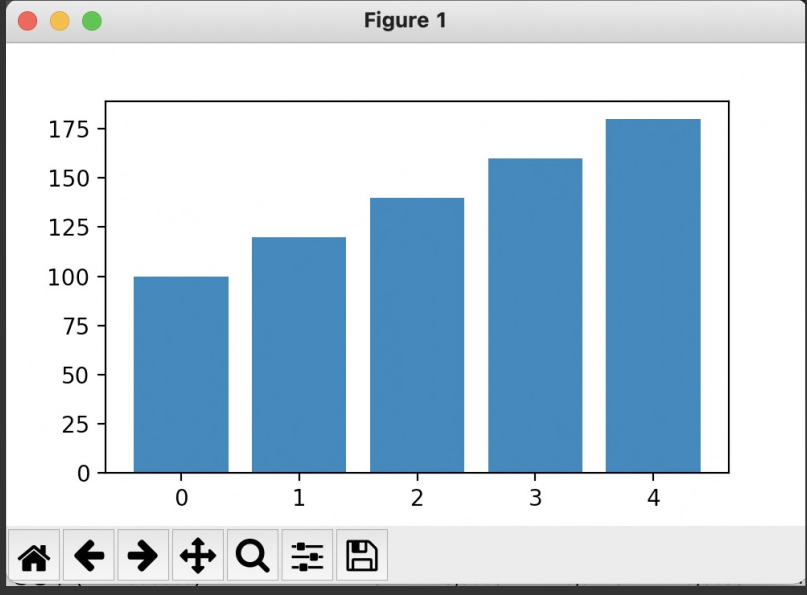
* width 속성—〉 막대의 너비 조절
2개 이상의 항목을 같이 출력 ——〉 width 를 적절히 조절

```python
import matplotlib.pyplot as plt

test1 = [100, 120, 140, 160, 180]
test2 = [50, 90, 120, 180, 360]

plt.figure(figsize=(5, 3))
plt.bar(test1)
plt.show()
```

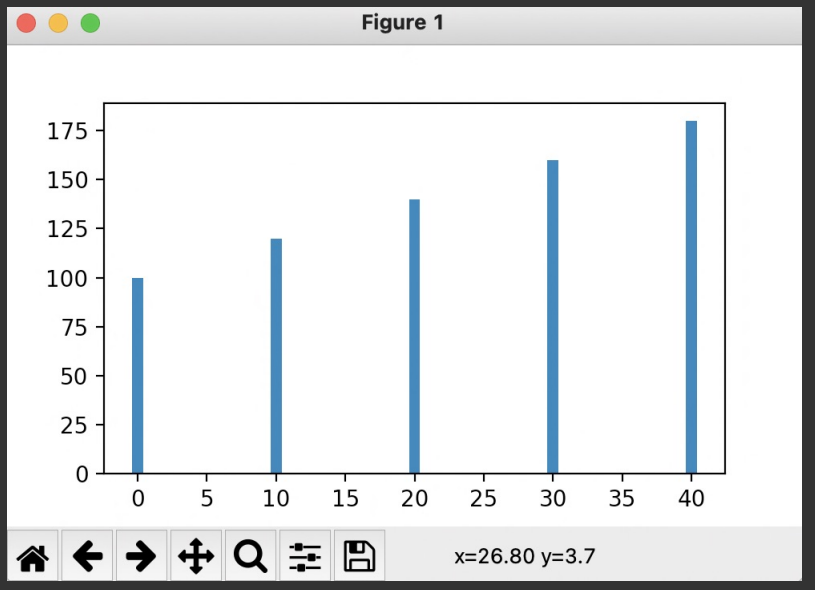


———〉 높이 속성을 지정해야한다고 한다.

```python
import matplotlib.pyplot as plt

test1 = [100, 120, 140, 160, 180]
test2 = [50, 90, 120, 180, 360]

plt.figure(figsize=(5, 3))
plt.bar(range(0, 5, 1), test1)
plt.show()
```

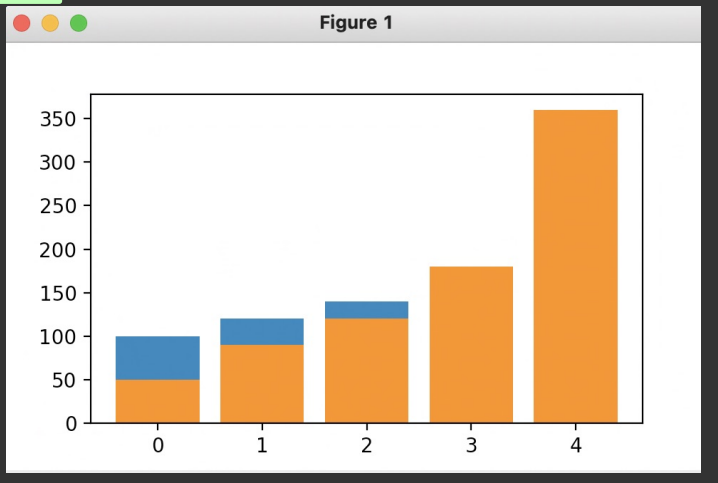



```python
import matplotlib.pyplot as plt

test1 = [100, 120, 140, 160, 180]
test2 = [50, 90, 120, 180, 360]

plt.figure(figsize=(5, 3))
plt.bar(range(0, 50, 10), test1)
plt.show()
```



——〉 막대그래프 하나 더 추가하면 겹쳐져서 나타낸다

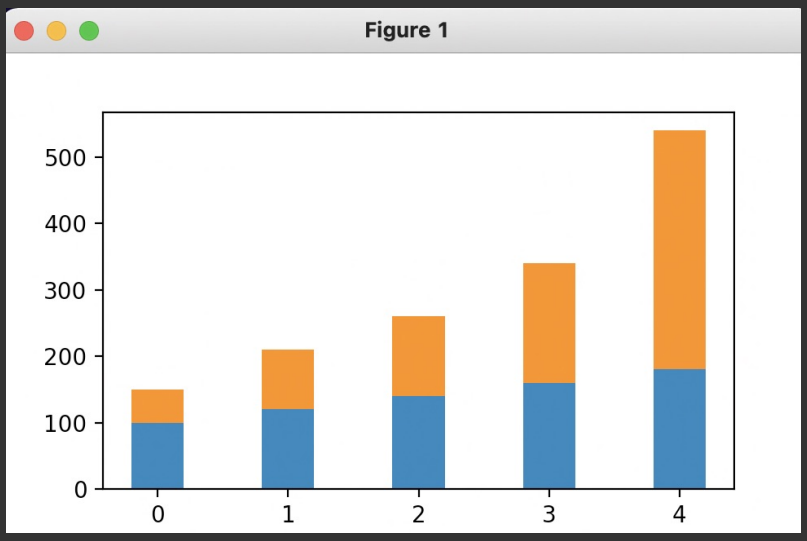
```python
import matplotlib.pyplot as plt

test1 = [100, 120, 140, 160, 180]
test2 = [50, 90, 120, 180, 360]

plt.figure(figsize=(5, 3))
plt.bar(range(0, 5, 1), test1)
plt.bar(range(0, 5, 1), test2)
plt.show()
```



——〉 이 경우 width 를 사용한다.

width 와 bottom 사용

```
import matplotlib.pyplot as plt

test1 = [100, 120, 140, 160, 180]
test2 = [50, 90, 120, 180, 360]

plt.figure(figsize=(5, 3))
plt.bar(range(0, 5, 1), test1, width=0.4)
plt.bar(range(0, 5, 1), test2, width=0.4, bottom=test1)
plt.show()
```



파이 그래프

: 전처리과정에서 자주 실행되지 않음

기여도를 표시하기 위해서 많이 사용
——〉 결과 출력 용도

〈 pie 함수 이용 〉

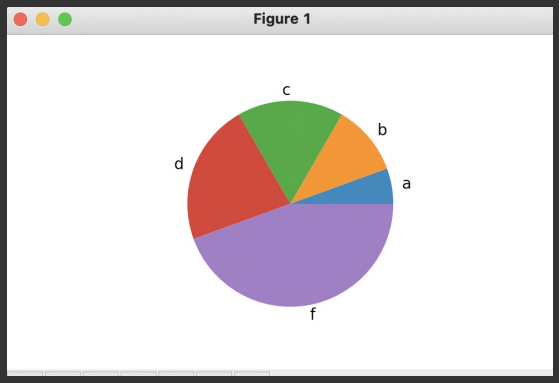
1. labels 옵션
: 레이블 출력 가능

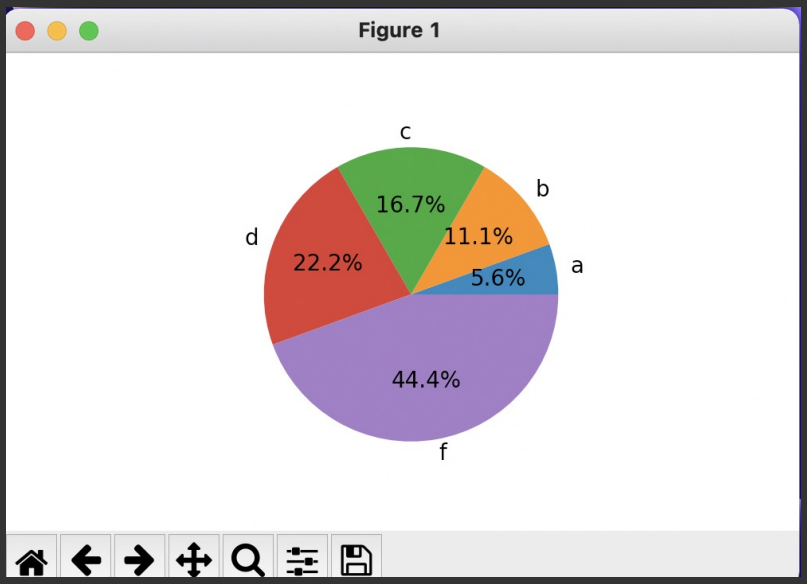
2. colors 옵션
: 색상 조절 가능

3. explode 옵션
: 분할 가능

4. autopct 옵션
: 백분율 출력 가능

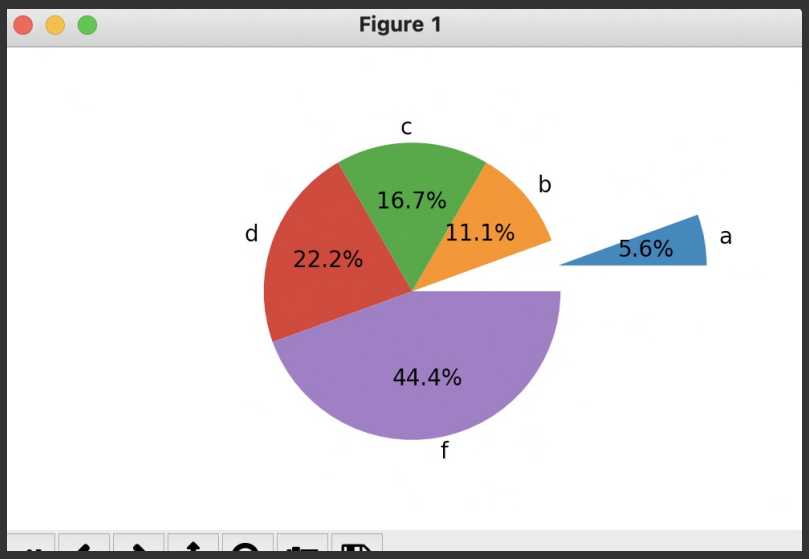
```
import matplotlib.pyplot as plt


data0 = [100, 200, 300, 400, 800]
# labels  과 함께 출력 // 데이터 개수와 맞아야함
labels = ['a', 'b', 'c', 'd', 'f']

plt.figure(figsize=(5, 3))

# 파이그래프
plt.pie(data0, labels=labels)

# 파이그래프 출력
plt.show()
```



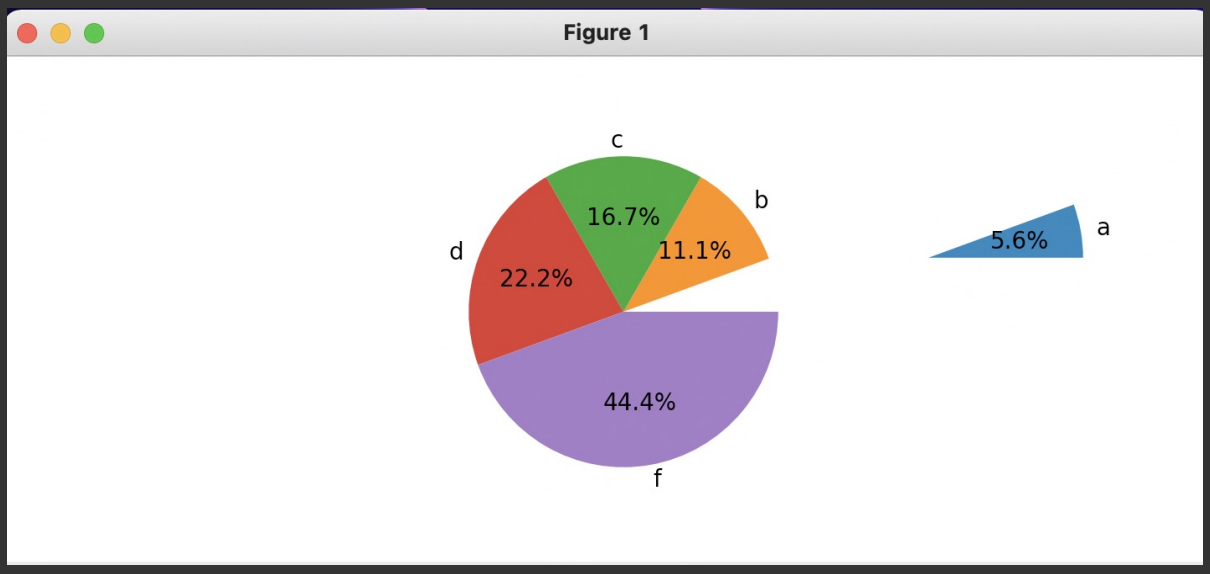



```
import matplotlib.pyplot as plt


data0 = [100, 200, 300, 400, 800]
# labels  과 함께 출력 // 데이터 개수와 맞아야함
labels = ['a', 'b', 'c', 'd', 'f']

plt.figure(figsize=(5, 3))

# 파이그래프
# % 를 부여
# 추출 구현
plt.pie(data0, labels=labels, autopct='%.1f%%', explode=(1.0, 0, 0, 0, 0),
        colors=["pink", "orange", "yellow", "gray", "green"])

# 파이그래프 출력
plt.show()
```

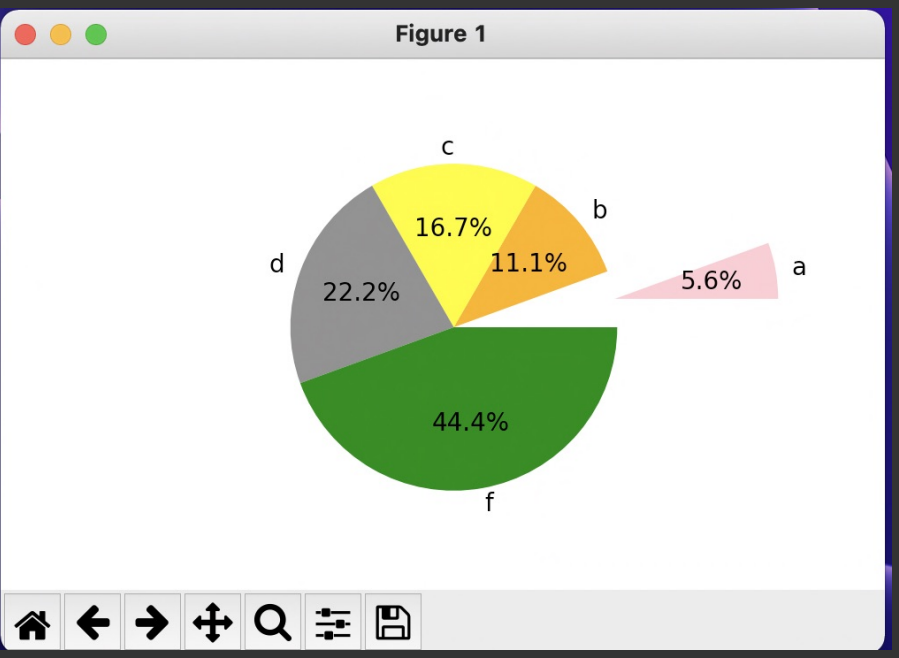
```python
import matplotlib.pyplot as plt


data0 = [100, 200, 300, 400, 800]
# labels  과 함께 출력 // 데이터 개수와 맞아야함
labels = ['a', 'b', 'c', 'd', 'f']

plt.figure(figsize=(5, 3))

# 파이그래프
# % 를 부여
# 추출 구현

plt.pie(data0, labels=labels, autopct='%.1f%%', explode=(1.0, 0, 0, 0, 0),
        colors=["pink", "orange", "yellow", "gray", "green"])

# 파이그래프 출력
plt.show()
```




```python
import matplotlib.pyplot as plt


data0 = [100, 200, 300, 400, 800]
# labels  과 함께 출력 // 데이터 개수와 맞아야함
labels = ['a', 'b', 'c', 'd', 'f']

plt.figure(figsize=(5, 3))

# 파이그래프
# % 를 부여
# 추출 구현

plt.pie(data0, labels=labels, autopct='%.1f%%', explode=(5.0, 0, 0, 0, 0),
        colors=["pink", "orange", "yellow", "gray", "green"])

# 파이그래프 출력
plt.show()
```



```python
import matplotlib.pyplot as plt


data0 = [100, 200, 300, 400, 800]
# labels  과 함께 출력 // 데이터 개수와 맞아야함
labels = ['a', 'b', 'c', 'd', 'f']

plt.figure(figsize=(5, 3))

# 파이그래프
# % 를 부여
# 추출 구현
plt.pie(data0, labels=labels, autopct='%.1f%%', explode=(1.0, 0, 0, 0, 0),
            colors=["pink", "orange", "yellow", "gray", "green"])

# 파이그래프 출력
plt.show()
```



Histogram

: 빈도수를 알고자 할 때 사용

hist
— 함수를 이용해서 작성

bins
— 옵션에 구간의 범위를 설정
— 범위의 개수만큼 출력

*** CSV 읽을 때 확인**
1. 한글 존재 여부
2. 구분자 확인

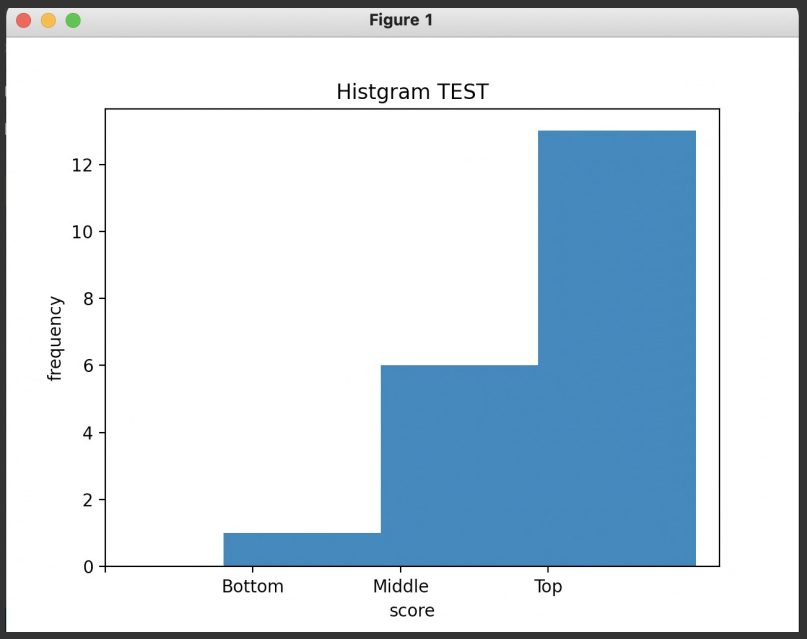
```
['이름', '점수']
['강소희', '76']
['고예림', '80']
['김연경', '100']
['김해란', '85']
['문정원', '82']
['배유나', '87']
['양효진', '50']
['황연주', '50']
['김희진', '50']
['염혜선', '70']
['유서연', '70']
['이다영', '20']
['김수지', '52']
['이소영', '82']
['이재영', '80']
['정대영', '76']
['표승주', '87']
['하혜진', '80']
['박정아', '80']
```

```python
import matplotlib as plt
import csv

# korea.csv 사용

# 이름과 점수를 저장할 list
names = []
scores = []

# korea.csv 파일 열기
f = open('./korea.csv', encoding='ms949')
data = csv.reader(f)

# 이상 없는지 확인
for i in data:
    print(i)
```

```python
import matplotlib as plt
import csv

# korea.csv 사용

# 이름과 점수를 저장할 list
names = []
scores = []

# korea.csv 파일 열기
f = open('./korea.csv', encoding='ms949')
data = csv.reader(f)

# 첫번째 행이 열의 제목 : 넘어가기 next()

next(data)

# 각 줄을 읽어서 첫번째 데이터는 names
# 두번째 데이터는 scores 에 추가
for row in data:
    names.append(row[0])
    scores.append(row[1])

# 기본적으로 읽으면 문자열
print(type(scores[0]))
# reuslt -----> <class 'str'>

# scores 의 모든 데이터를 정수로 변환
for i in range(0, len(scores), 1):
    scores[i] = int(scores[i])

print(type(scores[0]))
# result -----> <class 'int'>
```

```
/Users/mac/PycharmProjects/pythonProject/ve
<class 'str'>
<class 'int'>
```

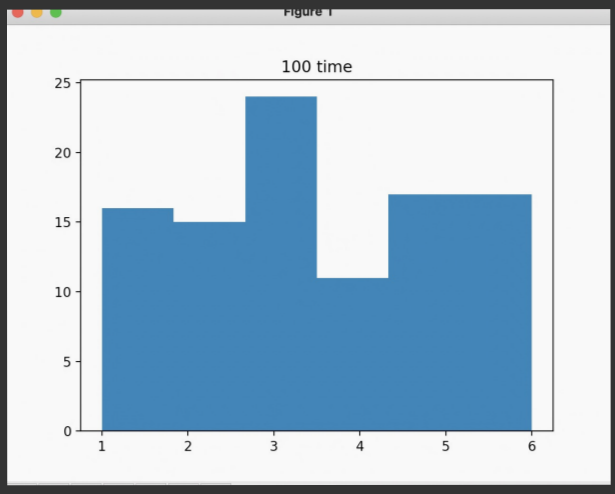
```python
import matplotlib.pyplot as plt
import csv

# korea.csv 사용

# 이름과 점수를 저장할 list
names = []
scores = []

# korea.csv 파일 열기
f = open('./korea.csv', encoding='ms949')
data = csv.reader(f)

# 이상 없는지 확인
# for i in data:
#     print(i)

# 첫번째 행이 열의 제목 : 넘어가기 next()

next(data)

# 각 줄을 읽어서 첫번째 데이터는 names
# 두번째 데이터는 scores 에 추가
for row in data:
    names.append(row[0])
    scores.append(row[1])

# 기본적으로 읽으면 문자열
print(type(scores[0]))
# reuslt -----> <class 'str'>

# scores 의 모든 데이터를 정수로 변환
# 시각화를 해야하기 때문에
for i in range(0, len(scores), 1):
    scores[i] = int(scores[i])

print(type(scores[0]))
# result -----> <class 'int'>

# histogram 그리기
plt.figure()

# 히스토그램을 그릴 때는 (빈도수 파악) 범주형 데이터가 아닌 경우
# 구간화(binding)을 하는 것이 좋다
plt.hist(scores, bins=3)


plt.xticks(range(0, 100, 25), labels=['', 'Bottom', 'Middle', 'Top'])

# Option
plt.title("Histgram TEST")
plt.xlabel("score")
plt.ylabel("frequency")

plt.show()
```



——— simulation 횟수의 차이

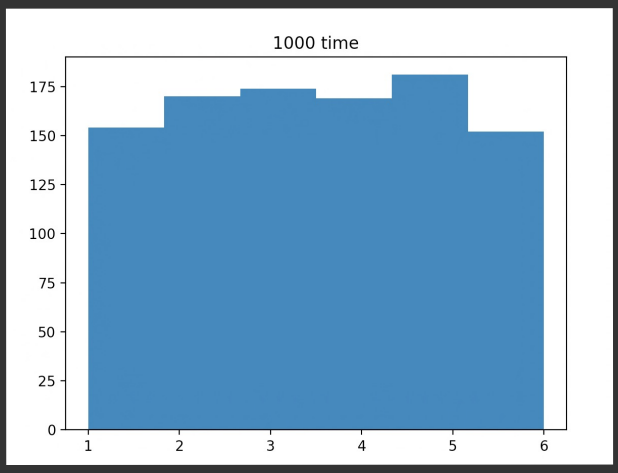
———〉 100

```
import random
import matplotlib.pyplot as plt


dice = []
for i in range(0, 100, 1):
    dice.append(random.randint(1, 6))

plt.figure()
plt.hist(dice, bins=6)
plt.show()
```



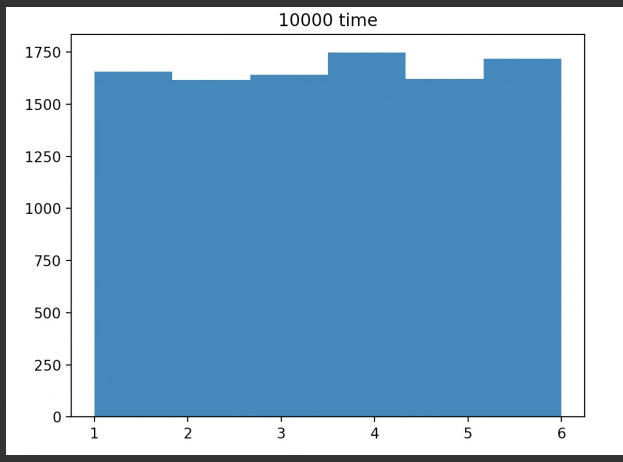
———〉 1000

```
import random
import matplotlib.pyplot as plt


dice = []
for i in range(0, 1000, 1):
    dice.append(random.randint(1, 6))

plt.figure()
plt.hist(dice, bins=6)
plt.show()
```



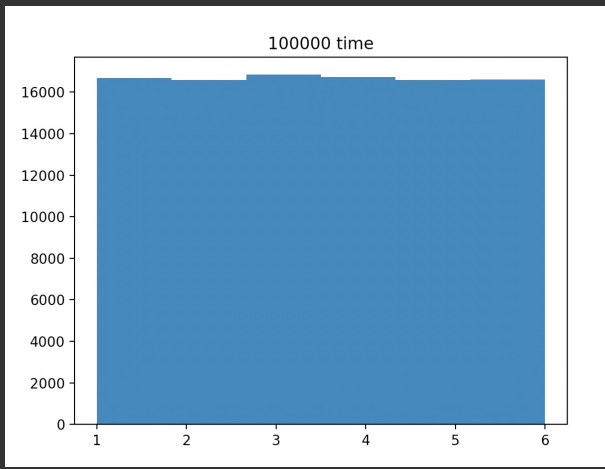
———〉 10000

```
import random
import matplotlib.pyplot as plt


dice = []
for i in range(0, 10000, 1):
    dice.append(random.randint(1, 6))

plt.figure()
plt.hist(dice, bins=6)
plt.show()
```



———〉 100000

```
import random
import matplotlib.pyplot as plt


dice = []
for i in range(0, 100000, 1):
    dice.append(random.randint(1, 6))

plt.figure()
plt.hist(dice, bins=6)
plt.show()
```



BoxPlot

: 상자 그림
- 데이터의 분포를 확인할 때 많이 사용하는 그래프
- boxplot이라는 함수로 출력

상자의 경계선
: 1/4 이 되는 수 (1 사분위 수 와 3사분위 수)

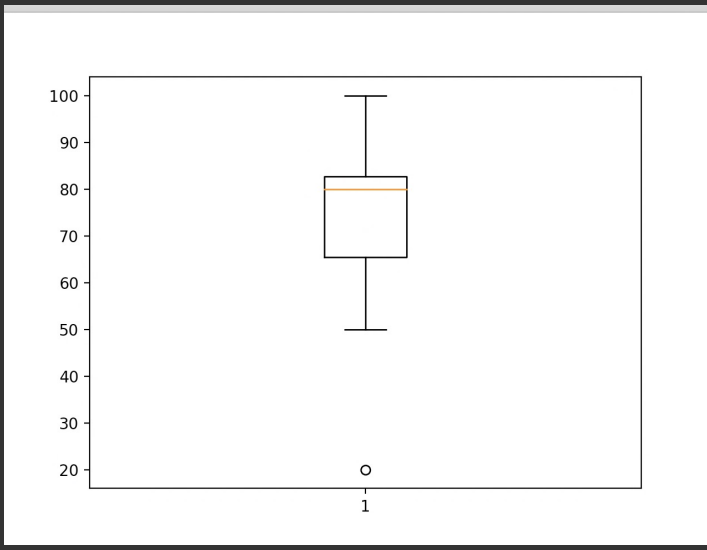
상자의 가운데 선
: 중앙 값

상자의 좌우 수염
: 1사분위수 - 1.5*(3사분위수 - 1사분위수) + 3사분위 수 + 1.5* (3사분위수 - 1사분위수)

수염을 벗어난 동그라미는 outlier(이상치)로 간주

상자안에 데이터가 50% 가 배치
수염안에 99 % 정도 배치

```
import matplotlib.pyplot as plt
import csv

# korea.csv 사용

# 이름과 점수를 저장할 list
names = []
scores = []

# korea.csv 파일 열기
f = open('./korea.csv', encoding='ms949')
data = csv.reader(f)


next(data)

for row in data:
    names.append(row[0])
    scores.append(row[1])

for i in range(0, len(scores), 1):
    scores[i] = int(scores[i])

print(type(scores[0]))

plt.figure()
plt.boxplot(scores)
plt.show()
```



그래프 여러개를 개별로 그리고자 하는 경우

: figure() 의 리턴 값을 받아서
add_subplot()을 호출해서 SubPlot 을 생성

parameter —〉 행의 개수, 열의 개수, Index 설정

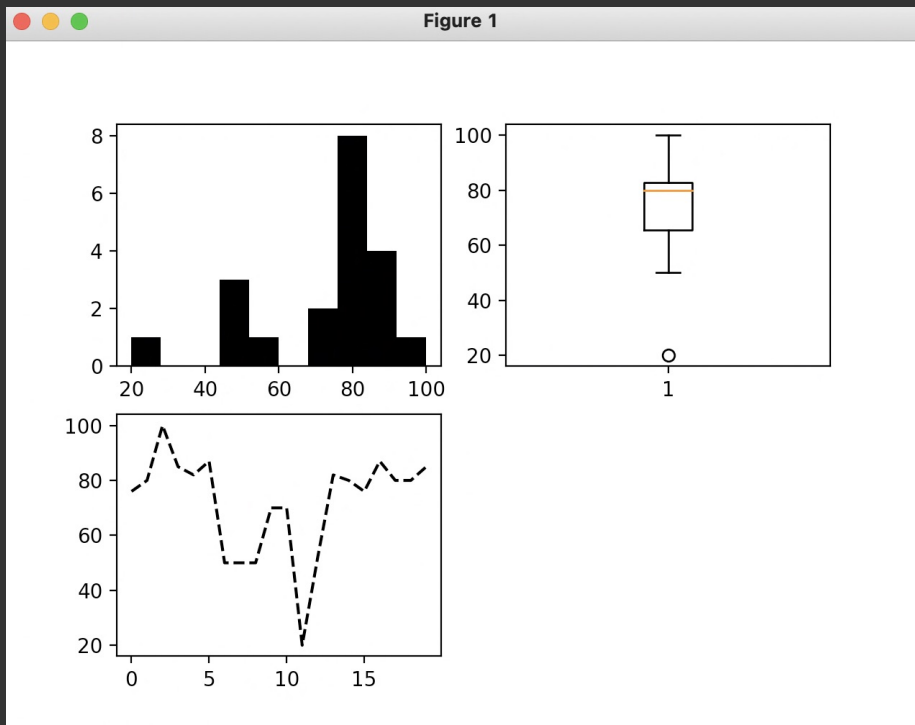
——〉 subplot 을 생성

* subplot을 이용해서 그래프를 출력하면
하나의 영역에 여러개의 그래프를 그리는 것이 가능

* 보고서를 만들거나, application 출력시 사용

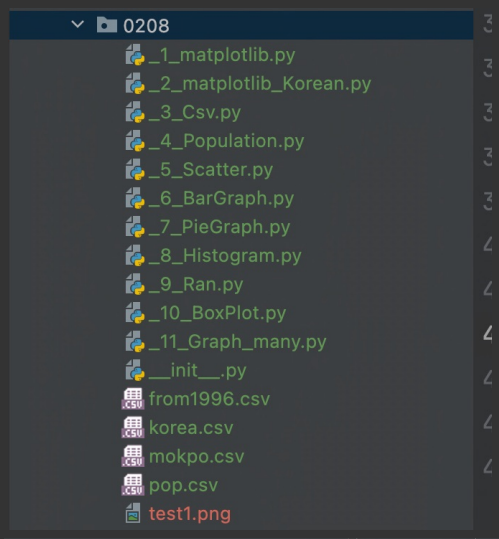
(탐색에는 사용을 잘하지 않음)

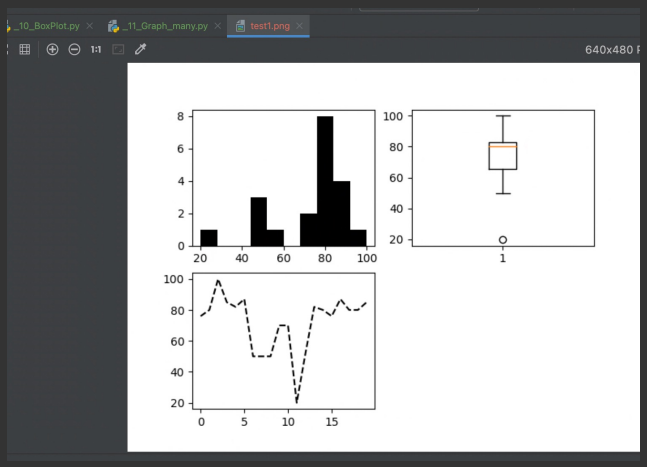
```
import matplotlib.pyplot as plt
import csv

# korea.csv 사용

# 이름과 점수를 저장할 list
names = []
scores = []

# korea.csv 파일 열기
f = open('./korea.csv', encoding='ms949')
data = csv.reader(f)


next(data)

for row in data:
    names.append(row[0])
    scores.append(row[1])

scores = list(map(int, scores))

# 영역을 생성
fig = plt.figure()

# Sub 영역을 생성
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)

# 현재 활성화된 영역에 산점도를 출력
plt.plot(scores, 'k--')

# ax1 에 히스토그램 출력
ax1.hist(scores, color='k')

# ax2 에 boxplot 출력
ax2.boxplot(scores)


plt.show()
```



〈 그래프 결과 저장 〉

```python
import matplotlib.pyplot as plt
import csv

# korea.csv 사용

# 이름과 점수를 저장할 list
names = []
scores = []

# korea.csv 파일 열기
f = open('./korea.csv', encoding='ms949')
data = csv.reader(f)


next(data)

for row in data:
    names.append(row[0])
    scores.append(row[1])

scores = list(map(int, scores))

# 영역을 생성
fig = plt.figure()

# Sub 영역을 생성
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)

# 현재 활성화된 영역에 산점도를 출력
plt.plot(scores, 'k--')

# ax1 에 히스토그램 출력
ax1.hist(scores, color='k')

# ax2 에 boxplot 출력
ax2.boxplot(scores)
```






```python
# 현재 영역에 그려진 그래프를 그림 파일로 저장
fig.savefig('./test1.png')

plt.show()
```

** Python GUI Programming

tkinter

PyQt5
: 전통적으로 많이 해옴
  python이나 anaconda 의 기본 package r가 아님
    —〉 pycharm 등을 사용하면 Error 를 발생하는 경우도 있음

wxPython

데이터 분석에 많이 사용하는 Package

〈 numpy 〉
: 수치 해석이나 , 선형 대수 계산 기능을 제공하는 package

특징
1. C언어 버전과 fortran 버전이 있음
2. 배열을 제공하고 벡터화된 연산을 지원

〈 pandas 〉
: 테이블 형태의 데이터를 다루는 DataFrame 과 Series 자료 구조를 제공하는 package

특징
1. 데이터 탐색 , 정리에 매우 유용한 Package
2. 데이터 전처리 과정이나, 기술 통계 분석에 많이 사용

〈 matplotlib 〉
: 시각화를 위한 package

〈 seaboran 〉
: matplotlib에 비해 추가된 시각화 library(sample 데이터도 제공)

〈 pgmpy 〉
: 확률론적 그래프 모형을 제공하는 package

〈 StatusModels 〉
: 통계 및 회기 분석이나 시계열 분석용 package

〈 scipy 〉
: 고급 수학 함수 , 미적분 , 최적화 , 신호 처리 등의
  다양한 과학 기술 계산 기능을 제공한 package

〈 sympy 〉
: 인수분해 , 미분, 적분 등의 연산 기능을 제공하는 package

〈 scikit-learn 〉
: 머신러닝 학습용 package

〈 tensorflow 〉
: 구글이 만든 AI 용 package

특징
안드로이드 용, 자바스크립트 버전 등을 같이제공
—〉 현업에서 많이 사용

〈 keras 〉
: tesorflow 가 저수준이라서 이를 고수준으로 rapping 한 package

〈 pytoch 〉
: face 제공하는 AI용 package
자유도가 높아서 연구 개발에 많이 사용하는 Deep Learning package

NUMPY

: 고성능의 과학적 계산(선형 대수)를 수행하기 위한 핵심 package

특징

1. python machine Learning stack 의 기초
   Deep Learning 은 기본적으로 numpy 의 array 만 사용 가능

2. anaconda를 설치하면 같이 설치가 된다

import

: import numpy as np
—〉 numpy package 의 모든 것들을 np라는 이름으로 가져와서 사용

ndarray

: numpy 에서 제공하는 배열
—〉 python 의 list 보다 생성방법 다양
—〉 연산의 종류도 많고 , 벡터화된 연산을 지원하므로 연산속도도 빠름

——〉 list와 ndarray의 작업시간

```
import numpy as np
import datetime

# list 와 ndarray 작업 속도 비교

# list 생성 - 1 부터 100000000 까지
li = list(range(1, 10000001, 1))

# 현재 시간
listStartTime = datetime.datetime.now()
print(f"List Operating start time : {listStartTime}")

# 모든 데이터에 10 을 곱해서 저장
for i in li:
    i * 10;

listEndTime = datetime.datetime.now()
print(f"List Operating End time : {listEndTime}")



listResult = listEndTime-listStartTime;
print(f"Operating Time of List is : {listResult}")


# ndarray 시간 ----------------------------
print('----------------------------------------')



ar = np.arange(1, 10000001, 1)
arrayStartTime = datetime.datetime.now()
print(f"Array Operating start time : {arrayStartTime}")

# 모든 데이터에 10 을 곱해서 저장
ar * 10

arrayEndTime = datetime.datetime.now()
print(f"Array Operating End time : {arrayEndTime}")



arrayResult = arrayEndTime-arrayStartTime;
print(f"Operating Time of array is : {arrayResult}")


if arrayResult<listResult:
    print("Array 가 더 빠르다")
else:
    print("list 가 더 빠르다")

# ndrayy
```

```
List Operating start time : 2022-02-08 16:49:09.980271
List Operating End time : 2022-02-08 16:49:10.523201
Operating Time of List is : 0:00:00.542930
----------------------------------------
Array Operating start time : 2022-02-08 16:49:10.546074
Array Operating End time : 2022-02-08 16:49:10.573971
Operating Time of array is : 0:00:00.027897
Array 가 더 빠르다
```

〈 ndrray 의 생성 〉
—〉 numpy. array(iterator 객체. copy=True)

copy 는 복제 여부 설정하는 것으로 기본값을 복제해주는 True

특징
1. ndarray 나 copy. asarray 등의 함수로도 생성이 가능 —〉 잘 사용하지 않음

2. 생성할 때 dtype paramter를 이용해서 type 설정 가능
—〉 설정하지 않으면 numpy 가 유추 해서 생성

〈 ndarray 정보 확인 〉

특징
1. print
: 함수를 이용해서 데이터 전체를 확인하는 것이 가능

2. dtype 속성을 이용
: 요소 1개의 자료형을 확인하는 것이 가능

3. ndim 속성을 이용
: 배열의 차원

4. shape 속성
: 각 차원의 크기를 저장하고 있는 튜블

5. size 속성
: 전체 데이터의 개수를 확인

6. itemsize
: 하나의 항목이 차지하는 메모리의 크기

7. nbytes
: 전체가 차지하는 memory의 크기

———— 생성하고 정보 확인
외부에서 제공되는 데이터의 경우
반드시 type으로 전체 데이터의 자료형을 확인
dtype으로 하나의 요소의 자료형을 확인
shape 로 차원을 확인

```python
# 실제 생성은 학습할 때만

import numpy as np

# 일차원 배열 생성
ar = np.array([1, 2, 3])

# ar 의 자료형을 확인
print(type(ar))

# ar의 요소의 자료형 확인
print(ar.dtype)
```

```
<class 'numpy.ndarray'>
int64
```

```python
# 1차원을 출력
print(ar.ndim)

# 1차원인데 데이터는 3개
print(ar.shape)
```

```
1
(3,)
```

```python
# 2차원배열 생성하기 -----> 이미지가 대부분 이런 형식
ar = np.array([[1, 2, 3], [4, 5, 6]])
print(ar.ndim)
print(ar.shape)
```

2 차원

2 차원인데 데이터는 3개

```
2
(2, 3)
```

Ndarray 생성

1. 일련 번호 형태의 생성

* 대괄호가 있는 것은 생략 가능 —> start , step 생략가능
* dtype = None 이 있음으로 마찬가지로 생략 가능

numpy. arrange([start,] stop [, step ,] dtype=None)

2. 시작점 과 끝점을 기준으로 균일한 간격으로 개수를 설정하여 생성

* Sample Data 만들 때 자주 사용

numpy.linespace(
    start,
    stop,
    num = 50 ,
    endPoint = True ,
    retstep= False ,
    dtype = None
    axis=0
)

```
[0 1 2 3 4 5 6 7 8 9]
--------------horizon-----------------

[0.  0.2 0.4 0.6 0.8 1. ]
--------------horizon-----------------

[0.         0.16666667 0.33333333 0.5        0.66666667 0.83333333]
--------------horizon-----------------
```

* num : 기본적으로 50
* endPoint : 기본적으로 True , 끝자리 추가할 것 인지

```python
import numpy as np

ar = np.arange(10)
print(ar)
# -------> [0 1 2 3 4 5 6 7 8 9]
print("---------------horizon------------------")
print()

# 마지막 데이터를 포함
ar = np.linspace(0, 1, 6)
print(ar)
print("---------------horizon------------------")
print()
# -------> [0.  0.2 0.4 0.6 0.8 1. ]


# 마지막 데이터를 미포함
ar = np.linspace(0, 1, 6, endpoint=False)
print(ar)
print("---------------horizon------------------")
print()
# -------> [0.         0.16666667 0.33333333 0.5        0.66666667 0.83333333]
```

특수 행렬 생성

zeros() , ones()
: 0 과 1 로 채워진 배열을 생성하는데 차수를 정수나 튜플로 설정

zero_like() 와 one_like()
: 다른 배열을 parameter 로 받아서 동일한 크기의 배열을 생성
0 이나 1 로 채움

empty() , empty_like
: 초기화 하지 않은 값으로 배열을 채움
대각선 방향으로 1을 채운 행렬을 생성

N 에 행의 수

M에 열의 수

k 에 대각선의 위치를 dtype에 요소의 자료형을 설정

N만 설정하면 정방 행렬
(행 과 열의 수가 같은 단위 행렬 생성)

diag()
: 행렬에서 대각선 방향의 값들만 추출해서 vector를 생성하는 함수

```
==================================================

[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

==================================================

[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]

==================================================
```

```python
import numpy as np
print("==================================================")
print()
# 1로 채워진 10 개의 데이터를 가진 배열 생성
ar1 = np.ones(10)
print(ar1)
print()
print("==================================================")


print()
# 1로 채워진  5*5 배열을 생성
ar2 = np.ones((5, 5))
print(ar2)
print()
print("==================================================")
```

대각 행렬

```
# 2 * 2 정방향 행렬을 만들고 대각선 방향으로 1을 채운 행렬 생성
print("---------------------------------- 2*2 대각 행렬 생성 ")
ar3 = np.eye(2)
print(ar3)
print()
print("================================================")


# 4 * 4 정방향 행렬을 만들고 대각선 방향으로 1을 채운 행렬 생성
print("---------------------------------- 4*4 대각 행렬 생성 ")
ar3_1 = np.eye(4)
print(ar3_1)
print()
print("================================================")
```

```
================================================
---------------------------------- 2*2 대각 행렬 생성
[[1. 0.]
 [0. 1.]]

================================================
---------------------------------- 4*4 대각 행렬 생성
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

================================================
```

```python
# 대각 행렬의 1 의 위치를 수정하기 -- 1
print("---------------------------------- 4*4 대각 행렬 생성 ")
ar3_1 = np.eye(4, k=1)
print(ar3_1)
print()
print("================================================")

# 대각 행렬의 1 의 위치를 수정하기 -- 2
print("---------------------------------- 4*4 대각 행렬 생성 ")
ar3_1 = np.eye(4, k=-1)
print(ar3_1)
print()
print("================================================")
```

```
================================================
---------------------------------- 4*4 대각 행렬 생성
[[0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 0.]]

================================================
---------------------------------- 4*4 대각 행렬 생성
[[0. 0. 0. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]

================================================
```

```python
# 대각선에서 데이터를 추출해서 새로운 1차 배열 생성
print("대각선에서 데이터를 추출해서 새로운 1차 배열 생성")
br = np.diag(ar3_1, k=-1)
print(br)
print()
print("==================================================")
```

```
==================================================
대각선에서 데이터를 추출해서 새로운 1차 배열 생성
[1. 1. 1.]

==================================================
```

ndarray 의 자료형

: 모든 요소의 자료형이 일치해야한다

특징

1 . 딥러닝에서 기본 자료구조
    머신러닝에서는 DataFrame을 사용하기도 한다

2. 생성할 때 dtype 속성을 설정하지 않으면
    numpy 가 유추
      문자열 —〉 실수 —〉 정수

3 . type 을 변경하고자 하면
    astype 함수를 이용
      기존 배열을 가지고 새로운 배열을 생성

numpy 의 자료형

정수
: int 8 , int16 , int32, int64

양수
: uint 8 , uint 16 , uint 32, uint 64

실수
: float16, float32, float64, float128
—〉 정밀한 계산 가능

복소수
: complex64, complex128, complex256

boolean
: bool

객체
: object

문자열
: string

unicode
: unicode_

Type 변경을 할 수 있어야 한다

: 이미지를 가지고 Deep Learning 을 하는 경우가 많음

이때 , 이미지들의 모든 자료형이 일치해야

이미지를 가지고 딥러닝을 할 수 있음

```
import numpy as np
print()
# 일반적으로 생성하면 int 64, int 32 이다
ar = np.array([1, 2, 3])
print(ar.dtype)
# -----------> int64
print("-------------------------")

# U로 바뀜
ar = np.array([1, 2, '3'])
print(ar.dtype)
# -----------> <U21
print("-------------------------")

ar = np.array([1, 2, '3'], dtype=np.int32)
print(ar.dtype)
# -----------> int32
print("-------------------------")
```

```
int64
-------------------------
<U21
-------------------------
int32
-------------------------
```

```
# Type 을 float 32로 변경
ar = ar.astype(np.float32)
print(ar.dtype)
# -----------> int32
print("-------------------------")
```

```
-------------------------
float32
-------------------------
```

행렬의 구조 변환 —〉 차원을 변경

머신러닝이나 딥러닝에서는
특정 모델에 따라 입력되는 데이터의 차원이 결정되어 있다 .

사용하고자 하는 데이터가 모델의 입력으로 맞지 않으면
——〉 차원을 변경해서 맞춰주어야 한다

최근 딥러닝
모델을 직접 생성해서 작업을 수행 하는 것 보다
기존의 모델을 활용한 **전이학습**을 많이한다 (전이학습)

reshape 함수 이용

: 원하는 차원의 개수를 튜플로 대입하면
다른 차원 또는 데이터 개수를 조절해서 구조를 변경하는 것이 가능

reshape 함수에
-1 을 대입하면 1차원 배열로 변환

flattern 함수에 다차원 배열 대입
1 차원 배열로 변환

reshape 와 flattern 의 차이

1. reshape 는 기존 데이터를 변경.
2. flattern 은 새로운 배열을 생성해서 리턴

```python
import numpy as np

ar = np.arange(10)
print(ar)
# ---> [0 1 2 3 4 5 6 7 8 9]
print()
print("--------------------------")

print(" Reshape :ar 을 2 차원 배열로 변환 ====================== ")
br = ar.reshape((2, 5))
print(br)
'''
[[0 1 2 3 4]
 [5 6 7 8 9]]
'''
print()
print("ReShape 다시 복원---------------------- ")
print(br.reshape(-1))
print()

print("--------------================ 1차원 배열을 3차원으로 변환 ")
# 1 차원 배열을 생성
ar = np.arange(20)
# 1 차원 배열을 3차원으로 변환
print(ar.reshape(2, 5, 2))

print("--------------================ 2대신 -1   ")
# 마지막 차원의 개수가 -1 일때 알아서 분할하여 생성한다
print(ar.reshape(2, 5, -1))
```

```
[0 1 2 3 4 5 6 7 8 9]

--------------------------
 Reshape :ar 을 2 차원 배열로 변환 ======================
[[0 1 2 3 4]
 [5 6 7 8 9]]

ReShape 다시 복원----------------------
[0 1 2 3 4 5 6 7 8 9]

--------------================ 1차원 배열을 3차원으로 변환
[[[ 0  1]
  [ 2  3]
  [ 4  5]
  [ 6  7]
  [ 8  9]]

 [[10 11]
  [12 13]
  [14 15]
  [16 17]
  [18 19]]]
--------------================ 2대신 -1
[[[ 0  1]
  [ 2  3]
  [ 4  5]
  [ 6  7]
  [ 8  9]]

 [[10 11]
  [12 13]
  [14 15]
  [16 17]
  [18 19]]]
```