10. python 데이터 분석 라이브러리

contents

1. 데이터분석 라이브러리개요

- **python의 주요 데이터분석 라이브러리**
  - 넘파이(NumPy)
    - Python 데이터분석의 기본적인 기능들을 제공
    - 벡터 및 행렬 연산과 관련된 편리한 기능들 제공
  - 판다스(Pandas)
    - Series, DataFrame 등의 자료 구조를 활용하여 데이터 분석에 우수한 성능 발휘
    - 대량의 데이터를 빠른 속도로 처리 가능
  - 맷플롯립(Matplotlib)
    - 데이터 분석 결과에 대한 시각화를 빠르고 직관적으로 수행

1. 데이터분석 라이브러리의 개요

- **라이브러리 설치**

```
pip install numpy
pip install pandas
pip install matplotlib
```

anaconda 배포판 이미 설치되어 있음

- **라이브러리 import**

```
import numpy as np
import pandas as pd
import matplotlib
```

1. 데이터분석 라이브러리개요

- **numpy**

```
#50개의 난수 생성
data=np.random.rand(50)
print(type(data))
print(data)

# list를 사용하여 numpy 배열 만들기
a=[1,2,3,4,5]
data=np.array(a)
```

1. 데이터분석 라이브러리의 개요

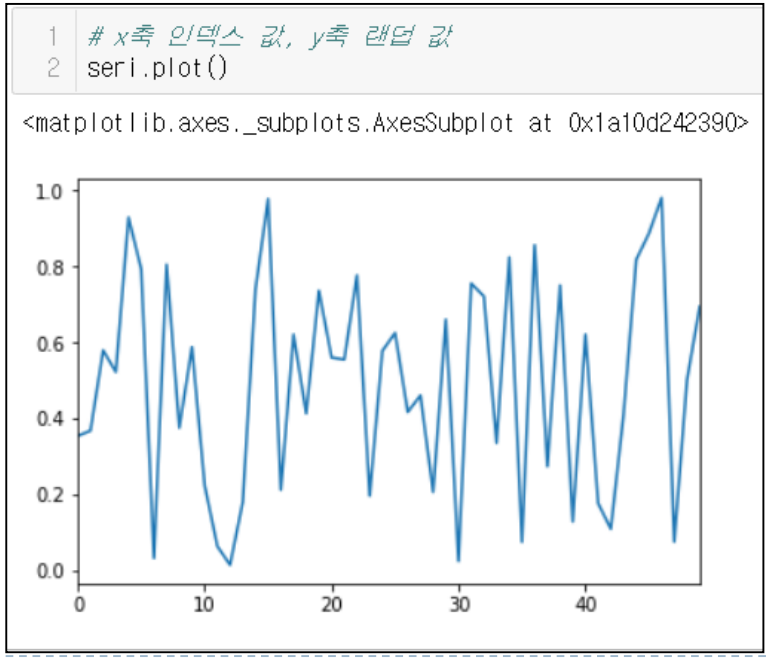
- **pandas**

```
# 넘파이 배열을 판다스의 시리즈 자료형으로 변환(인덱스,데이터의 조합)
seri=pd.Series(data)
print(type(seri))
print(seri)
```

- **pandas Series 데이터로 차트 작성**

```
# x축 인덱스 값, y축 랜덤 값
seri.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10d242390>
```



1. 데이터분석 라이브러리의 개요

- **pandas 데이터프레임**

```
#rand(rows, cols)
#50개의 난수 생성
data_set=np.random.rand(10,3)
print(data_set)
print(type(data_set)) # 변수의 자료형 확인
print(data_set.shape) #행렬의 차원 확인
```

```
[[0.34485559 0.28497075 0.78006357]
 [0.50854493 0.67141959 0.30071159]
 [0.41970225 0.4573258  0.96723893]
 [0.1959138  0.42263276 0.49936199]
 [0.71791036 0.59629664 0.03647939]
 [0.38704617 0.08416631 0.48284201]
 [0.66055622 0.01640587 0.37022955]
 [0.73999638 0.16915373 0.41333187]
 [0.41217824 0.54741988 0.75350144]
 [0.25845176 0.9562979  0.08382933]]
<class 'numpy.ndarray'>
(10, 3)
```

```
#넘파이 행렬을 판다스의 데이터 프레임으로 변환
#데이터프레임: 행과 열로 데이터를 조회할 수 있음
df=pd.DataFrame(data_set, columns=["A","B","C"])
print(df)
print(type(df))
```

```
          A         B         C
0  0.344856  0.284971  0.780064
1  0.508545  0.671420  0.300712
2  0.419702  0.457326  0.967239
3  0.195914  0.422633  0.499362
4  0.717910  0.596297  0.036479
5  0.387046  0.084166  0.482842
6  0.660556  0.016406  0.370230
7  0.739996  0.169154  0.413332
8  0.412178  0.547420  0.753501
9  0.258452  0.956298  0.083829
<class 'pandas.core.frame.DataFrame'>
```

```
df=pd.DataFrame(data_set, columns=["A","B","C"])
df
#print(df)
#print(type(df))
```

	A	B	C
0	0.344856	0.284971	0.780064
1	0.508545	0.671420	0.300712
2	0.419702	0.457326	0.967239
3	0.195914	0.422633	0.499362
4	0.717910	0.596297	0.036479
5	0.387046	0.084166	0.482842
6	0.660556	0.016406	0.370230
7	0.739996	0.169154	0.413332
8	0.412178	0.547420	0.753501
9	0.258452	0.956298	0.083829

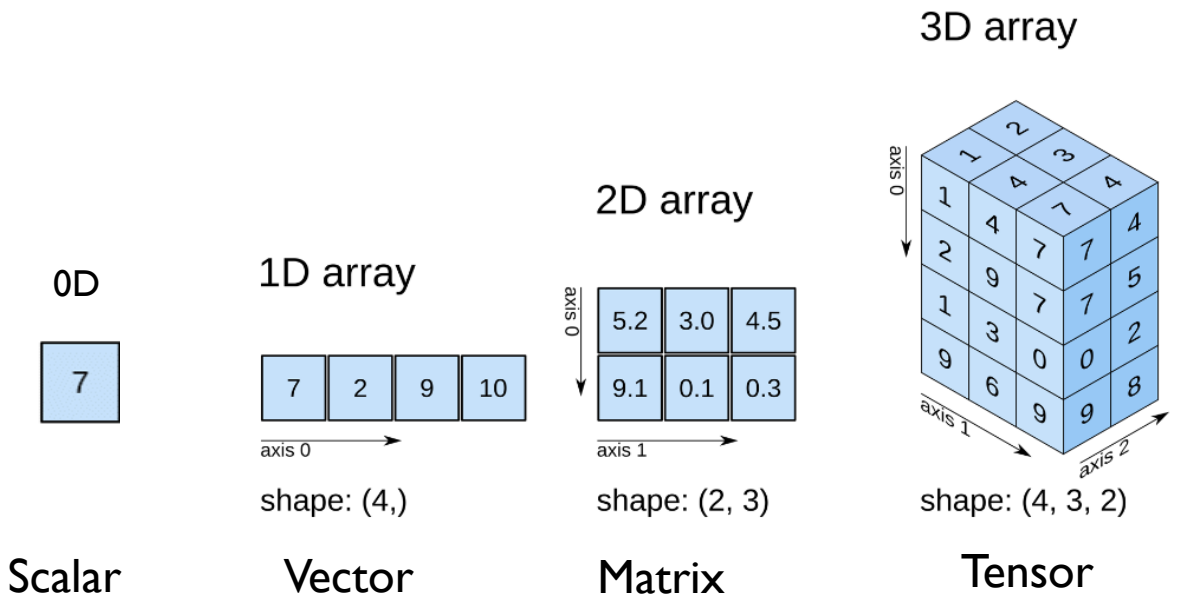
2. numpy

- **numpy 개념**
  - 벡터 및 행렬 연산에 특화된 라이브러리
  - array단위 데이터를 관리함, 행렬(matrix)와 비슷함
  - 동적으로 생성되는 리스트와 달리 numpy 배열은 정적 메모리 할당
  - 딥런닝을 위한 텐서플로우에서 효과적 활용
  - pandas와 함께 데이터 분석에 많이 사용됨

순서	**①** **②** **③**	**numpy 모듈 가져오기** **파이썬 데이터형(예: 리스를) numpy형식으로 변환** **numpy에서 제공하는 기능 수행(예 평균)**
명령문	① ② ③	import numpy as np x= np.array([1, 3, 5]) print(x.mean())

2. numpy

- **넘파이 배열의 차원**
  - 스칼라(Scalar) : 0차원 배열, 배열에서 값을 표현하는 가장 기본 단위, 하나의 실수 값을 담음
  - 벡터(Vector) : 1차원 배열,
  - 행렬(Matrix) : 2차원 배열
  - 텐서(Tensor) : 3차원 이상의 배열



numpy 배열의 차원

2. numpy

- **numpy 배열 생성**

1차원 배열 생성

```
import numpy as np
list1=[1,2,3,4,5]
a=np.array(list1)
print("a:", a)
print("a.shape:", a.shape)
print("a[0]:", a[0])
```

✓ 0.0s

```
a: [1 2 3 4 5]
a.shape: (5,)
a[0]: 1
```

2차원 배열 생성

```
b=np.array([[1,2,3],[4,5,6]])
print("b:", b)
print("b.shape:", b.shape)
print("b[0,0]:", b[0,0])
print("b[0]:", b[0])
```

✓ 0.0s

```
b: [[1 2 3]
 [4 5 6]]
b.shape: (2, 3)
b[0,0]: 1
b[0]: [1 2 3]
```

2. numpy

- **배열 생성 함수**

함수	설명
array()	리스트를 배열로 변환
arange()	특정 범위의 값으로 배열 생성, range() 함수와 유사
ones()	1로 채워진 n차원 배열 생성
zeros()	0로 채워진 n차원 배열 생성
ones_like()	주어진 배열의 크기로 1을 채워서 배열 생성
zeros_like()	주어진 배열의 크기로 0을 채워서 배열 생성
empty()	초기화 되지 않는 빈 n차원 배열 생성
eye() 또는 identity()	대각석 요소만 1을 채우고 그 외의 요소는 0으로 채워진 2차 원배열 생성(단위행열)
linespace	초깃값부터 최종값까지 지정한 간격의 수를 채워 배열 생성
full()	지정한 모양에 지정한 값으로 채워진 배열 생성

2. numpy

- 배열 생성 실습

```
a1=np.arange(1, 10, 1)
print(a1)
a2=np.zeros(5)
a3=np.ones(5)
print(a2, a3)
a4=np.ones_like(a3)
a5=np.zeros_like(a3)
print(a4, a5)
a6=np.eye(3)
print(a6)
a7=np.full((2,3), 5)
print(a7)
a8=np.linspace(0, 10, 5, endpoint=True, retstep=True) //endpoint : 마지막 값 표시 여부, retstep : 간격 표시 여부
print(a8)
a9=np.empty((3,2))   // 초기화 되지 않은 임의 값으로 채워 3행 2열 배열 생성
print(a9)
```

2. numpy

- **numpy 자료형**
  - 정수, 실수, 논리값, 복소수, 문자열 등
- **자료형 변환**

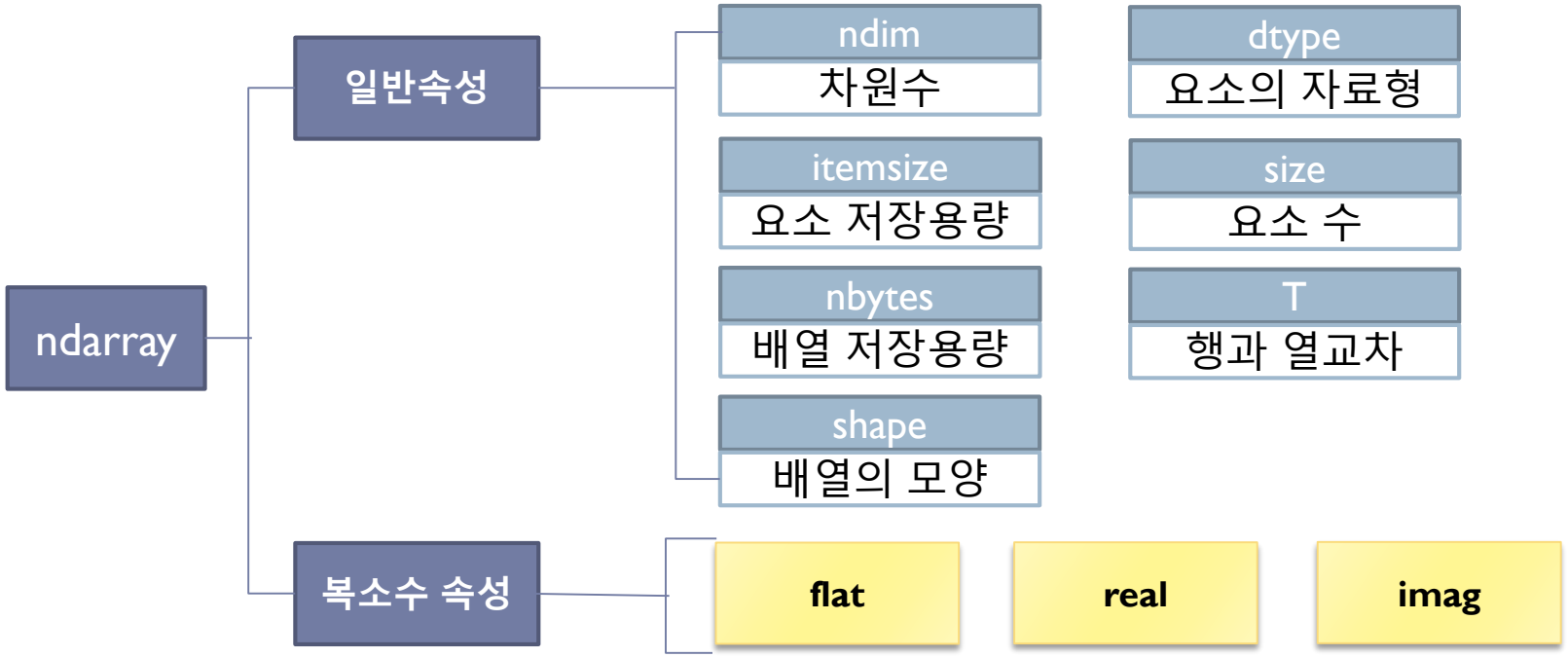
```
ai=np.array([1,2], dtype=np.float64)
print(ai.dtype)
ai_2=ai.astype(np.int8)
print(ai_2.dtype)
```

✓ 0.0s

```
float64
int8
```

2. numpy

- **numpy 배열의 속성**
  - ndarray : 다차원 배열 객체, 속성은 일반 속성과 복소수 속성(범위 벗어남)



2. numpy

- **배열 속성 호출**

```
arr=np.array([[0,1,2],[3,4,5]])
print('arr:\n', arr)
print('type(arr):', type(arr))
print('arr.ndim:', arr.ndim)
print('arr.dtype:', arr.dtype)
print('arr.itmesize:', arr.itemsize)
print('arr.size:', arr.size)
print('arr.nbytes:', arr.nbytes)
print('arr.T:\n', arr.T)
print('arr.shape:', arr.shape)
```

실행결과

```
arr:
 [[0 1 2]
 [3 4 5]]
type(arr): <class 'numpy.ndarray'>
arr.ndim: 2
arr.dtype: int32
arr.itmesize: 4
arr.size: 6
arr.nbytes: 24
arr.T:
 [[0 3]
 [1 4]
 [2 5]]
arr.shape: (2, 3)
```

2. numpy

- **배열의 모양 변경**
  - 배열 모양 변경 함수

함수	설명
flatten()	1차원 배열로 변경
resize()	배열의 모양을 i x j로 변경
reshape()	배열의 모양을 I x j로 변경 resize()와 차이점?
transpose()	열과 행 교차

  - 속성으로 모양 변경

객체.shape=(행 크기, 열 크기)

```
a=np.arange(8)
print('a:\n', a)

# 다차원 배열로 변경하기
a.shape=(2,4)
print('a.shape:\n',a)

#1차원 배열로 변경하기
print("flatten:\n", a.flatten())

#resize 함수로 모양변경
a.resize(4,2)
print("a.resize:\n",a)

# reshape() 함수로 모양변경
print('a.reshape(2,4):\n', a.reshape(2,4))
```

[실행결과]

```
a:
 [0 1 2 3 4 5 6 7]
a.shape:
 [[0 1 2 3]
 [4 5 6 7]]
flatten:
 [0 1 2 3 4 5 6 7]
a.resize:
 [[0 1]
 [2 3]
 [4 5]
 [6 7]]
a.reshape(2,4):
 [[0 1 2 3]
 [4 5 6 7]]
```

2. numpy

- **배열의 행과 열 교차**

```
#1차원 배열 생성
a=np.arange(12)
print("a:\n",a)
# 3행, 4열 2차원 배열로 변환
a.resize(3,4)
print("a:\n",a)
# T 속성으로 행/열 교차
print("a.T:\n",a.T)
# transpose() 함수로 행열 교차
print("a.transpose():\n", a.transpose())
```

실행 결과

```
a:
 [ 0  1  2  3  4  5  6  7  8  9 10 11]
a:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
a.T:
 [[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
a.transpose():
 [[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

2. numpy

- **numpy 배열 슬라이싱**
  - list 슬라이싱

```
list1=[[1,2,3],[4,5,6],[7,8,9]]
print(list1[1][1])
print(list1[1][:1])
print(list1[:][1]) #원하는 결과인가?
print(list1[1, 1]) #배열 슬라이싱 방법
print(list1[:,1]) #배열 슬라이싱 방법
```

실행결과

```
5
[4, 5, 6]
[4]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 5
      3 print(list1[:][1]) #원하는 결과인가?
      4 print(list1[1][:1])
----> 5 print(list1[1, 1])
      6 print(list1[:,1])

TypeError: list indices must be integers or slices, not tuple
```

2. numpy

- **numpy 배열 슬라이싱**

```
np_arr=np.arange(1, 10).reshape(3,3)
print("np_arr:\n", np_arr)
print("np_arr[1,1]:\n", np_arr[1,1])
print("np_arr[:,1]:\n",np_arr[:,1])
print("np_arr[:2,1]:\n", np_arr[:2,1])
print("np_arr[::2]:\n", np_arr[::2, 1])
```

```
np_arr:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
np_arr[1,1]:
 5
np_arr[:,1]:
 [2 5 8]
np_arr[:2,1]:
 [2 5]
np_arr[::2]:
 [2 8]
```

2. numpy

- **마스킹**
  - 조건에 맞는 값만 출력하는 기능
  - 데이터 양이 많을 때 반복문으로 조건을 검사하지 않고 원하는 데이터를 찾을 수 있음

```
mask=np.array([0,1,1,0], dtype=bool)
print("mask:\n",mask)

data=np.random.randn(4,2)
print('data:\n', data)

print("마스킹된 데이터 출력:\n", data[mask])
print("마스킹 역전된 데이터 출력:\n", data[~mask])
```

실행결과

```
mask:
 [False  True  True False]
data:
 [[-0.6687363  -1.40278166]
 [ 0.89832411 -1.46249534]
 [ 0.18657767 -0.31847503]
 [-1.18471329 -1.50295108]]
마스킹된 데이터 출력:
 [[ 0.89832411 -1.46249534]
 [ 0.18657767 -0.31847503]]
마스킹 역전된 데이터 출력:
 [[-0.6687363  -1.40278166]
 [-1.18471329 -1.50295108]]
```

2. numpy

- **조건식으로 마스킹 하기**

```
posit=data[data>0]
print("양의 데이터 출력: \n", posit)

# 다중 조건
over1=data[1][data[1]>0]
print("두 번째 행의 양수 데이터 출력:\n", over1)
```

실행결과

```
양의 데이터 출력:
 [0.89832411 0.18657767]
두 번째 행의 양수 데이터 출력:
 [0.89832411]
```

2. numpy

- **numpy 배열 연산**

```
x=np.random.randn(5)
y=np.random.randn(5)
print('x:',x)
print('y:',y)
```

✓ 0.0s

```
x: [ 0.32404751 -2.09838139  0.5792544   0.82862635 -0.30951144]
y: [-1.88740972 -0.051279    0.07806282 -1.41524482 -0.90990586]
```

```
print('np.minimum(x,y):',np.minimum(x,y))
print('np.maximum(x,y):',np.maximum(x,y))
```

✓ 0.0s

```
np.minimum(x,y): [-1.88740972 -2.09838139  0.07806282 -1.41524482 -0.90990586]
np.maximum(x,y): [ 0.32404751 -0.051279    0.5792544   0.82862635 -0.30951144]
```

2. numpy

- **numpy 배열 연산**

사칙 연산

```
print("x+10:", x+10)
print("x+y:", x+y)
print("x-y:", x-y)
print("x*y:", x*y)
print("x/y:", x/y)
print("x//y:", x//y)
print("x%y", x%y)
```

연산 함수

```
print('np.abs(x):', np.abs(x))
print('np.fabs(x):', np.fabs(x))
print('np.sqrt(x):', np.sqrt(x))
print('np.squard(x):', np.square(x))
print('np.exp(x):', np.exp(x))
print('np.log(x):', np.log(x))
print("np.add(x, y):", np.add(x, y))
print("np.substract(x, y):", np.subtract(x, y))
print("np.multiply(x, y):", np.multiply(x, y))
print("np.divide(x, y):", np.divide(x, y))
print("np.floor_divide(x, y):", np.floor_divide(x, y))
print("np.mod(x, y):", np.mod(x, y))
```

2. numpy

- **배열의 연산**

```
a=np.arange(1, 17).reshape(4,4)
print("a:\n", a)
print("np.sum(a):",np.sum(a))  #= a.sum()
print("np.mean(a):", a.mean()) #= np.mean(a)
print("np.sum(a, axis=0):", np.sum(a, axis=0)) # 열기준 계산
print("np.sum(a, axis=1):", np.sum(a, axis=1)) # 행기준 계산
```

✓ 0.0s

```
a:
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
np.sum(a): 136
np.mean(a): 8.5
np.sum(a, axis=0): [28 32 36 40]
np.sum(a, axis=1): [10 26 42 58]
```

2. numpy

- **numpy 배열 연산**

```
1  names=np.array(['김철수','이승호','박철로','김철수','홍성민','마이클','존'])
2  ages=np.array([20,25,20,23,28,29,30,31,30,40,45])
3  print(np.unique(names))
4  print(len(np.unique(names)))
5  print(np.unique(ages))
```

```
['김철수' '마이클' '박철로' '이승호' '존' '홍성민']
6
[20 23 25 28 29 30 31 40 45]
```

2. numpy

- **numpy 배열 연산**
  - 불리언(Boolean)

```
b=[[5,3,4],[2, 5,1],[7,6,9]]
arr_b=np.array(b)

b>5      #리스트일 때 결과
arr_b>b  #numpy배열일 때 결과
np.all(arr_b > 5) #arr_b의 모든 요소가 5보다 클때 True
np.any(arr_b>5) #arr_b의 어떤 요소가 5보다 클때 True
```

  - 등간격 나누기

```
np.linspace(0,15,4) # 시작값, 종료값, 생성갯수
np.linspace(0,15,4,retstep=True)
np.linspace(0,15,4,endpoint=False)
np.logspace(1,2,4) # log10 10
```

2. numpy

- **배열 복사**
  - 얕은 복사: (=) 연산자 사용, 배열 자체가 새로운 장소에 복사되지 않고 원본 데이터 주소가 복사됨
  - 깊은 복사: copy() 함수 사용, 배열 복사본을 생성, 원본을 수정해도 복사본은 바뀌지 않음

얕은 복사 예

```
a=np.arange(6)
b=a
print(a)
print(b is a)
b[0]=10
print(a)
```

✓ 0.0s

```
[0 1 2 3 4 5]
True
[10  1  2  3  4  5]
```

깊은 복사 예

```
c=a.copy()

c[0]=20
print('a:', a)
print('c:', c)
```

✓ 0.0s

```
a: [10  1  2  3  4  5]
c: [20  1  2  3  4  5]
```

2. numpy

- **배열의 정렬**
  - 함수 설명 참고

```
a=np.array([[3,2,4],[2,7,6],[8, 9, 1]])
print("a:\n", a)
print('np.sort(a):\n', np.sort(a))
print('np.sort(a, axis=1):\n', np.sort(a, axis=1)) # 행기준 정렬
print('np.sort(a, axis=0):\n', np.sort(a, axis=0)) # 열기준 정렬
print('np.sort(a, axis=0)[::-1]:\n', np.sort(a, axis=0)[::-1]) # 열기준 역순정렬
print('np.sort(a, axis=1)[:,::-1]:\n', np.sort(a, axis=1)[:,::-1]) #행기준 역순정렬
```

2. numpy

- **시중에 판매되는 초콜릿 중에 우수 상품의 특징을 파악**
  - 초콜릿에 1점~5점 사이 평점 매겨진 데이터 분석
  - 파일명 : chocolate_rating.csv

    첫 번째 열 : 초콜릿 번호
    두 번째 열 : 평점 기록 연도
    세 번째 열 : 카카오 함유량
    네 번째 열 : 평점

```
data=np.loadtxt('chocolate_rating.csv', delimiter=',')
print("차원:", data.ndim)
print("모양:", data.shape)
print("원소의 수:", data.size)
print(data)
```

✓ 0.0s

```
차원: 2
모양: (1795, 4)
원소의 수: 7180
[[1.000e+00 2.016e+03 6.300e-01 3.750e+00]
 [2.000e+00 2.015e+03 7.000e-01 2.750e+00]
 [3.000e+00 2.015e+03 7.000e-01 3.000e+00]
 ...
 [1.793e+03 2.011e+03 6.500e-01 3.500e+00]
 [1.794e+03 2.011e+03 6.200e-01 3.250e+00]
 [1.795e+03 2.010e+03 6.500e-01 3.000e+00]]
```

2. numpy

- 시중에 판매되는 초콜릿 중에 우수 상품의 특징을 파악

```
#모든 초코릿의 평균 평점
ratings_mean=data[:,3].mean()
print("ratings_mean:", ratings_mean)
```

✓ 0.0s

```
ratings_mean: 3.185933147632312
```

```
# 평점이 4점 이상인 우수 초콜릿 골라 내기
high_level=data[data[:, 3]>=4]
# 우수 초콜릿의 id의 데이터 형을 정수로 변환
high_id=high_level[:,0].astype(np.int64)
print('우수 초콜릿의 수:', len(high_id), high_id.size)
print("high_id:\n", high_id)
```

2. numpy

- **시중에 판매되는 초콜릿 중에 우수 상품의 특징을 파악**

```
# 우수 초콜릿 카카오 함유량 분석
high_kakao=high_level[:,2]
# unique() : 배열에서 중복값 제거
unique_values, value_counts=np.unique(high_kakao, return_counts=True)
print("카카오 함유량: ", unique_values)
print('함유량별 빈:', value_counts)

for i in range(len(unique_values)):
    print(unique_values[i],':', value_counts[i])
```

✓ 0.0s

```
카카오 함유량:  [0.6  0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.7  0.71 0.72 0.73 0.74 0.75
 0.78 0.8  0.88]
함유량별 빈: [ 1  3  4  2  1  3  2  2 45  1 11  1  4 17  1  1  1]
0.6 : 1
0.63 : 3
0.64 : 4
0.65 : 2
```

2. numpy

- **시중에 판매되는 초콜릿 중에 우수 상품의 특징을 파악**

```
# 우수 초콜릿 중 가장 빈도 수가 큰 카카오 함유량 구하기
max_index=np.argmax(value_counts) # 가장 큰 빈도수를 가진 위치 구하기
print("max_index:", max_index)
print("빈도수가 가장 높은 함유량:",unique_values[max_index])
print("우수 초콜릿 {} 가지 중 {}가지의 카카오 함유량이 {}입니다"
      .format(high_id.size, value_counts[max_index], unique_values[max_index]*100) )
```

https://grouplens.org/datasets/movielens

✓ 0.0s

```
max_index: 8
빈도수가 가장 높은 함유량: 0.7
우수 초콜릿 100 가지 중 45가지의 카카오 함유량이 70.0입니다
```

numpy 실습

- **영화평점 데이터 실습**
  - https://grouplens.org/datasets/movielens
  - older datasets /MovieLens 1M Dataset / ml-1m.zip 다운로드
  - 다운로드 후 d:/data/movielens 디렉토리에 압축해제
  - 영화에 대한 평점 데이터 불러오기

```
data=np.loadtxt("D:/data/movielens/ratings.dat", delimiter="::",dtype=np.int64)
```

  - 처음 5행의 데이터만 확인

```
data[:5,:]
```

  - 데이터의 형태 확인(행,열)

```
data.shape
```

```
1 data.shape
(1000209, 4)
```

numpy 실습

- **영화평점 데이터 실습**
  - 전체 평균 평점 계산

```
mean_rating_total=data[:,2].mean() #data[:,2] 전체 행, 2번째 열
mean_rating_total
```

  - 사용자 아이디 수집

```
user_ids=np.unique(data[:,0])
print(user_ids)
print(len(user_ids))
```

numpy 실습

- **영화평점 데이터 실습**
  - 사용자별 평점평균

```
mean_values=[]
for user_id in user_ids:
    data_for_user=data[data[:,0]==user_id,:]
    value=data_for_user[:,2].mean()
    mean_values.append([user_id,value])

mean_values[:5] :# 전체에서 0~6번행까지만 출력
```

  - 리스트를 넘파이 배열로 변환

```
mean_array=np.array(mean_values,dtype=np.float32)
print(mean_array[:5])
print(mean_array.shape)
```

  - 평점결과 csv 파일로 저장

```
np.savetxt("data/movielens/resutl.csv",mean_array, fmt="%.1f",delimiter=",")
```

3. pandas

- **판다스(Pandas)**
  - Numpy와 함께 가장 많이 사용하는 데이터분석 라이브러리
  - 행과 열로 구성된 데이터 객체를 만들어 다룸
  - 안정적으로 대용량의 데이터들을 처리 도구
  - Pandas의 기본적으로 정의되는 자료구조 : Series, DataFrame
  - 빅데이터 분석에서 높은 수준의 성능, 대량의 데이터를 빠른 속도로 처리 가능
  - Series – 1차원 배열, 복수의 행(row)과 하나의 열로 구성, Index로 데이터에 접근
  - DataFrame - 2차원 배열(다수의 행과 열고 구성), 표 형태

```
import numpy as np
import pandas as pd
```

3. pandas

- **Series 생성**

```
import numpy as np
import pandas as pd

list1=[4, 3, 6, 8, 7]
s1=pd.Series(list1)
print('s1:\n', s1)
s2=pd.Series(list1, index=['a','b','c','d', 'e'])
print('s2:\n', s2)
print("type(s2):", type(s2)) # class type 확인
print("dtype :", s2.dtype) # 데이터 형 확인
print("s2.values :", s2.values) # 값만 확인
print("s2.index :", s2.index) # 인덱스 확인
print("s1.index :", s1.index)
```

3. pandas

- **Series :**
  - index 변경

```
# 인덱스 변경하기하기
s1.index=['one', 'two', 'three','four', 'five']
print(s1)
```

  - dictionary 데이터로 Series 생성

```
#dictionary 자료형으로 Series 만들기
sdata={'kim':3500, 'lee':5500, 'chio': 4500, 'hong': 6000}
s3=pd.Series(sdata)
print('s3 : \n', s3)
```

3. pandas

- **DataFrame : 생성**
  - 형식: pd.DataFrame(data, columns=[], index=[])

dictionary 데이터 활용

```
data={'names':['민준', '현우', '서연','동현','지현'],
      'years':[2013, 2014,2045,2016, 2015],
      'points':[1.5, 1.7, 3.6, 2.4, 2.9]
      }
df=pd.DataFrame(data)
df
```

2차원 배열 데이터 활용

```
data1=[['민준', '현우', '서연','동현','지현'],
       [2013, 2014,2045,2016, 2015],
       [1.5, 1.7, 3.6, 2.4, 2.9]]
data=np.array(data1).T
df1=pd.DataFrame(data)
df1
```

3. pandas

- **DataFrame : 생성**
  - columns 설정

```
df1.columns=['names', 'years','points']; df1
```

  - index 설정

```
df1.index=['one','two','three','four','five']; df1
```

  - index, columns, values 정보 보기

```
print(df1) # df와 print(df) 차이
print(df1.index)
print(df1.columns)
print(df1.values)
```

3. pandas

- **DataFrame : 생성**
  - csv, excel 파일로 데이터프레임 생성

```
# 파일-> 데이터프레임으로 읽어오기
chocolate_df=pd.read_csv('./chocolate_rating.csv')
characters_df=pd.read_excel('./characters.xlsx')

# 데이터프레임 -> 파일로 저장하기
characters_df.to_csv('data/characters.csv', index=False)
characters_df.to_excel('data/chocolate.xlsx', index=False)
```

3. pandas

- **DataFrame : 조회**
  - 데이터 보기

```
# columns 설정
chocolate_df.columns=['번호','연도','함유량','평점']

#데이터 전체 조회
print(chocolate_df)
chocolate_df
```

```
      번호    연도   함유량   평점
0      2  2015  0.70  2.75
1      3  2015  0.70  3.00
2      4  2015  0.70  3.50
3      5  2015  0.70  3.50
4      6  2014  0.70  2.75
...  ...   ...   ...   ...
1789  1791  2011  0.70  3.75
1790  1792  2011  0.65  3.00
1791  1793  2011  0.65  3.50
1792  1794  2011  0.62  3.25
1793  1795  2010  0.65  3.00

[1794 rows x 4 columns]
```

	번호	연도	함유량	평점
0	2	2015	0.70	2.75
1	3	2015	0.70	3.00
2	4	2015	0.70	3.50
3	5	2015	0.70	3.50
4	6	2014	0.70	2.75
...	...	...	...	...
1789	1791	2011	0.70	3.75
1790	1792	2011	0.65	3.00
1791	1793	2011	0.65	3.50
1792	1794	2011	0.62	3.25
1793	1795	2010	0.65	3.00

1794 rows × 4 columns

3. pandas

- **info() 함수 : DataFrame 정보 보기**
  - class, index, column 정보와 각 column별(name, non-null여부, 개수, 데이터 타입 확인)

```
df1.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, one to five
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   names   5 non-null      object
 1   years   5 non-null      object
 2   points  5 non-null      object
dtypes: object(3)
memory usage: 332.0+ bytes
```

3. pandas

- **DataFrame : 조회**
  - 데이터 보기

```
# head() : 앞부분 데이터 5개 보기
print(chocolate_df.head())

# tail() : 뒷부분 데이터 5개 보기
print(chocolate_df.tail())

# 인덱스 0~9까지 데이터 보기
print(chocolate_df[:10])

# 인덱스 10~19가지 데이터 보기
print(chocolate_df[10:20])
```

3. pandas

- **데이터 조회**
  - 열 이름, 인덱스 조회

```
# 열 이름 조회
print(chocolate_df.columns)

# index
print(chocolate_df.index)
```

✓ 0.0s

```
Index(['번호', '연도', '함유량', '평점'], dtype='object')
RangeIndex(start=0, stop=1794, step=1)
```

3. pandas

- **describe() 함수 : 데이터 요약 정보를 보여줌**

```
chocolate_df.describe()
```

✓ 0.0s

	번호	연도	함유량	평점
count	1794.000000	1794.000000	1794.000000	1794.000000
mean	898.500000	2012.323300	0.717051	3.185619
std	518.027509	2.926739	0.063214	0.478010
min	2.000000	2006.000000	0.420000	1.000000
25%	450.250000	2010.000000	0.700000	2.812500
50%	898.500000	2013.000000	0.700000	3.250000
75%	1346.750000	2015.000000	0.750000	3.500000
max	1795.000000	2017.000000	1.000000	5.000000

3. pandas

- **정렬**

```
# 인덱스 기준 정렬
# ascending : True(오름차순), False(내림차순)
df1.sort_index(axis=0, ascending=False)

# 열기준 정렬
df1.sort_values(by=['names','points'], ascending=False)
```

3. pandas

- **데이터 조회**
  - 형식 :

데이터프레임[조건식] : 시리즈로 출력하기
데이터프레임[[조건식]] : 데이터프레임으로 출력하기

  - 열 이름으로 데이터 조회

시리즈로 출력하기
데이터프레임[열이름]
데이터프레임.열이름
데이터프레임으로 출력하기
데이터프레임[[열이름1, 열이름2, …]] :

```
df1.names
df1['names']
df1[['names','points']]
```

3. pandas

- **데이터 조회**
  - iloc : 행열의 인덱스로 데이터 조회

    데이터프레임.iloc[행 인덱스, 열 인덱스]

```
print(df1.iloc[0:3, 0:2])
print(df1.iloc[[0,3], [0,2]])
print(df1.iloc[::2, :2])
```

3. pandas

- **데이터 조회**
  - loc : 행, 열 조회

    데이터프레임.loc[행 조건, 열 조건]

```
print(df1.loc['one'])
print(df1.loc['one':'three'])
print(df1.loc[['one', 'three']])
print(df1.loc['one':'four', ['names','points']])
```

```
#데이터프레임 정보보기
print(df1.info())
# 데이터프레임의 column의  데이터 형 변환
df1.points=df1.points.astype('Float64')
# 행조건 설정으로 데이터 조회하기
print(df1.loc[df1['points'] > 2.5, ['names','points']])
```

3. pandas

- **데이터 조회**
  - isin(): 일치하는 데이터 조회

```
df1[df1['names'].isin(['민준','현우'])]
df1[df1.names.isin(['민준','현우'])]
```

  - 조건 만족하는 데이터 조회

```
df1[df1.points>2]
```

  - 두 조건식 모두 만족(&)한 데이터 조회

```
df1[(df1.years=='2015') & (df1.points>2)]
df1.years=df1.years.astype('Int64')
df1[(df1.years==2015) & (df1.points>2)]
```

  - 두 조건식 중 하나만족(|)한 데이터 조회

```
df1[(df1.years==2014) | (df1.points>2)]
```

  - 특정 문자열이 포함된 데이터 조회

```
df1[df1.names.str.contains('현')]
```

3. pandas

- **데이터프레임 수정**
  - 데이터 수정

```
# 모든 행의  points 칼럼값 수정
df1['points']=df1['points']+0.5; print(df1)

# 특정행의 points 칼럼값 수정
df1.loc['one','points']=df1.loc['one', 'points']+1; df1

# 반복 연산자 사용
df1.loc['two':'three', 'points']=[0]*2; df1

# 여러 개 값 수정
df1.loc['two':'three', 'points']=[2.7, 4.6]; df1
```

3. pandas

- **데이터프레임 구조 변경**
  - 인덱스 변경

```
# names 컬럼 인덱스로 설정
df1.set_index('names'); df1
df1.set_index('names', inplace=True);df1
```

차이점?

  - 인덱스 복수

```
df1.reset_index(inplace=True); df1
```

  - 열 추가

```
df1['보너스']=df1.points*0.5; df1
df1['보너스1']=10; df1
df1['보너스2']=[1,2,3,4, 5]; df1
df1['보너스3']=[0.5, 0.2]; df1  #에러 발생(개수가 맞지 않음)
```

3. pandas

- **데이터프레임 구조 변경**
  - 열 삭제

```
df1.drop('보너스2', axis=1); df1
df1.drop('보너스2', axis=1, inplace=True);df1
```

  - 데이터 치환

```
df1['gender']=['남자','남자','여자','남자','여자'];df1
rep_cond={'gender':{'남자':1, '여자':2}}
df2=df1.replace(rep_cond); df2
```

3. pandas

- 데이터프레임 연산

```
df1.describe()
print(df1.years.value_counts())
print(df1.groupby('gender')['points'].sum())
print(df1.groupby('gender')[['points', '보너스', '보너스1']].mean())
print(df1.groupby(['years','gender'])['points'].median())
print(df1.groupby('gender')['points'].agg(['sum','mean','count']))
print(df1.groupby('gender').agg({'points':'count','보너스':'mean', '보너스1':'sum'}))
```

3. pandas

- **데이터프레임 조인**

```
#조인할 데이터프레임 만들기
jdf1=df1[['names','years','points']]; print(jdf1)
jdf2=df1[['names','years']]
jdf2['scores']=[80,85,75,65,90];print(jdf2)

# 조인의 기준이 될 인덱스 설정
jdf1.set_index('names',inplace=True);print(jdf1)
jdf2.set_index('names', inplace=True); print(jdf2)

# 조인
join_df1=jdf1.merge(jdf2, left_index=True, right_index=True); print(join_df1)
join_df2=jdf1.merge(jdf2); join_df2
```

3. pandas

- **데이터프레임 병합**

```
# 행방향 병합: 칼럼의 수가 같아야 함
cdf1=df1[:4];print(cdf1)
cdf2=df1[3:]; print(cdf2)
ccdf=pd.concat([cdf1, cdf2], axis=0); print(cdf)

# 열방향 병합 :  행의 수가 같아야 함
rcdf=pd.concat([jdf1, jdf2], axis=1); print(rcdf)
```

3. pandas

- **실습**

```
name=['허준호', '이가원','배규민','고고림','이새봄','이보람','이루리','오다현']
gender=['남자','여자','남자','남자','여자','여자','여자','여자']
age=[30,24,23,21,28,26,24,24]
hight=[183,162,179,182,160,163,157,172]
data={'이름':name,'성별':gender,'나이':age,'키':hight}
df=pd.DataFrame(data);df
```

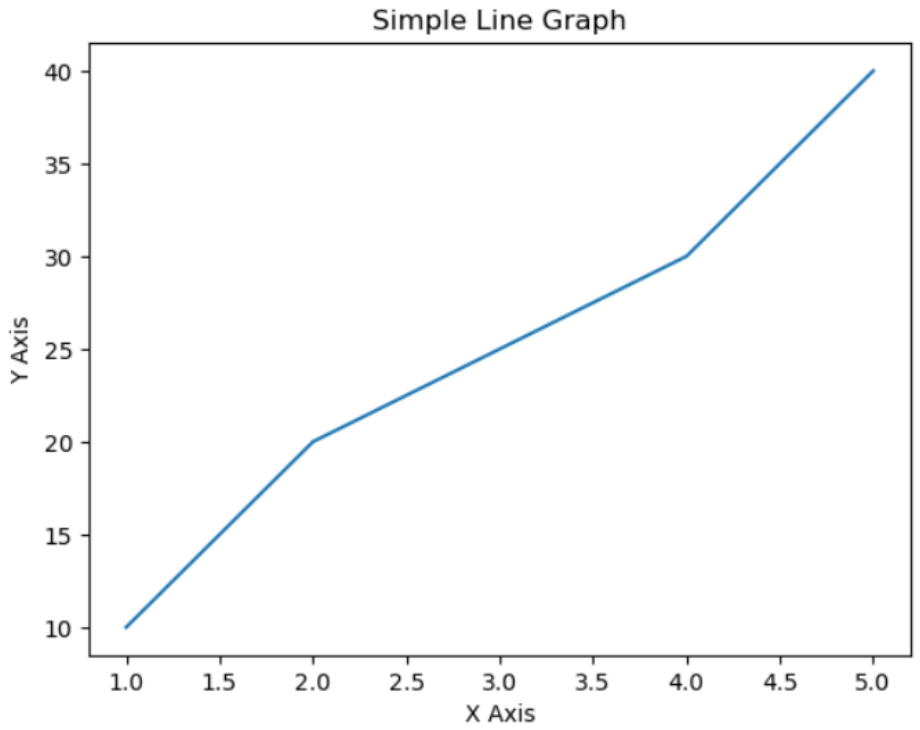
① 몸무게 column 추가 : [82.5, 50.0, 78.0 , 80.5, 48.0, 51.2, 46.0, 65.0]
② 키가 180 이상은 사람의 이름과 키를 조회하라.
③ 여자 중 키가 160보다 큰 사람을 조회하라.
④ 성별 키의 평균을 구하라
⑤ 성별 키와 몸무게의 평균. 합, 인원수를 구하라.
⑥ 이름과 성별 칼럼을 추출하여 df2를 생성한다.
⑦ df2에 [연락처] 칼럼을 추가한다.(데이터 : ['010-1111-2525',….. 8개]
⑧ df와 df2의 이름 칼럼을 인덱스로 설정한다.
⑨ df와 df2를 인덱스를 기준으로 조인(joindf)한다.
⑩ 데이터프레임을 csv파일(test_df.csv)로 저장한다.
⑪ 저장된 test_df.csv) 파일을 데이터프레임으로 로드한다.

4. matplotlib

- **Matplotlib 설치**

```
! pip install matplotlib
```

- **기본적인 그래프 그리기**

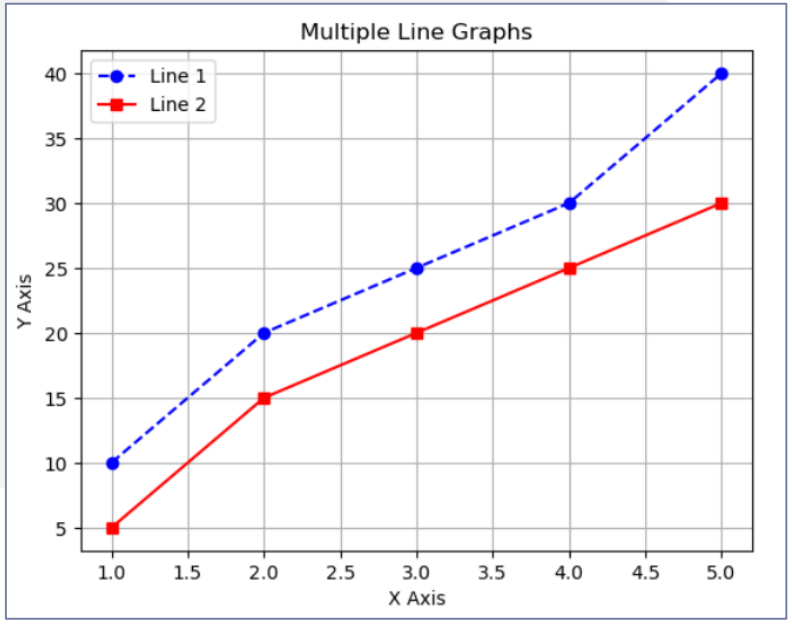



```
# 데이터 생성
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]

# 그래프 그리기
plt.plot(x, y)

# 그래프 제목 및 레이블 추가
plt.title("Simple Line Graph")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")

# 그래프 출력
plt.show()
```

4. matplotlib

- 여러 개의 선 그리기

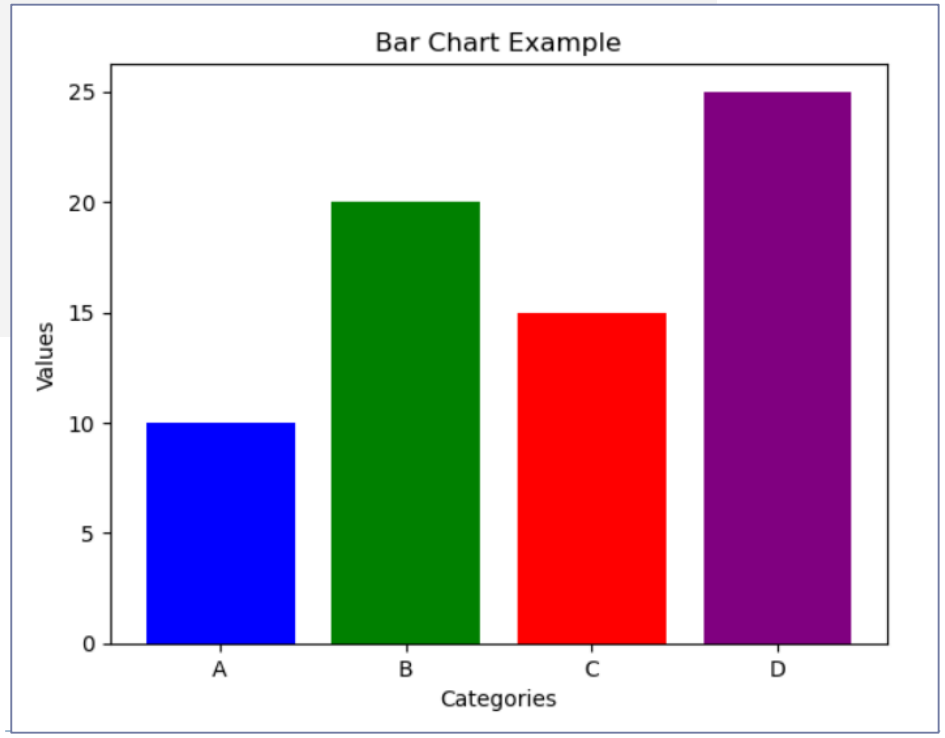
```
x = [1, 2, 3, 4, 5]
y1 = [10, 20, 25, 30, 40]
y2 = [5, 15, 20, 25, 30]

plt.plot(x, y1, label="Line 1", color='blue', linestyle='--', marker='o')
plt.plot(x, y2, label="Line 2", color='red', linestyle='-', marker='s')

plt.title("Multiple Line Graphs")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.legend()  # 범례 추가
plt.grid(True)  # 격자 추가

plt.show()
```



4. matplotlib

- **Bar 차트 그리기**

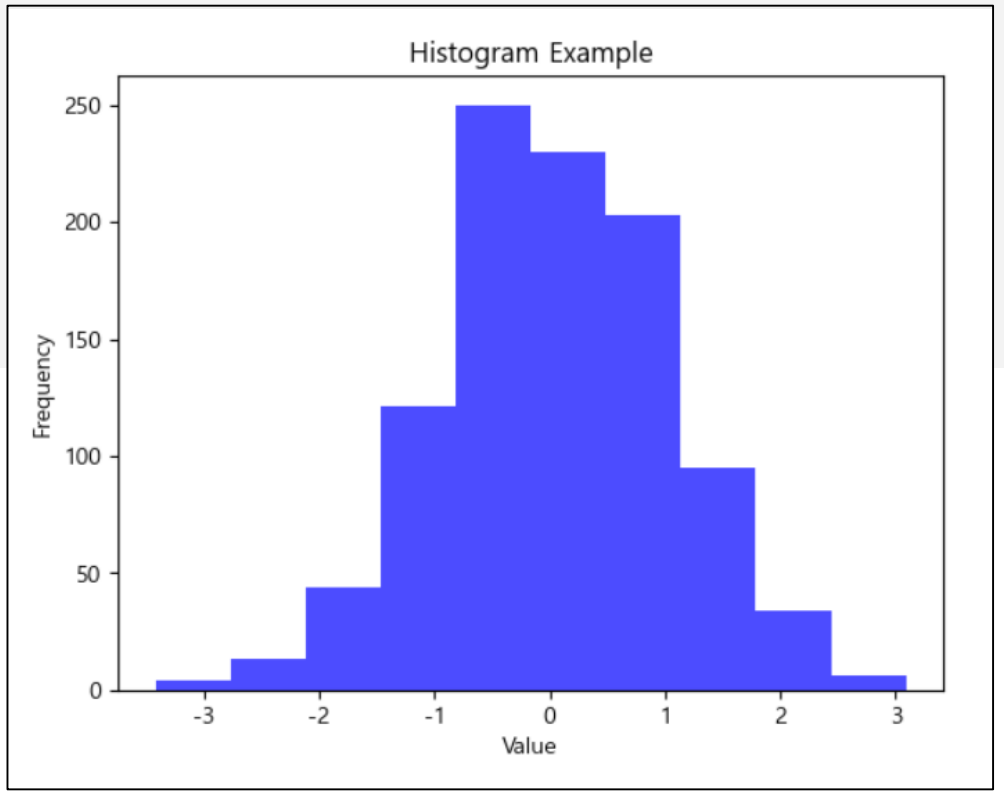
```
categories = ['A', 'B', 'C', 'D']
values = [10, 20, 15, 25]

plt.bar(categories, values, color=['blue', 'green', 'red', 'purple'])

plt.title("Bar Chart Example")
plt.xlabel("Categories")
plt.ylabel("Values")

plt.show()
```



4. matplotlib

- **Histogram 차트**

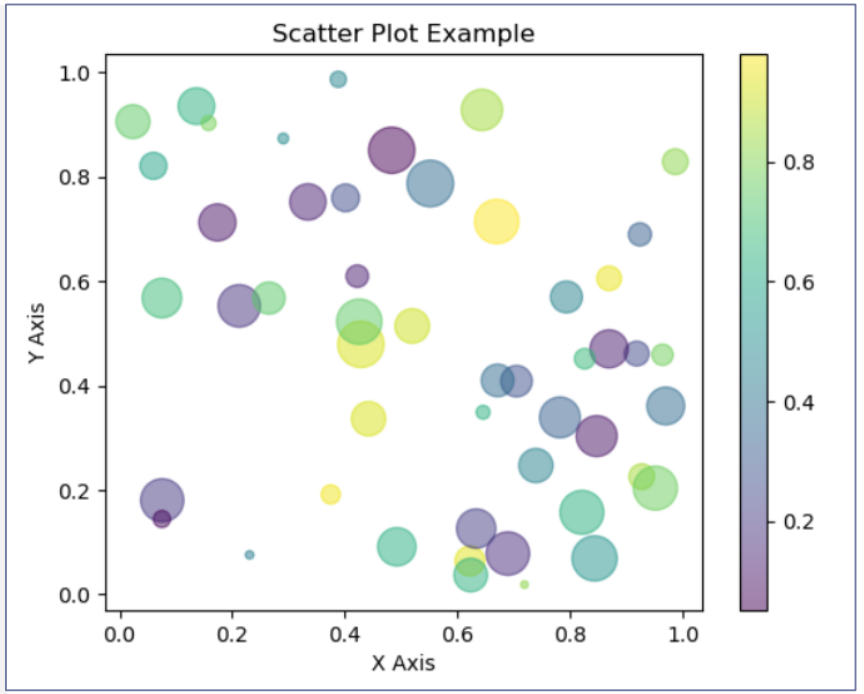
```
data = np.random.randn(1000)  # 정규 분포를 따르는 난수 생성

plt.hist(data, bins=10, color='blue', alpha=0.7)  # bins는 막대 개수, alpha는 투명도

plt.title("Histogram Example")
plt.xlabel("Value")
plt.ylabel("Frequency")

plt.show()
```



4. matplotlib

- **산점도(Scatter) 차트**

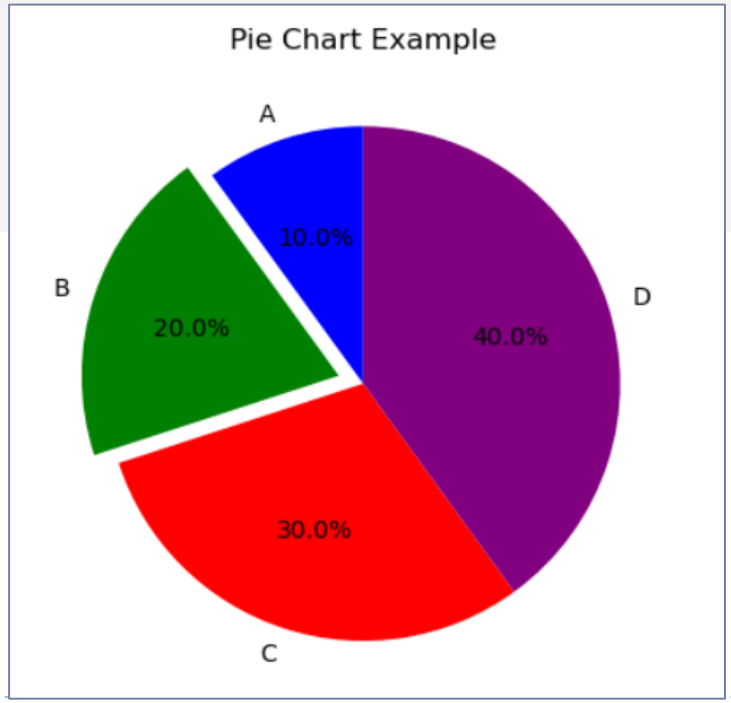
```
x = np.random.rand(50)
y = np.random.rand(50)
colors = np.random.rand(50)  # 색상 지정
sizes = np.random.rand(50) * 500  # 점 크기 지정

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='viridis')

plt.title("Scatter Plot Example")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")

plt.colorbar()  # 색상 바 추가
plt.show()
```



4. matplotlib

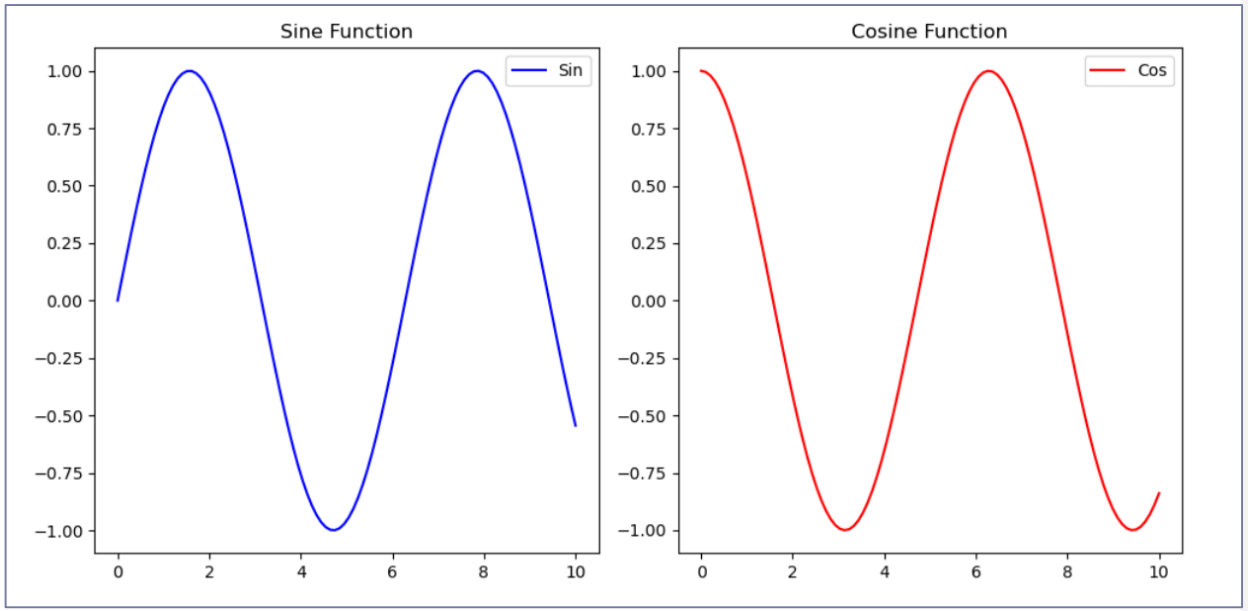
- **파이(Pie) 차트**

```
labels = ['A', 'B', 'C', 'D']
sizes = [10, 20, 30, 40]
colors = ['blue', 'green', 'red', 'purple']
explode = (0, 0.1, 0, 0)  # 두 번째 조각 강조

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', explode=explode, startangle=90)

plt.title("Pie Chart Example")

plt.show()
```



4. matplotlib

- 서브플롯(Subplots)




```python
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

plt.figure(figsize=(10, 5))  # 그래프 크기 설정

# 첫 번째 그래프 (사인 함수)
plt.subplot(1, 2, 1)  # (행, 열, 위치)
plt.plot(x, y1, label="Sin", color='blue')
plt.title("Sine Function")
plt.legend()

# 두 번째 그래프 (코사인 함수)
plt.subplot(1, 2, 2)
plt.plot(x, y2, label="Cos", color='red')
plt.title("Cosine Function")
plt.legend()

plt.tight_layout()  # 그래프 간격 조정

plt.show()
```

4. matplotlib

- **그래프 저장하기**

plt.savefig("graph.png", dpi=300) # dpi는 해상도 설정

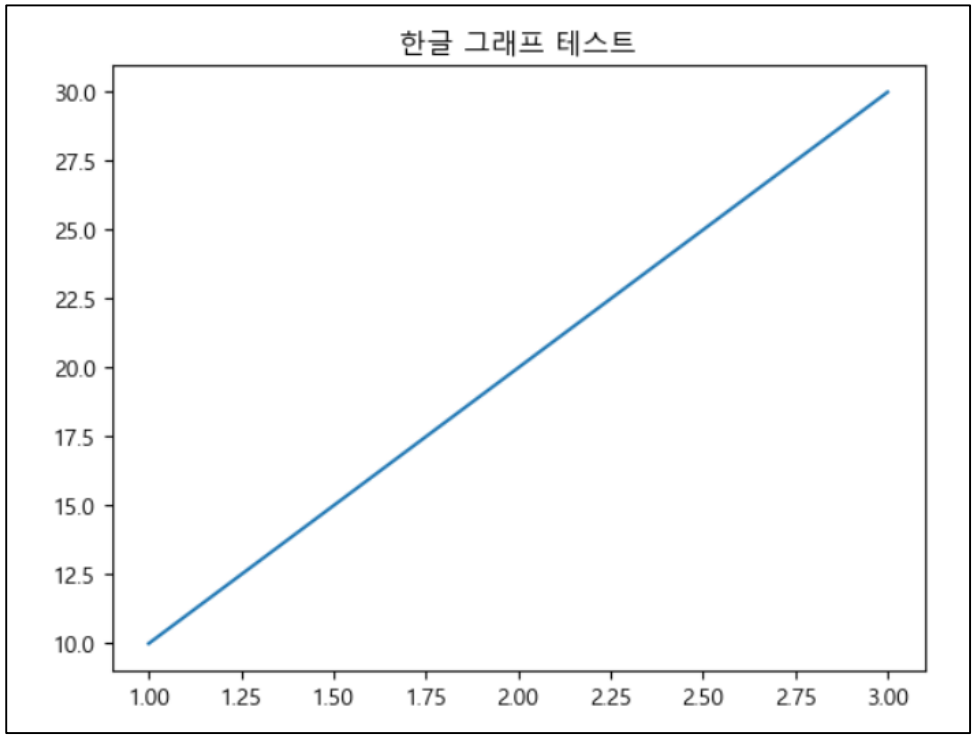
- plt.show() 명령보다 먼저 수행해야 그래프 이미지 저장됨

```
labels = ['A', 'B', 'C', 'D']
sizes = [10, 20, 30, 40]
colors = ['blue', 'green', 'red', 'purple']
explode = (0, 0.1, 0, 0)  # 두 번째 조각 강조

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', explode=explode, startangle=90)

plt.title("Pie Chart Example")

plt.savefig("graph.png", dpi=300)  # dpi는 해상도 설정

plt.show()
```

4. matplotlib

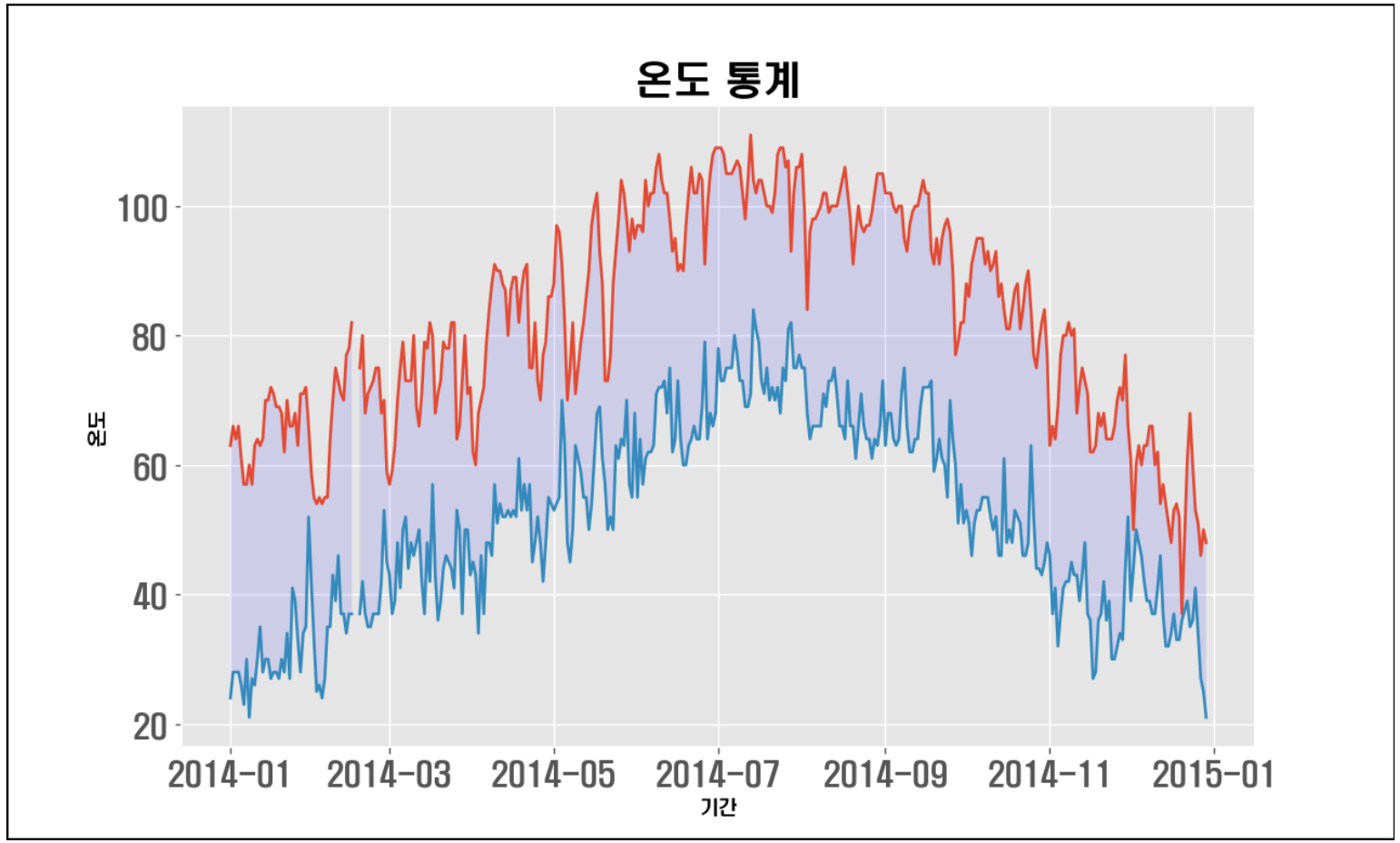
- **한글 폰트 설정**
  - 차트에 한글을 표시하려면 반드시 한글 폰트 설정을 해야함

```
import matplotlib.font_manager as fm

# Windows: 'Malgun Gothic', Mac: 'AppleGothic'
plt.rc('font', family='Malgun Gothic')
# 음수 기호 깨짐 방지
plt.rc('axes', unicode_minus=False)

# 그래프 테스트
plt.plot([1, 2, 3], [10, 20, 30])
plt.title("한글 그래프 테스트")
plt.show()
```



4. matplotlib : 실습

- temperature_2014.csv파일을 읽어서 날짜별 최저온도(low)와 최고온도(high)를 이용하여 다음과 같은 차트를 작성하여 'chart.png ' 로 저장하고 화면에 출력한다.



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

# csv 파일 데이터프레임으로 읽기
data=pd.read_csv("data/temperature_2014.csv")

# 데이터프레임 데이터확인
data
# 데이터프레임 정보 확인
data.info()
```

```
# PST 필드 타입 str->datetiem64 로 형변환
data['PST']=pd.to_datetime(data['PST'])
data.info()
```

```
# 한글 사용을 위한 설정 : # Windows: 'Malgun Gothic', Mac: 'AppleGothic'
plt.rc('font', family='Malgun Gothic')
# 음수 기호 깨짐 방지
plt.rc('axes', unicode_minus=False)
# 차트 그림영역 생성
flg=plt.figure(dpi=128,figsize=(10,6))
# 날짜를 x, 최고온도를 y 값으로, color는 red, 불투명도 0.5로 그리기
plt.plot(list(data['PST']),list(data['Max TemperatureF']), c='red', alpha=0.5)
# 날짜를 x, 최저온도를 y 값으로, color는 blue, 불투명도 0.5로 그리기
plt.plot(list(data['PST']),list(data['Min TemperatureF']), c='blue', alpha=0.5)
# 최고온도와 최저온도 사이의 공간을 색칠함
plt.fill_between(data['PST'], list(data['Max TemperatureF']),
                 list(data['Min TemperatureF']), facecolor='green',alpha=0.1)
plt.title('온도 통계',fontsize=20)
plt.xlabel("기간", fontsize=16)
plt.ylabel("온도",fontsize=16)
plt.tick_params(axis='both', labelsize=12)
plt.savefig("data/chart.png")  # 차트 저장
plt.show()
```