zhining@naver.com

Ch3. 데이터 분석을 위한 파이썬 응용 문법

zhining@naver.com

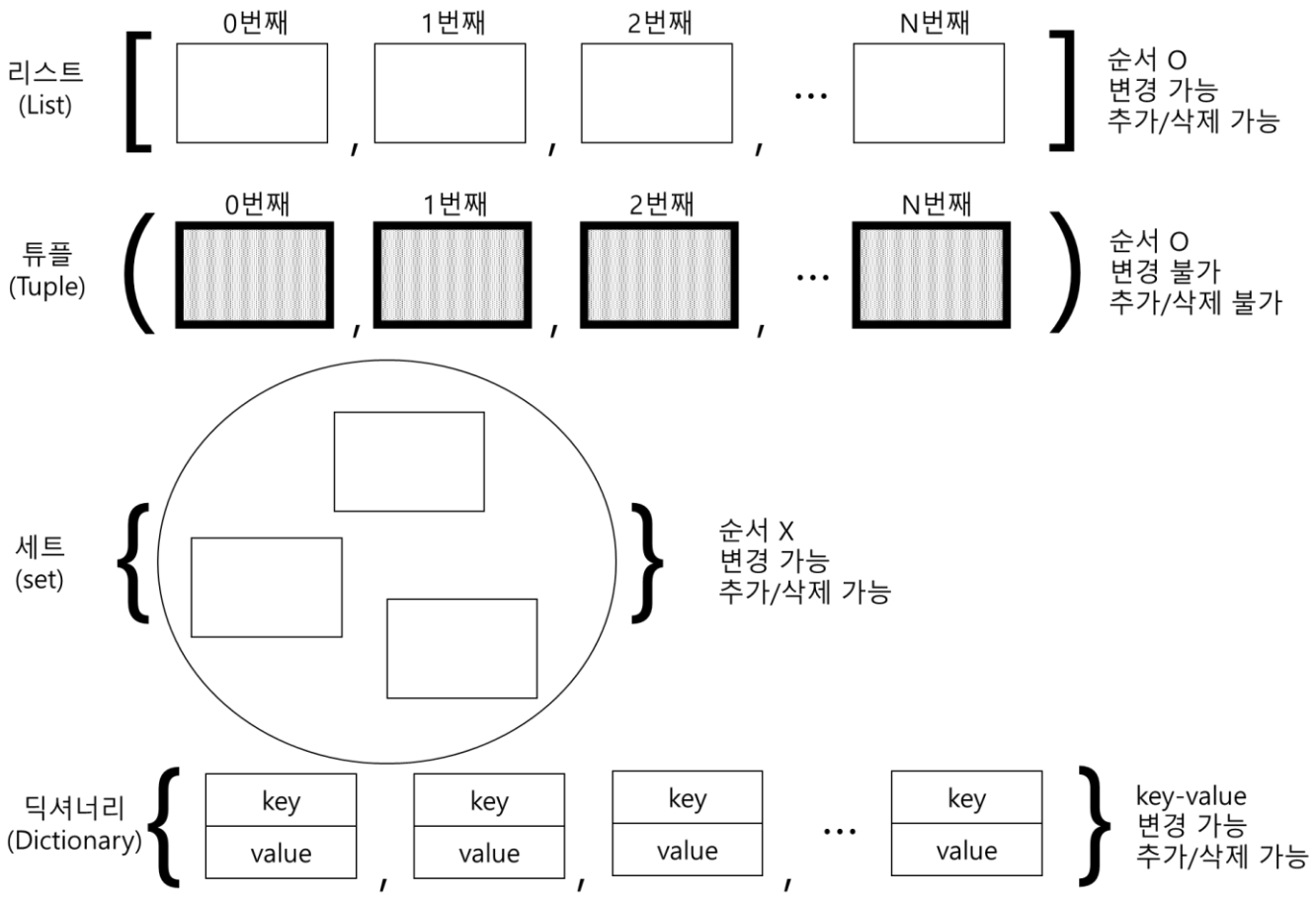
- code: https://github.com/zzhining/python_data_basic

zhining@naver.com

Review – Python 기본 문법

zhining@naver.com

- 컨테이너 타입 정리



zhining@naver.com

- 컨테이너 타입: 여러 개의 값을 다루는 자료형
  - 리스트(list) : 데이터를 연속적으로 관리, 값을 바꿀 수 있음
  - 튜플(tuple) : 데이터를 연속적으로 관리, 값을 바꿀 수 없음
  - 세트(set) : 집합과 같은 속성
  - 딕셔너리 : key-value 쌍

리스트	[‘가’, ‘나’, ‘다’, ‘라’]
튜플	(‘가’, ‘나’, ‘다’, ‘라’)
세트	{‘가’, ‘나’, ‘다’, ‘라’}
딕셔너리	{‘가’:123, ‘나’:456, ‘다’:789}

zhining@naver.com

■ 조건문, 반복문, 함수

[문제] 동요 '산토끼'에서 '토'는 몇 번이나 나올까요? 함수로 작성해보세요.

산토끼 토끼야. 어디를 가느냐. 깡충깡충 뛰면서. 어디를 가느냐.
산고개 고개를. 나혼자 넘어서. 토실토실 알밤을. 주워 올 테야

zhining@naver.com

[문제] 동요 '산토끼'에서 '토'는 몇 번이나 나올까요? 함수로 작성해보세요.

산토끼 토끼야. 어디를 가느냐. 깡충깡충 뛰면서. 어디를 가느냐.
산고개 고개를. 나혼자 넘어서. 토실토실 알밤을. 주워 올 테야

```
def get_char_count(lyric, char):
    count = 0
    for l in lyric:
        if(l == char):
            count = count + 1
    return count
```

```
lyric = """산토끼 토끼야. 어디를 가느냐. 깡충깡충 뛰면서. 어디를 가느냐.
산고개 고개를. 나혼자 넘어서. 토실토실 알밤을. 주워 올 테야."""

print(get_char_count(lyric, '토'))
```

```
4
```

zhining@naver.com

01 NumPy

zhining@naver.com

- 효율?!

zhining@naver.com

- NumPy
  - Numerical Python
  - 다차원 배열의 연산 기능을 모아 놓은 꾸러미
  - 수치 연산을 위한 다양한 기능 제공

```
import numpy as np
```

```
from numpy import *
```

square()	제곱
sqrt()	제곱근(루트)
exp()	지수승
log()	로그
add()	덧셈
sum()	합계
cumsum()	누적합
mean()	평균
var()	편차
std()	표준편차
min()	최소값
max()	최대값

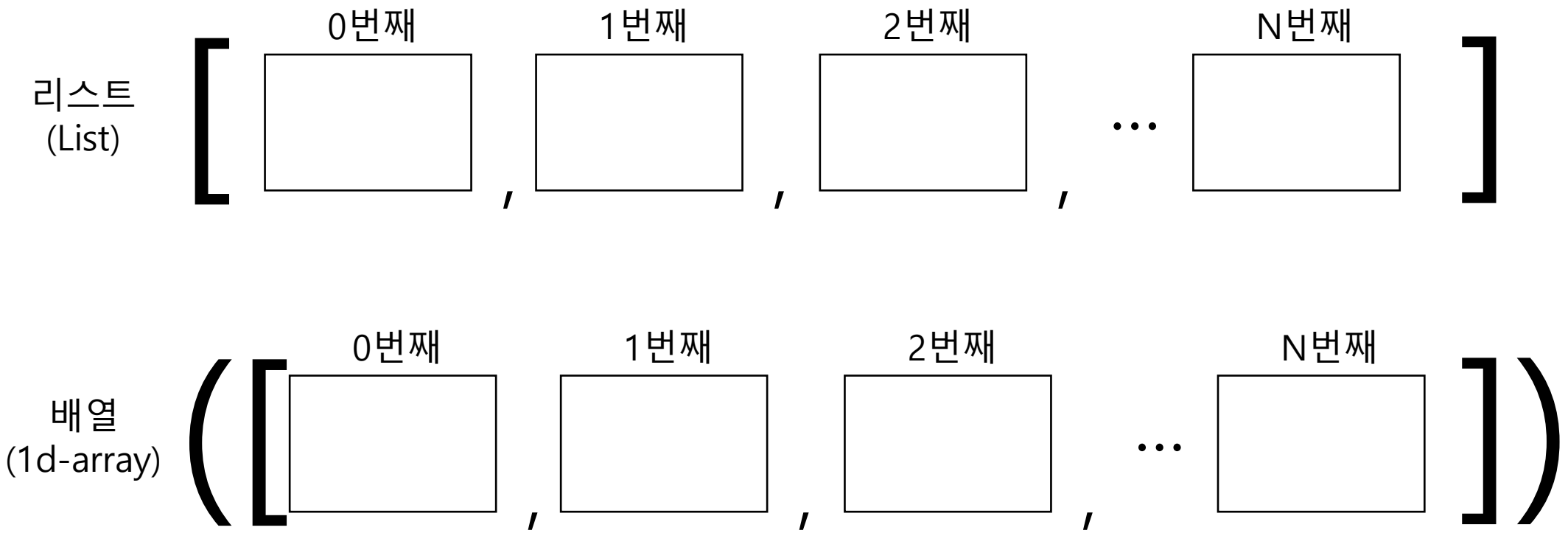
zhining@naver.com

- NumPy
  - Numerical Python
  - 다차원 배열의 연산 기능을 모아 놓은 꾸러미
  - 수치 연산을 위한 다양한 기능 제공

  - NumPy의 배열 연산 특징
    - 적은 메모리: 데이터를 연속된 메모리 블록에 저장
    - 빠른 처리: 내부 연산이 C언어로 작성되어 메모리를 직접 조작

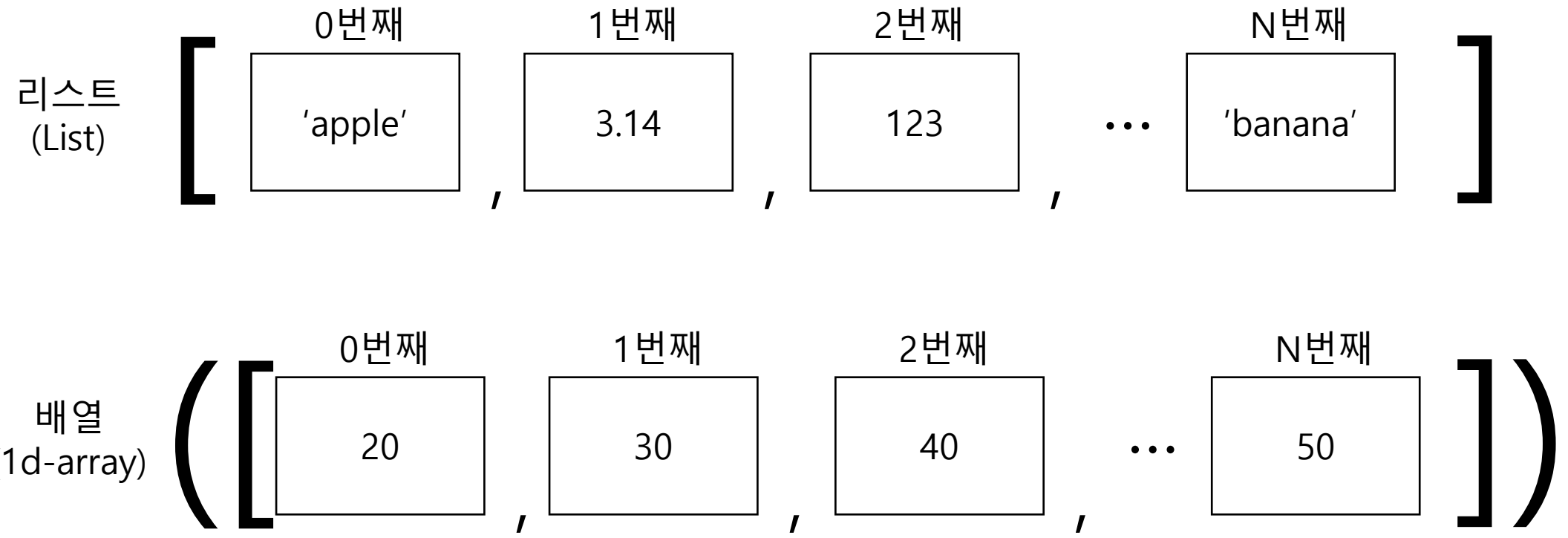
zhining@naver.com

- 배열



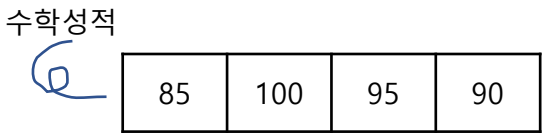
zhining@naver.com

- 배열
  - 배열은 같은 종류의 데이터를 담는다

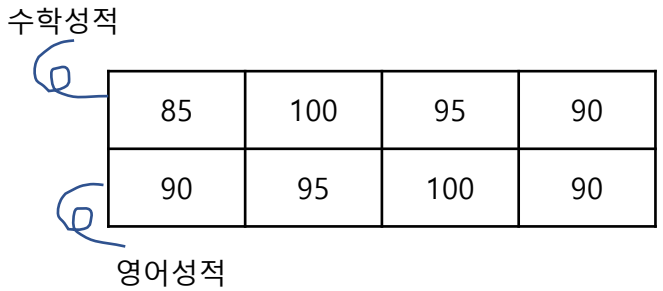


zhining@naver.com

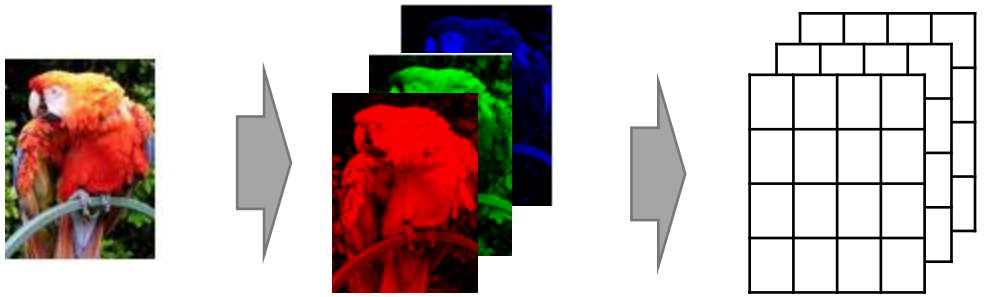
- 배열
  - 배열은 차원을 갖는다.



1차원 배열



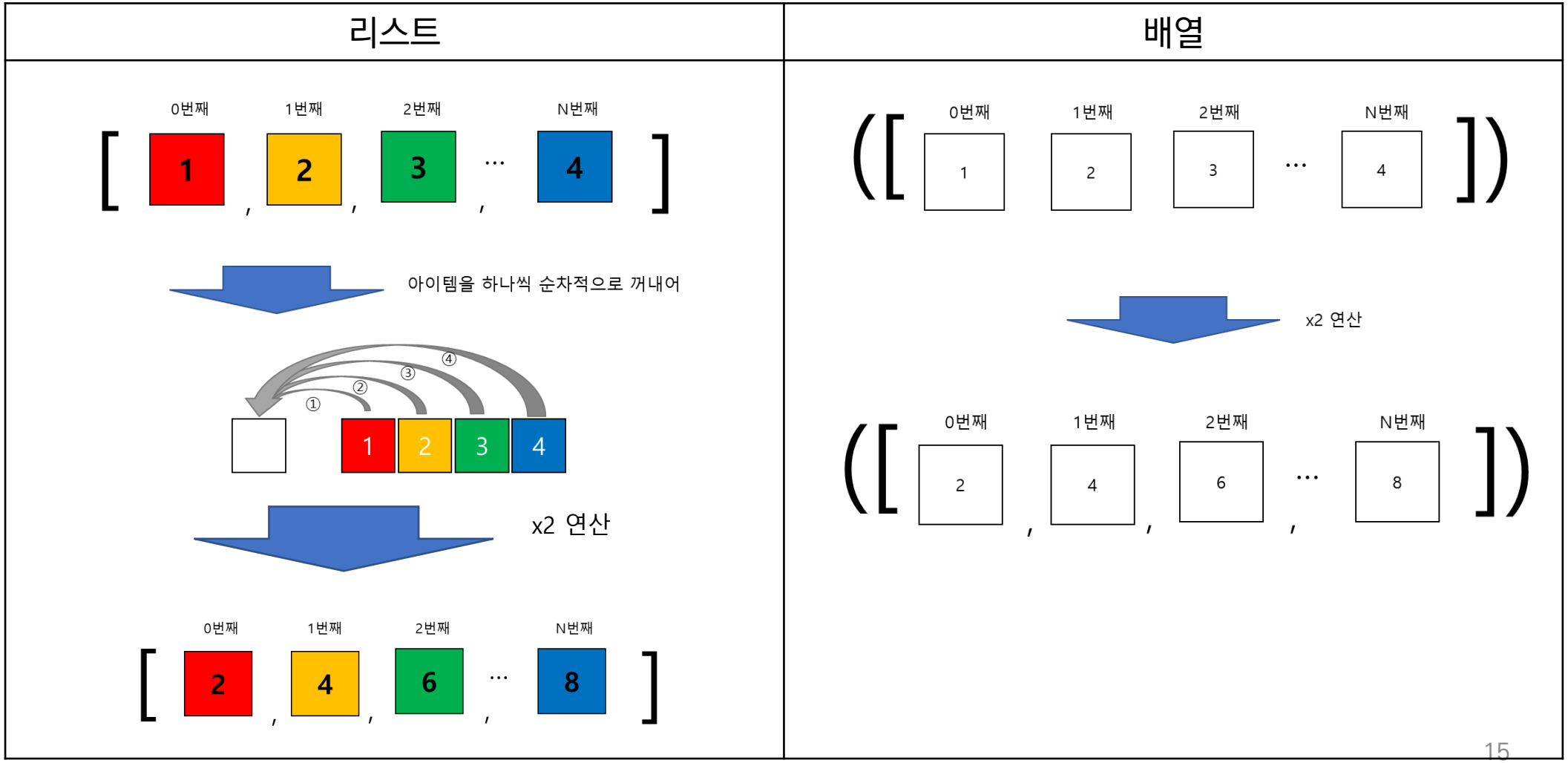
2차원 배열



3차원 배열

zhining@naver.com

- 배열
  - 배열은 빠르게 연산을 처리한다.(반복문을 사용하지 않는다!)



zhining@naver.com

- 배열의 생성

코드	설명
np.array([1,2,3])	초기화할 값을 지정하여 배열 생성
np.zeros()	값을 0으로 초기화 하여 배열 생성
np.ones()	값을 1로 초기화 하여 배열 생성
np.arange()	수의 순차적인 증감을 이용하여 배열 생성
np.rand()	랜덤한 숫자로 배열 생성

zhining@naver.com

▪ 배열의 생성

np.array()

```
import numpy as np

list1 = [1,2,3]
arr1 = np.array(list1)
arr1
```

```
array([1, 2, 3])
```

np.zeros()

```
np.zeros(5)
```

```
array([0., 0., 0., 0., 0.])
```

```
np.zeros((2,3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

np.ones()

```
np.ones(7)
```

```
array([1., 1., 1., 1., 1., 1., 1.])
```

np.empty()

```
np.empty(5)
```

```
array([0., 0., 0., 0., 0.])
```

np.arange()

```
np.arange(10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.arange(20,30,2)
```

```
array([20, 22, 24, 26, 28])
```

np.random.rand()

```
np.random.rand(2,3)
```

```
array([[0.80611079, 0.74013366, 0.17053214],
       [0.90518732, 0.91322483, 0.23444548]])
```

zhining@naver.com

- 배열의 생성

np.reshape()

```
arr2 = np.arange(16)
arr2
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

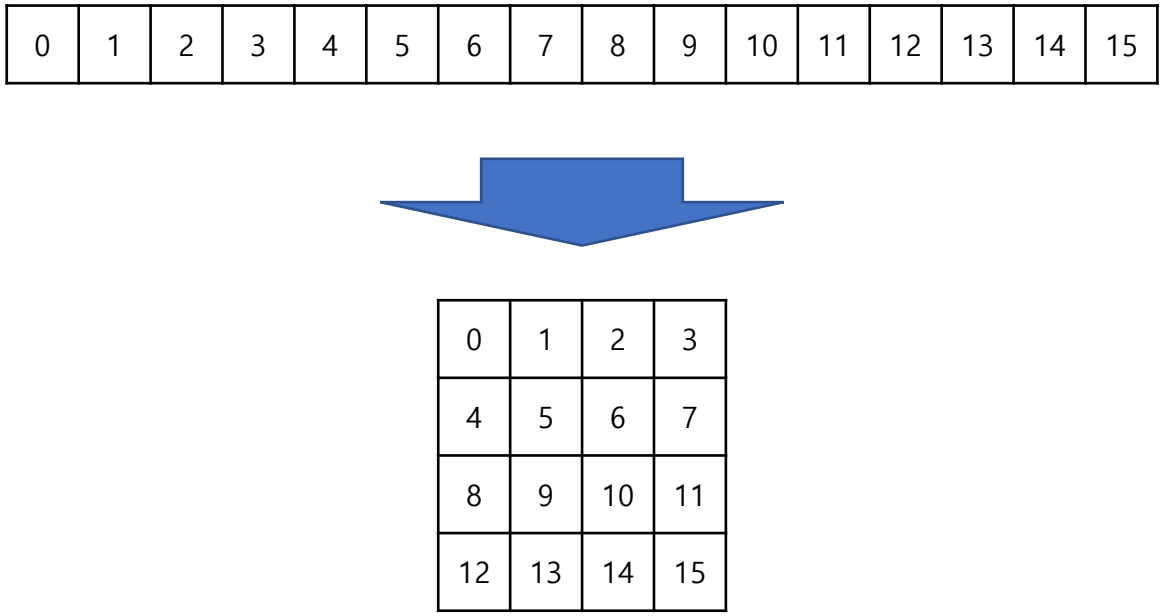
```
arr2.reshape(4,4)
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

```
arr2.reshape(2,2,4)
```

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7]],

       [[ 8,  9, 10, 11],
        [12, 13, 14, 15]]])
```



zhining@naver.com

▪ 배열의 생성

np.reshape()

```
arr2 = np.arange(16)
arr2
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

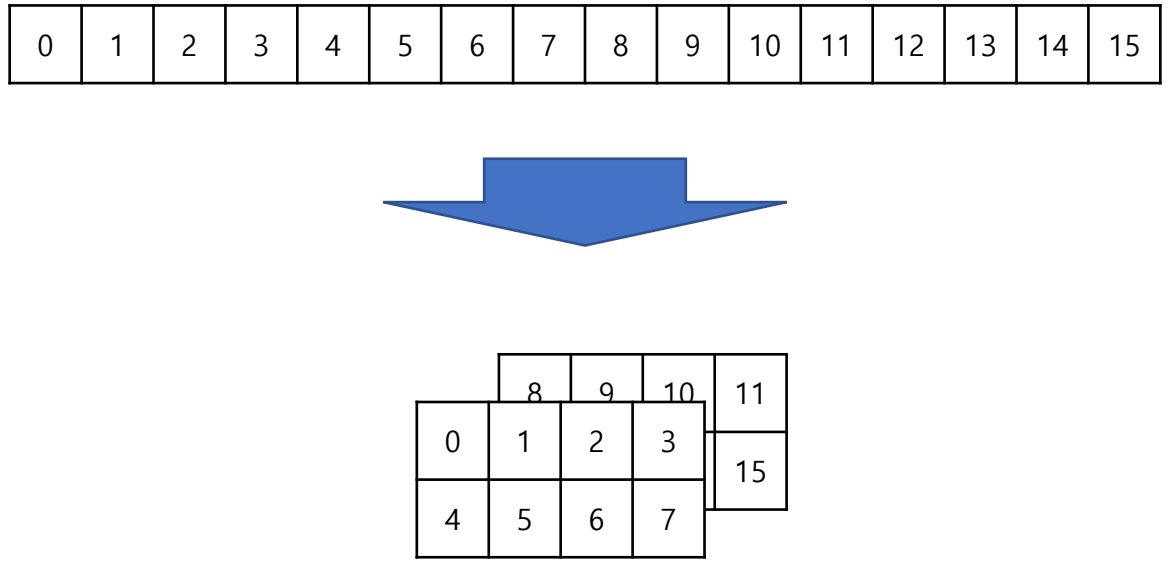
```
arr2.reshape(4,4)
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

```
arr2.reshape(2,2,4)
```

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7]],

       [[ 8,  9, 10, 11],
        [12, 13, 14, 15]]])
```



zhining@naver.com

- 배열의 생성

shape, dtype(), astype()

```
arr = np.arange(16)
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

```
arr.shape
```

```
(16,)
```

```
arr.dtype
```

```
dtype('int32')
```

```
arr.astype(float)
```

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.,
       13., 14., 15.])
```

zhining@naver.com

■ 선택

• 인덱스를 사용한 선택

```
import numpy as np
arr = np.arange(16)
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

```
# 기본 인덱스
arr[5]
```

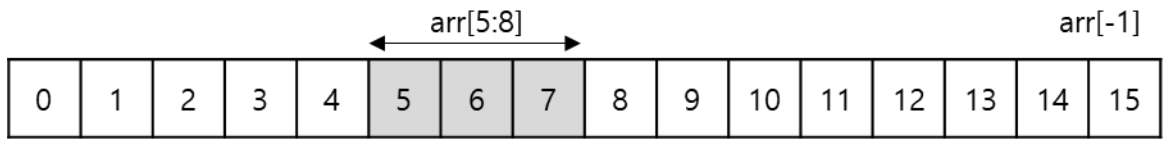
```
5
```

```
# 마이너스 인덱스
arr[-1]
```

```
15
```

```
# 인덱스 범위
arr[5:8]
```

```
array([5, 6, 7])
```



zhining@naver.com

■ 선택

• 인덱스를 사용한 선택

```
# 다차원 배열
arr2 = np.arange(16).reshape(4,4) # 메쏘드 체인
arr2
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

```
arr2[1]
```

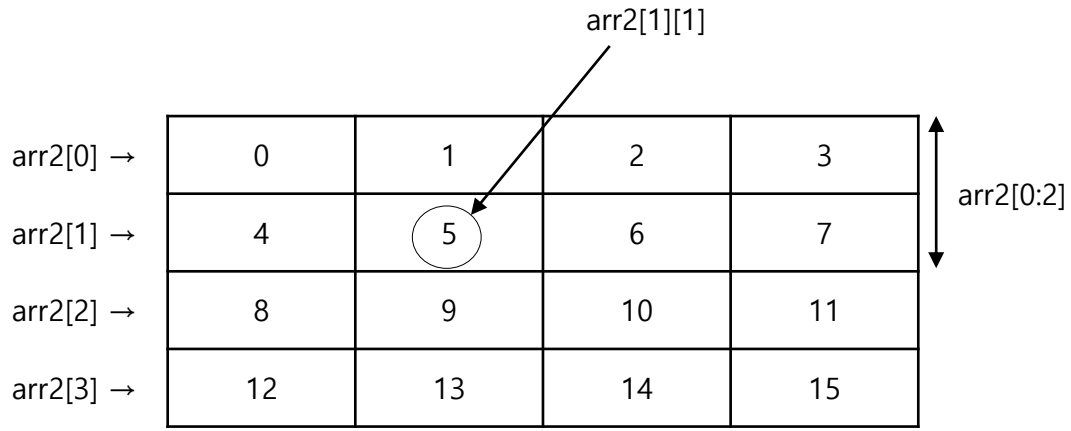
```
array([4, 5, 6, 7])
```

```
arr2[1][1]
```

```
5
```

```
arr2[1,2] #arr2[1][2]와 동일
```

```
6
```



zhining@naver.com

■ 선택

• 불리언 인덱스를 사용한 선택

```
scores = [80, 90, 70, 65, 85, 95, 90, 80, 75, 80]
score_arr = np.array(scores)
score_arr
```

```
array([80, 90, 70, 65, 85, 95, 90, 80, 75, 80])
```

```
score_arr >= 90
```

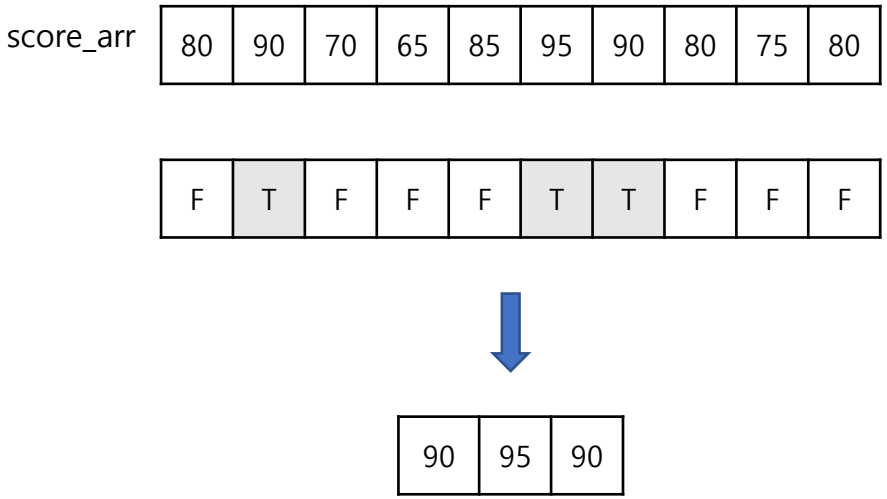
```
array([False,  True, False, False, False,  True,  True, False, False,
       False])
```

```
score_arr[score_arr >= 90]
```

```
array([90, 95, 90])
```

```
score_arr[score_arr >= 90] = 999
score_arr
```

```
array([ 80, 999,  70,  65,  85, 999, 999,  80,  75,  80])
```



zhining@naver.com

▪ 주요 연산

기본 사칙 연산

```
arr = np.arange(1, 5)
arr
```

```
array([1, 2, 3, 4])
```

```
arr + 2
```

```
array([3, 4, 5, 6])
```

```
arr * 2
```

```
array([2, 4, 6, 8])
```

```
list = [1, 2, 3, 4]
list
```

```
[1, 2, 3, 4]
```

```
list * 2
```

```
[1, 2, 3, 4, 1, 2, 3, 4]
```

```
arr1 = np.arange(5, 9)
arr1
```

```
array([5, 6, 7, 8])
```

```
arr + arr1
```

```
array([ 6,  8, 10, 12])
```

```
list1 = [5, 6, 7, 8]
list + list1
```

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

zhining@naver.com

▪ 주요 연산

주요 산술 연산 함수

```
# 제곱
np.square(arr)
```

```
array([ 1,  4,  9, 16], dtype=int32)
```

```
# 제곱근(루트)
np.sqrt(arr)
```

```
array([1.        , 1.41421356, 1.73205081, 2.        ])
```

```
# exp(지수승)
np.exp(arr)
```

```
array([ 2.71828183,  7.3890561 , 20.08553692, 54.59815003])
```

```
# log
np.log(arr)
```

```
array([0.        , 0.69314718, 1.09861229, 1.38629436])
```

```
# 더하기
np.add(arr, arr1)
```

```
array([ 6,  8, 10, 12])
```

```
#합계
np.sum(arr)
```

```
10
```

```
#누적합
np.cumsum(arr)
```

```
array([ 1,  3,  6, 10], dtype=int32)
```

```
# 평균
np.mean(arr)
```

```
2.5
```

```
# 편차
np.var(arr)
```

```
1.25
```

```
# 표준편차
np.std(arr)
```

```
1.118033988749895
```

zhining@naver.com

■ 변형

```
arr = np.arange(10).reshape(2,5)
arr
```

```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

```
arr.transpose()
```

```
array([[0, 5],
       [1, 6],
       [2, 7],
       [3, 8],
       [4, 9]])
```

```
arr.T #arr.transpose()와 동일
```

```
array([[0, 5],
       [1, 6],
       [2, 7],
       [3, 8],
       [4, 9]])
```

0	1	2	3	4
5	6	7	8	9

0	5
1	6
2	7
3	8
4	9

zhining@naver.com

- 저장

```
arr = np.arange(10).reshape(2,5)
arr
```

```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

```
np.save('array', arr)
```

```
load_arr = np.load('array.npy')
```

```
load_arr
```

```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```


array.npy

zhining@naver.com

02 Pandas

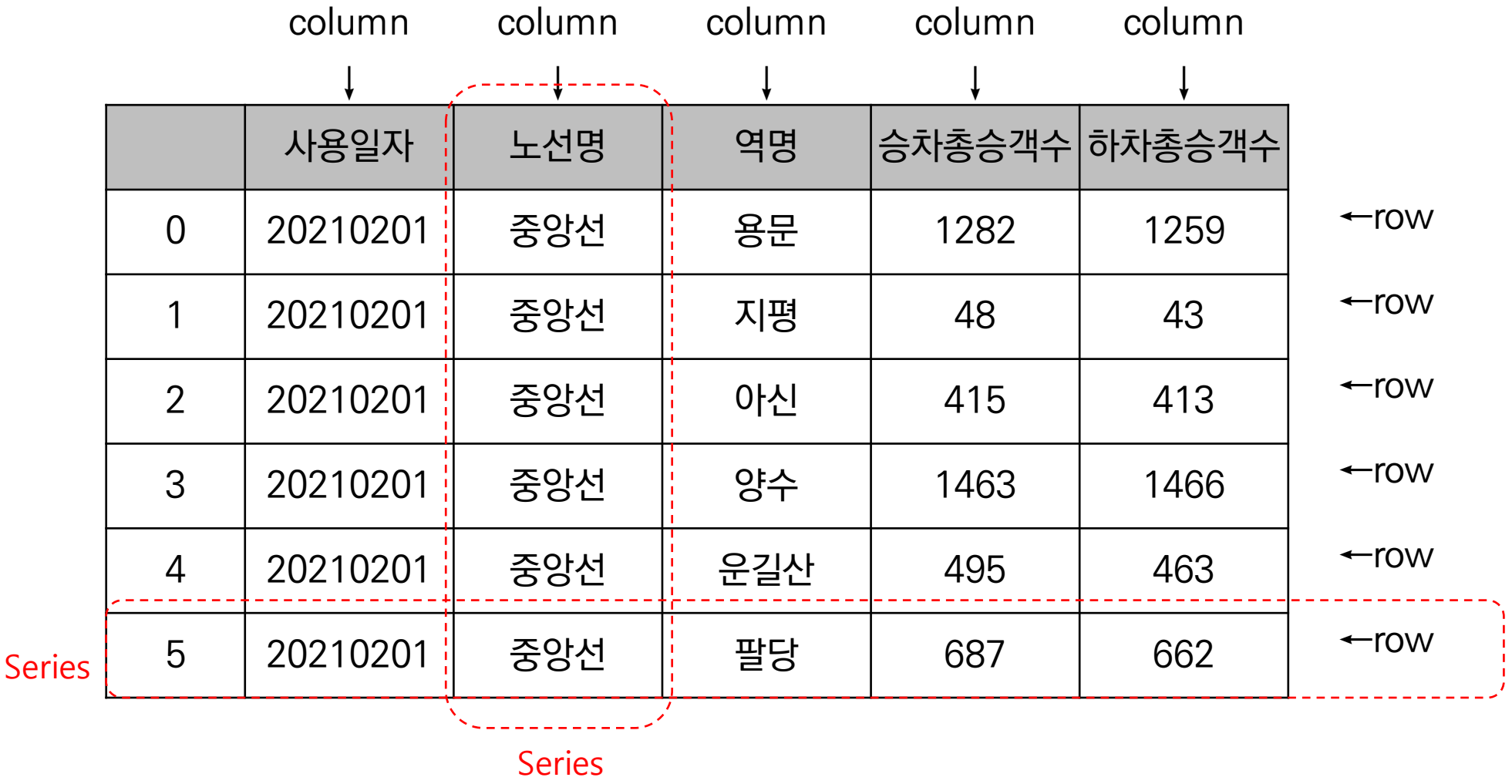
zhining@naver.com

- Table

	사용일자	노선명	역명	승차총승객수	하차총승객수
0	20210201	중앙선	용문	1282	1259
1	20210201	중앙선	지평	48	43
2	20210201	중앙선	아신	415	413
3	20210201	중앙선	양수	1463	1466
4	20210201	중앙선	운길산	495	463
5	20210201	중앙선	팔당	687	662

zhining@naver.com

- Table



	사용일자	노선명	역명	승차총승객수	하차총승객수
0	20210201	중앙선	용문	1282	1259
1	20210201	중앙선	지평	48	43
2	20210201	중앙선	아신	415	413
3	20210201	중앙선	양수	1463	1466
4	20210201	중앙선	운길산	495	463
5	20210201	중앙선	팔당	687	662

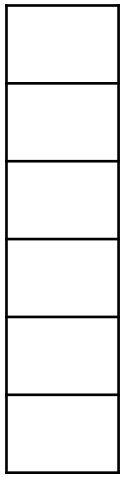
zhining@naver.com

- Pandas
  - Series, DataFrame

```
import pandas as pd
```

zhining@naver.com

- Array, Series, DataFrame



1D-array

Series

DataFrame

zhining@naver.com

▪ 생성

```
data = {'학번':range(2000,2010),
    '성적': [85, 95, 75, 70, 100, 100, 95, 85, 80, 85]}
df = pd.DataFrame(data)
df
```

	학번	성적
0	2000	85
1	2001	95
2	2002	75
3	2003	70
4	2004	100
5	2005	100
6	2006	95
7	2007	85
8	2008	80
9	2009	85

```
df = pd.DataFrame(data, columns =['성적', '학번'])
df
```

	성적	학번
0	85	2000
1	95	2001
2	75	2002
3	70	2003
4	100	2004
5	100	2005
6	95	2006
7	85	2007
8	80	2008
9	85	2009

zhining@naver.com

■ 생성

```
data = pd.read_csv('CARD_SUBWAY_MONTH_202102.csv')
data
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
20210201	중앙선	용문	1282	1259	20210204	NaN
20210201	중앙선	지평	48	43	20210204	NaN
20210201	중앙선	아신	415	413	20210204	NaN
20210201	중앙선	양수	1463	1466	20210204	NaN
20210201	중앙선	운길산	495	463	20210204	NaN
...	...	...	...	...	...	...
20210228	중앙선	오빈	216	236	20210303	NaN
20210228	중앙선	양평	3299	3187	20210303	NaN
20210228	중앙선	원덕	322	304	20210303	NaN
20210228	중앙선	용문	1899	1716	20210303	NaN
20210228	중앙선	지평	30	31	20210303	NaN

16751 rows × 6 columns

```
df = pd.read_csv('CARD_SUBWAY_MONTH_202102.csv', index_col= False)
df
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204
...	...	...	...	...	...	...
16746	20210228	중앙선	오빈	216	236	20210303
16747	20210228	중앙선	양평	3299	3187	20210303
16748	20210228	중앙선	원덕	322	304	20210303
16749	20210228	중앙선	용문	1899	1716	20210303
16750	20210228	중앙선	지평	30	31	20210303

16751 rows × 6 columns

zhining@naver.com

■ 탐색

[문제]
데이터프레임(df)의 처음 다섯 개의 로우를 출력하세요

```
df.head()
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

[Tip] 데이터프레임의 뒤쪽의 아이템을 확인하는 tail()

```
df.tail(7)
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
16744	20210228	중앙선	국수	616	611	20210303
16745	20210228	중앙선	아신	535	510	20210303
16746	20210228	중앙선	오빈	216	236	20210303
16747	20210228	중앙선	양평	3299	3187	20210303
16748	20210228	중앙선	원덕	322	304	20210303
16749	20210228	중앙선	용문	1899	1716	20210303
16750	20210228	중앙선	지평	30	31	20210303

zhining@naver.com

■ 탐색

[문제]
데이터프레임(df)의 몇 개의 로우와 행으로 이루어져 있는지 구조(shape)를 확인하세요.

```
df.shape
```

```
(16751, 6)
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

■ 탐색

[문제]
데이터프레임(df)의 평균, 표준편차, 사분위수 등 주요 통계 지표를 확인하세요

```
df.describe()
```

	사용일자	승차총승객수	하차총승객수	등록일자
count	1.675100e+04	16751.000000	16751.000000	1.675100e+04
mean	2.021021e+07	8567.344696	8536.474419	2.021023e+07
std	8.083073e+00	9006.651687	9064.054078	2.749315e+01
min	2.021020e+07	1.000000	0.000000	2.021020e+07
25%	2.021021e+07	2652.500000	2594.500000	2.021021e+07
50%	2.021022e+07	5857.000000	5733.000000	2.021022e+07
75%	2.021022e+07	11257.000000	11117.000000	2.021022e+07
max	2.021023e+07	88904.000000	87891.000000	2.021030e+07

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

■ 탐색

[문제]
데이터프레임(df)에 몇 개의 컬럼이 있는지, 각 컬럼에 포함된 데이터들의 타입은 무엇인지 확인해보세요.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16751 entries, 0 to 16750
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   사용일자    16751 non-null  int64
 1   노선명     16751 non-null  object
 2   역명      16751 non-null  object
 3   승차총승객수  16751 non-null  int64
 4   하차총승객수  16751 non-null  int64
 5   등록일자    16751 non-null  int64
dtypes: int64(4), object(2)
memory usage: 785.3+ KB
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

■ 탐색

[문제]
'노선명' 컬럼이 가지고 있는 데이터의 종류를 확인하세요

```
df['노선명'].unique()
```

```
array(['중앙선', '장항선', '일산선', '우이신설선', '안산선', '수인선', '분당선', '과천선',
       '공항철도 1호선', '경춘선', '경인선', '경의선', '경원선', '경부선', '경강선', '9호선2~3단계',
       '9호선', '8호선', '7호선', '6호선', '5호선', '4호선', '3호선', '2호선', '1호선'],
      dtype=object)
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

■ 탐색

[문제]
'노선명' 컬럼의 각 데이터가 몇 개씩 포함되어 있는지 확인하세요.

```
df['노선명'].value_counts()
```

```
5호선          1484
7호선          1428
2호선          1400
경부선          1092
6호선          1051
분당선           967
3호선           937
경원선           821
경의선           737
4호선           728
9호선           700
중앙선           588
경인선           560
경춘선           532
수인선           504
8호선           476
공항철도 1호선       392
안산선           364
9호선2~3단계       364
우이신설선          364
경강선           308
일산선           282
1호선           280
과천선           224
장항선           168
Name: 노선명, dtype: int64
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

▪ 정렬

[문제]
'승차총승객수' 컬럼을 기준으로 데이터를 정렬하세요

```
df.sort_values(by=['승차총승객수'])
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
13752	20210224	3호선	충무로	1	0	20210227
7584	20210213	경원선	창동	1	0	20210216
12626	20210222	3호선	충무로	1	0	20210225
12932	20210222	경원선	창동	1	0	20210225
13043	20210222	분당선	복정	1	0	20210225
...	...	...	...	...	...	...
14384	20210225	2호선	강남	84374	82960	20210228
14314	20210224	2호선	강남	84628	82667	20210227
13705	20210223	2호선	강남	84922	83364	20210226
10857	20210219	2호선	강남	86800	80948	20210222
14984	20210226	2호선	강남	88904	87891	20210301

16751 rows × 6 columns

```
df.sort_values(by=['승차총승객수'], ascending=False)
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
14984	20210226	2호선	강남	88904	87891	20210301
10857	20210219	2호선	강남	86800	80948	20210222
13705	20210223	2호선	강남	84922	83364	20210226
14314	20210224	2호선	강남	84628	82667	20210227
14384	20210225	2호선	강남	84374	82960	20210228
...	...	...	...	...	...	...
14346	20210224	일산선	지축	1	0	20210227
4853	20210209	분당선	복정	1	0	20210212
14425	20210225	3호선	충무로	1	0	20210228
10025	20210217	경의선	김포공항	1	0	20210220
13752	20210224	3호선	충무로	1	0	20210227

16751 rows × 6 columns

zhining@naver.com

■ 선택

[문제]
'노선명' 컬럼을 선택하세요.

```
df['노선명']
```

```
0          중앙선
1          중앙선
2          중앙선
3          중앙선
4          중앙선
         ...
16746      중앙선
16747      중앙선
16748      중앙선
16749      중앙선
16750      중앙선
Name: 노선명, Length: 16751, dtype: object
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

■ 선택

[문제]
데이터프레임의 위에서부터 10개(0번째부터 9번째)의 데이터(로우)를 선택하세요.

```
df[0:10]
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204
5	20210201	중앙선	팔당	687	662	20210204
6	20210201	중앙선	도심	2237	1736	20210204
7	20210201	중앙선	덕소	5139	5082	20210204
8	20210201	중앙선	양정	721	606	20210204
9	20210201	중앙선	도농	9067	8349	20210204

```
df.iloc[1]
```

```
사용일자        20210201
노선명             중앙선
역명              지평
승차총승객수          48
하차총승객수          43
등록일자        20210204
Name: 1, dtype: object
```

zhining@naver.com

■ 선택

- 인덱스로 값을 선택하는 loc[ ], iloc[ ]

```
# 인덱스를 '사용일자' 로 바꾸어 설정
df2 = df.set_index('사용일자')
df2.head(3)
```

사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
20210201	중앙선	용문	1282	1259	20210204
20210201	중앙선	지평	48	43	20210204
20210201	중앙선	아신	415	413	20210204

```
df2.iloc[1]
```

```
노선명              중앙선
역명                지평
승차총승객수          48
하차총승객수          43
등록일자        20210204
Name: 20210201, dtype: object
```

```
df2.loc['20210201']
```

사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
20210201	중앙선	용문	1282	1259	20210204
20210201	중앙선	지평	48	43	20210204
20210201	중앙선	아신	415	413	20210204
20210201	중앙선	양수	1463	1466	20210204
20210201	중앙선	운길산	495	463	20210204
...	...	...	...	...	...
20210201	경원선	도봉	5561	5396	20210204
20210201	1호선	서울역	35821	33113	20210204
20210201	중앙선	신원	166	134	20210204
20210201	8호선	산성	6122	6090	20210204
20210201	중앙선	원덕	273	249	20210204

599 rows × 5 columns

zhining@naver.com

■ 선택

[문제]
데이터프레임에서 '노선명'이 '2호선'인 로우만 선택하세요.

```
df['노선명'] == '2호선'
```

```
0        False
1        False
2        False
3        False
4        False
         ...
16746    False
16747    False
16748    False
16749    False
16750    False
Name: 노선명, Length: 16751, dtype: bool
```

```
df[df['노선명'] == '2호선']
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
532	20210201	2호선	용두(동대문구청)	2365	2430	20210204
533	20210201	2호선	신정네거리	9269	9599	20210204
534	20210201	2호선	양천구청	6318	6738	20210204
535	20210201	2호선	도림천	1434	1469	20210204
536	20210201	2호선	신설동	3046	3110	20210204
...	...	...	...	...	...	...
16207	20210228	2호선	신설동	2806	2830	20210303
16208	20210228	2호선	도림천	480	477	20210303
16209	20210228	2호선	양천구청	3234	3281	20210303
16210	20210228	2호선	신정네거리	4814	4805	20210303
16211	20210228	2호선	용두(동대문구청)	1069	1029	20210303

1400 rows × 6 columns

zhining@naver.com

■ 선택

[문제]
데이터프레임에서 '승차총승객수가'이 50000명 이상인 로우의 '역명'을 확인하세요.

```
df[df['승차총승객수'] >= 50000]
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
550	20210201	2호선	구로디지털단지	57932	58583	20210204
552	20210201	2호선	신림	57124	56022	20210204
560	20210201	2호선	강남	77926	76056	20210204
562	20210201	2호선	선릉	53457	46468	20210204
566	20210201	2호선	잠실(송파구청)	56357	56358	20210204
...	...	...	...	...	...	...
14994	20210226	2호선	구로디지털단지	61720	61649	20210301
15001	20210226	2호선	홍대입구	54628	58375	20210301
15579	20210227	2호선	잠실(송파구청)	56		
15585	20210227	2호선	강남	57		
15602	20210227	2호선	홍대입구	56		

109 rows × 6 columns

```
df[df['승차총승객수'] > 50000]['역명'].unique()
```

```
array(['구로디지털단지', '신림', '강남', '선릉', '잠실(송파구청)', '역삼', '홍대입구', '삼성(무역센터)'],
      dtype=object)
```

zhining@naver.com

■ 삭제

[문제]
데이터프레임에서 0번 로우를 삭제하세요.

```
df.drop(0)
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204
5	20210201	중앙선	팔당	687	662	20210204
...	...	...	...	...	...	...
16746	20210228	중앙선	오빈	216	236	20210303
16747	20210228	중앙선	양평	3299	3187	20210303
16748	20210228	중앙선	원덕	322	304	20210303
16749	20210228	중앙선	용문	1899	1716	20210303
16750	20210228	중앙선	지평	30	31	20210303

16750 rows × 6 columns

[문제]
데이터프레임에서 '등록일자' 컬럼을 삭제하세요

```
df.drop(['등록일자'], axis=1)
```

	사용일자	노선명	역명	승차총승객수	하차총승객수
0	20210201	중앙선	용문	1282	1259
1	20210201	중앙선	지평	48	43
2	20210201	중앙선	아신	415	413
3	20210201	중앙선	양수	1463	1466
4	20210201	중앙선	운길산	495	463
...	...	...	...	...	...
16746	20210228	중앙선	오빈	216	236
16747	20210228	중앙선	양평	3299	3187
16748	20210228	중앙선	원덕	322	304
16749	20210228	중앙선	용문	1899	1716
16750	20210228	중앙선	지평	30	31

16751 rows × 5 columns

zhining@naver.com

■ 연산 – 산술연산

[문제]
'승차총승객수' 컬럼의 모든 값을 2씩 증가시키세요.

```
df['승차총승객수']+2
```

```
0         1284
1           50
2          417
3         1465
4          497
          ...
16746      218
16747     3301
16748      324
16749     1901
16750       32
Name: 승차총승객수, Length: 16751, dtype: int64
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

■ 연산 – 산술연산

[문제]
'승차총승객수'와 '하차총승객수' 컬럼의 차이를 계산한 '승하차총승객수차이' 컬럼을 생성하세요.

```
df['승하차총승객수차이'] = df['승차총승객수'] - df['하차총승객수']
df.head()
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자	승하차총승객수차이
0	20210201	중앙선	용문	1282	1259	20210204	23
1	20210201	중앙선	지평	48	43	20210204	5
2	20210201	중앙선	아신	415	413	20210204	2
3	20210201	중앙선	양수	1463	1466	20210204	-3
4	20210201	중앙선	운길산	495	463	20210204	32

zhining@naver.com

■ 연산 – 산술연산

[문제]
'승하차총승객수차이'의 평균값을 계산하세요

```
df['승하차총승객수차이'].mean()
```

```
30.870276401408873
```

zhining@naver.com

■ 연산 – 샘플링

[문제]
데이터프레임에서 10개의 아이템을 샘플링하세요.

```
#비복원추출
sample_df = df.sample(n=10)
sample_df
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자	승하차총승객수차이
673	20210202	수인선	호구포	3384	3614	20210205	-230
15298	20210226	경부선	의왕	8675	8602	20210301	73
8450	20210215	5호선	종로3가	9968	9839	20210218	129
16742	20210228	중앙선	양수	1859	1758	20210303	101
11224	20210219	분당선	서울숲	9156	9208	20210222	-52
3835	20210207	경부선	송탄	2470	2730	20210210	-260
2456	20210205	일산선	백석	9571	9697	20210208	-126
2711	20210205	9호선2~3단계	석촌	7594	6685	20210208	909
8493	20210215	분당선	보정	2128	1899	20210218	229
16111	20210227	우이신설선	정릉	3249	2748	20210302	501

zhining@naver.com

■ 연산 – 값 치환

[문제]
샘플링된 데이터프레임(sample_df)의 값에서 '노선명' 컬럼의 값이 1호선, 2호선, 3호선 4호선인 값을 영어(line1, line2, line3, line4)로 변경하세요

```
sample_df['노선명'].replace(['1호선', '2호선', '3호선', '4호선'], ['line1', 'line2', 'line3', 'line4'])
```

```
673            수인선
15298          경부선
8450           5호선
16742          중앙선
11224          분당선
3835           경부선
2456           일산선
2711      9호선2~3단계
8493           분당선
16111        우이신설선
Name: 노선명, dtype: object
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자	승하차총승객수차이
673	20210202	수인선	호구포	3384	3614	20210205	-230
15298	20210226	경부선	의왕	8675	8602	20210301	73
8450	20210215	5호선	종로3가	9968	9839	20210218	129
16742	20210228	중앙선	양수	1859	1758	20210303	101
11224	20210219	분당선	서울숲	9156	9208	20210222	-52
3835	20210207	경부선	송탄	2470	2730	20210210	-260
2456	20210205	일산선	백석	9571	9697	20210208	-126
2711	20210205	9호선2~3단계	석촌	7594	6685	20210208	909
8493	20210215	분당선	보정	2128	1899	20210218	229
16111	20210227	우이신설선	정릉	3249	2748	20210302	501

zhining@naver.com

■ 연산 – 함수 적용

[문제] 데이터프레임(df)의 '사용일자1' 컬럼을 새로 생성하고, '사용일자' 컬럼에 있는 값을 datetime64의 타입으로 변환하여 넣으세요.

```
df['사용일자1'] =  df['사용일자'].apply(getDate)
```

```
df.head()
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자	승하차총승객수차이	사용일자1
0	20210201	중앙선	용문	1282	1259	20210204	23	2021-02-01
1	20210201	중앙선	지평	48	43	20210204	5	2021-02-01
2	20210201	중앙선	아신	415	413	20210204	2	2021-02-01
3	20210201	중앙선	양수	1463	1466	20210204	-3	2021-02-01
4	20210201	중앙선	운길산	495	463	20210204	32	2021-02-01

```
df['사용일자1'].astype('datetime64')
```

```
0       2021-02-01
1       2021-02-01
2       2021-02-01
3       2021-02-01
4       2021-02-01
           ...
16746   2021-02-28
16747   2021-02-28
16748   2021-02-28
16749   2021-02-28
16750   2021-02-28
Name: 사용일자1, Length: 16751, dtype: datetime64[ns]
```

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

zhining@naver.com

- 연산 – 함수 적용
  - lambda 식을 활용한 연산

```
df['승차총승객수'].apply(lambda x: x+2)
```

```
0         1284
1           50
2          417
3         1465
4          497
          ...
16746      218
16747     3301
16748      324
16749     1901
16750       32
Name: 승차총승객수, Length: 16751, dtype: int64
```

```
plus2 = lambda x: x+2
df['승차총승객수'].apply(plus2)
```

```
0         1284
1           50
2          417
3         1465
4          497
          ...
16746      218
16747     3301
16748      324
16749     1901
16750       32
Name: 승차총승객수, Length: 16751, dtype: int64
```

zhining@naver.com

■ 연산 – 원핫인코딩

```
dummies = pd.get_dummies(sample_df['노선명'], prefix='노선')
dummies
```

	노선_5호선	노선_9호선2~3단계	노선_경부선	노선_분당선	노선_수인선	노선_우이신설선	노선_일산선	노선_중앙선
673	0	0	0	0	1	0	0	0
15298	0	0	1	0	0	0	0	0
8450	1	0	0	0	0	0	0	0
16742	0	0	0	0	0	0	0	1
11224	0	0	0	1	0	0	0	0
3835	0	0	1	0	0	0	0	0
2456	0	0	0	0	0	0	1	0
2711	0	1	0	0	0	0	0	0
8493	0	0	0	1	0	0	0	0
16111	0	0	0	0	0	1	0	0

zhining@naver.com

■ 연산 – 결측치 처리

```
import numpy as np
data = {'학번':range(2000,2010),
    '국어성적': [85, 95, 75, 70, 100, np.nan, 95, 85, 80, 85],
    '영어성적': [95, 70, 100, 85, 80, 85, 95, 95, np.nan, 70],
    '수학성적': [np.nan, np.nan, np.nan, np.nan, np.nan, np.nan, np.nan, 95, np.nan, 70],   }
df = pd.DataFrame(data)
df
```

	학번	국어성적	영어성적	수학성적
0	2000	85.0	95.0	NaN
1	2001	95.0	70.0	NaN
2	2002	75.0	100.0	NaN
3	2003	70.0	85.0	NaN
4	2004	100.0	80.0	NaN
5	2005	NaN	85.0	NaN
6	2006	95.0	95.0	NaN
7	2007	85.0	95.0	95.0
8	2008	80.0	NaN	NaN
9	2009	85.0	70.0	70.0

zhining@naver.com

■ 연산 – 결측치 처리

[문제] 위에서 생성한 데이터프레임에 결측치가 포함되어 있는지 확인하는 코드를 작성하세요.

```
df.isna()
```

	학번	국어성적	영어성적	수학성적
0	False	False	False	True
1	False	False	False	True
2	False	False	False	True
3	False	False	False	True
4	False	False	False	True
5	False	True	False	True
6	False	False	False	True
7	False	False	False	False
8	False	False	True	True
9	False	False	False	False

```
df.isna().sum()
```

```
학번      0
국어성적    1
영어성적    1
수학성적    8
dtype: int64
```

zhining@naver.com

■ 연산 – 결측치 처리

[문제] 데이터프레임에 포함된 모든 결측치를 0으로 치환하세요.

```
df.fillna(0)
```

	학번	국어성적	영어성적	수학성적
0	2000	85.0	95.0	0.0
1	2001	95.0	70.0	0.0
2	2002	75.0	100.0	0.0
3	2003	70.0	85.0	0.0
4	2004	100.0	80.0	0.0
5	2005	0.0	85.0	0.0
6	2006	95.0	95.0	0.0
7	2007	85.0	95.0	95.0
8	2008	80.0	0.0	0.0
9	2009	85.0	70.0	70.0

```
df['수학성적'].fillna(50, inplace=True)
df
```

	학번	국어성적	영어성적	수학성적
0	2000	85.0	95.0	50.0
1	2001	95.0	70.0	50.0
2	2002	75.0	100.0	50.0
3	2003	70.0	85.0	50.0
4	2004	100.0	80.0	50.0
5	2005	NaN	85.0	50.0
6	2006	95.0	95.0	50.0
7	2007	85.0	95.0	95.0
8	2008	80.0	NaN	50.0
9	2009	85.0	70.0	70.0

zhining@naver.com

■ 연산 – 결측치 처리

[문제] 데이터프레임에 포함된 모든 결측치를 삭제하세요.

```
df.dropna()
```

	학번	국어성적	영어성적	수학성적
0	2000	85.0	95.0	50.0
1	2001	95.0	70.0	50.0
2	2002	75.0	100.0	50.0
3	2003	70.0	85.0	50.0
4	2004	100.0	80.0	50.0
6	2006	95.0	95.0	50.0
7	2007	85.0	95.0	95.0
9	2009	85.0	70.0	70.0

▪ 변형

[문제] '서울시 지하철호선별 역별 승하차 인원 정보'의 데이터프레임을 '노선명'으로 그룹핑하고, 각 필드 의 데이터의 평균값을 표시하도록 코드를 작성하세요.

```
df = pd.read_csv('CARD_SUBWAY_MONTH_202102.csv', index_col= False)
df.groupby(['노선명']).mean()
```

원본

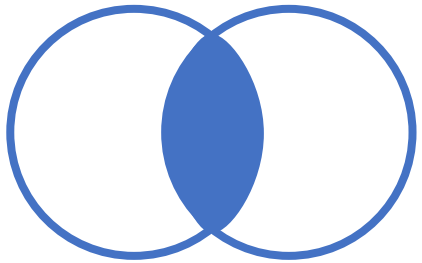
	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
0	20210201	중앙선	용문	1282	1259	20210204
1	20210201	중앙선	지평	48	43	20210204
2	20210201	중앙선	아신	415	413	20210204
3	20210201	중앙선	양수	1463	1466	20210204
4	20210201	중앙선	운길산	495	463	20210204

노선명	사용일자	승차총승객수	하차총승객수	등록일자
1호선	2.021021e+07	16711.396429	16224.032143	2.021023e+07
2호선	2.021021e+07	20258.510000	20406.605714	2.021023e+07
3호선	2.021021e+07	11563.884739	11542.535752	2.021023e+07
4호선	2.021021e+07	13839.026099	13953.564560	2.021023e+07
5호선	2.021021e+07	8774.492588	8727.040431	2.021023e+07
6호선	2.021021e+07	6499.918173	6405.854424	2.021023e+07
7호선	2.021021e+07	10210.968487	10060.233193	2.021023e+07
8호선	2.021021e+07	8356.453782	8425.422269	2.021023e+07
9호선	2.021021e+07	8110.387143	8246.852857	2.021023e+07
9호선2~3단계	2.021021e+07	5238.137363	5156.379121	2.021023e+07
경강선	2.021021e+07	1970.116883	1908.357143	2.021023e+07
경부선	2.021021e+07	8366.500000	8366.444139	2.021023e+07
경원선	2.021021e+07	5051.796590	4940.018270	2.021023e+07
경의선	2.021021e+07	2962.963365	2910.443691	2.021023e+07
경인선	2.021021e+07	10789.323214	10672.037500	2.021023e+07
경춘선	2.021021e+07	1462.635338	1409.640977	2.021023e+07
공항철도 1호선	2.021021e+07	5076.122449	4643.492347	2.021023e+07
과천선	2.021021e+07	8984.544643	8827.571429	2.021023e+07
분당선	2.021021e+07	7849.137539	8079.867632	2.021023e+07
수인선	2.021021e+07	2341.289683	2335.438492	2.021023e+07
안산선	2.021021e+07	6966.684066	6926.851648	2.021023e+07
우이신설선	2.021021e+07	2559.425824	2508.461538	2.021023e+07
일산선	2.021021e+07	8072.929078	7861.592199	2.021023e+07
장항선	2.021021e+07	1495.559524	1447.095238	2.021023e+07
중앙선	2.021021e+07	3235.462585	3173.188776	2.021023e+07

zhining@naver.com

■ 변형

이름	키
A	180
B	160
C	150



이름	성별
A	남자
B	여자
D	남자

이름	키	성별
A	180	남자
B	160	여자

```
df1 = pd.DataFrame({'이름': ['A', 'B', 'D'],
                    '성별':['남자', '여자', '남자']
                   })
df1
```

	이름	성별
0	A	남자
1	B	여자
2	D	남자

```
df2 = pd.DataFrame({'이름': ['A', 'B', 'C'],
                    '키':[180, 160, 150],
                   })
df2
```

	이름	키
0	A	180
1	B	160
2	C	150

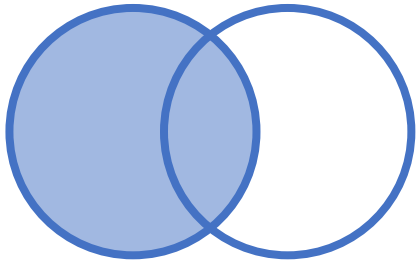
```
pd.merge(df1, df2)
```

	이름	성별	키
0	A	남자	180
1	B	여자	160

zhining@naver.com

■ 변형

이름	키
A	180
B	160
C	150



이름	성별
A	남자
B	여자
D	남자

이름	키	성별
A	180	남자
B	160	여자
C	150	

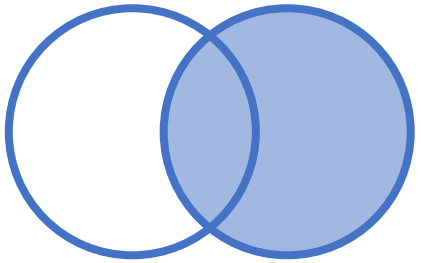
```
pd.merge(df1, df2, how='left')
```

	이름	성별	키
0	A	남자	180.0
1	B	여자	160.0
2	D	남자	NaN

zhining@naver.com

■ 변형

이름	키
A	180
B	160
C	150



이름	성별
A	남자
B	여자
D	남자

이름	키	성별
A	180	남자
B	160	여자
D		남자

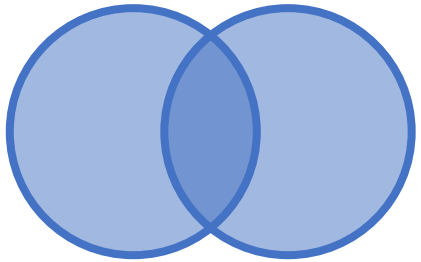
```
pd.merge(df1, df2, how='right')
```

	이름	성별	키
0	A	남자	180
1	B	여자	160
2	C	NaN	150

zhining@naver.com

■ 변형

이름	키
A	180
B	160
C	150



이름	성별
A	남자
B	여자
D	남자

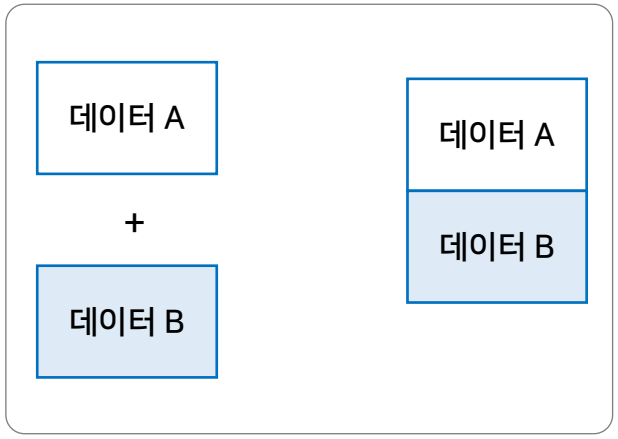
이름	키	성별
A	180	남자
B	160	여자
C	150	
D		남자

```
pd.merge(df1, df2, how='outer')
```

	이름	성별	키
0	A	남자	180.0
1	B	여자	160.0
2	D	남자	NaN
3	C	NaN	150.0

zhining@naver.com

■ 변형



데이터 행(row) 병합

```
df1 = pd.DataFrame({'이름': ['A', 'B', 'C', 'D', 'E'],
                    '키':[180, 160, 150, 170, 170],
                    '나이': [23, 46, 33, 65, 28],
                    '주소': ['서울', '부산', '대전', '원주', '제주']
                   })
df1
```

	이름	키	나이	주소
0	A	180	23	서울
1	B	160	46	부산
2	C	150	33	대전
3	D	170	65	원주
4	E	170	28	제주

```
df2 = pd.DataFrame({'이름': ['F', 'G', 'H', 'I', 'J'],
                    '키':[190, 180, 150, 160, 170],
                    '나이': [22, 18, 87, 45, 74],
                    '주소': ['전주', '서울', '수원', '부산', '인천']
                   })
df2
```

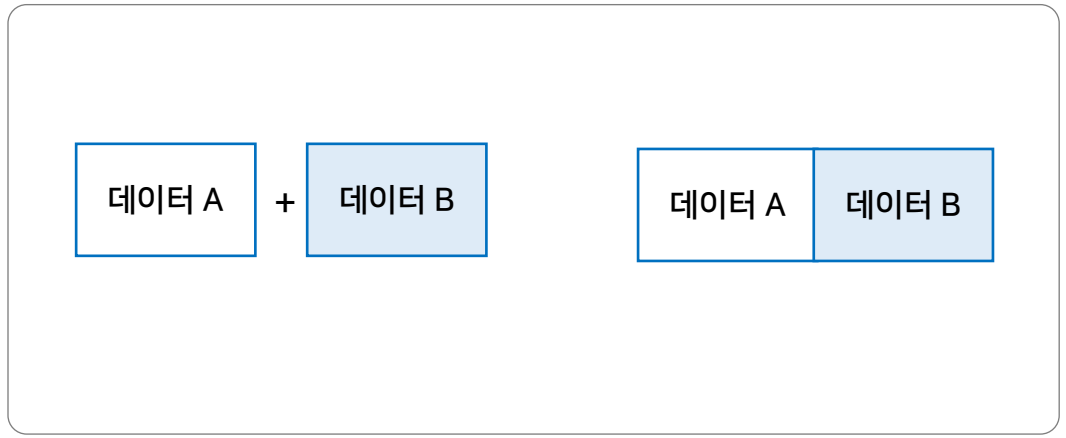
	이름	키	나이	주소
0	F	190	22	전주
1	G	180	18	서울
2	H	150	87	수원
3	I	160	45	부산
4	J	170	74	인천

```
pd.concat([df1, df2])
```

	이름	키	나이	주소
0	A	180	23	서울
1	B	160	46	부산
2	C	150	33	대전
3	D	170	65	원주
4	E	170	28	제주
0	F	190	22	전주
1	G	180	18	서울
2	H	150	87	수원
3	I	160	45	부산
4	J	170	74	인천

zhining@naver.com

■ 변형



데이터 열(column) 병합

```
pd.concat([df1, df2], axis = 1)
```

	이름	키	나이	주소	이름	키	나이	주소
0	A	180	23	서울	F	190	22	전주
1	B	160	46	부산	G	180	18	서울
2	C	150	33	대전	H	150	87	수원
3	D	170	65	원주	I	160	45	부산
4	E	170	28	제주	J	170	74	인천

zhining@naver.com

- 저장

```
df = pd.DataFrame({'이름': ['A', 'B', 'D'],
                   '성별':['남자', '여자', '남자']
                  })
df
```

	이름	성별
0	A	남자
1	B	여자
2	D	남자

```
df.to_csv('dataframe.csv', encoding='euc-kr')
```

zhining@naver.com

Q&A

zhining@naver.com

Appendix

zhining@naver.com

주요 문법(입력 인자는 생략)	설명
np.array()	초기화할 값을 지정하여 배열 생성
np.zeros()	값을 0으로 초기화하여 배열 생성
np.ones()	값을 1로 초기회하여 배열 생성
np.arange()	수의 순차적인 증감을 이용하여 배열 생성
np.rand()	랜덤한 숫자로 배열 생성
reshape()	배열 변형
shape	배열의 형태 확인
dtype()	배열의 데이터 타입 확인
astype()	배열의 데이터 타입 변경
square()	제곱
sqrt()	제곱근(루트)
exp()	지수승
log()	로그
add()	덧셈
sum()	합계
cumsum()	누적합
mean()	평균
var()	편차
std()	표준편차
min()	최소값
max()	최대값
argmin()	최소값의 인덱스
argmax()	최대값의 인덱스
transpose()	축 변형
np.save()	넘파이 형태로 저장(확장자 .npy)

zhining@naver.com

데이터프레임 생성

주요 문법	설명
pd.DataFrame(*data, index, columns…*)	리스트, 딕셔너리 등의 데이터를 데이터프레임 형태로 만들기
pd.read_csv('csv파일명')	표 형태의 파일 읽어오기

데이터프레임 탐색

주요 문법(입력 인자는 생략)	설명
head()	처음 다섯 개의 로우 출력
tail()	마지막 다섯 개의 로우 출력
shape	데이터프레임의 구조 반환
describe()	데이터프레임의 주요 통계 지표(수치형 컬럼)
info()	데이터프레임의 주요 정보
unique()	중복 제거된 유일 값 반환
value_counts()	유일 값의 개수

zhining@naver.com

데이터프레임 주요 연산

주요 문법(입력 인자는 생략)	설명
sort_values()	값 기준 정렬
sort_index()	인덱스 기준 정렬
iloc[n]	인덱스 번호로 선택(n번째 인덱스 로우 반환)
loc[name]	인덱스 이름으로 선택(인덱스 이름이 name인 로우 반환)
drop()	삭제
mean()	평균값 반환
max()	최대값 반환
min()	최소값 반환
sample()	전체 데이터 중 일부를 추출하여 반환
replace()	값 치환
apply()	함수 적용
get_dummies()	원핫인코딩 적용
isna()	결측치 여부 확인

zhining@naver.com

주요 문법(입력 인자는 생략)	설명
fillna()	결측치 치환
dropna()	결측치 삭제
pd.merge()	두 개의 데이터프레임 조인
pd.concat()	두 개의 데이터프레임 이어 붙이기
to_csv()	저장