/* elice */

파이썬으로 시작하는 데이터 분석

NumPy 사용해보기

임원균 선생님

목차

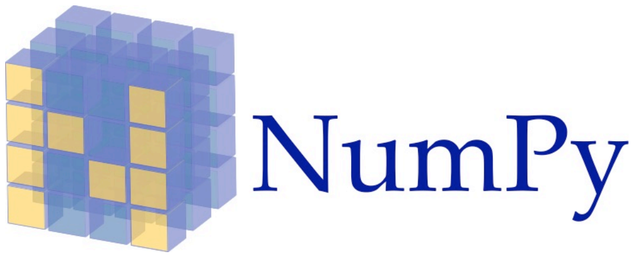
NumPy 소개

NumPy?

: Numerical Python

Python에서 **대규모 다차원 배열**을

다룰 수 있게 도와주는 라이브러리

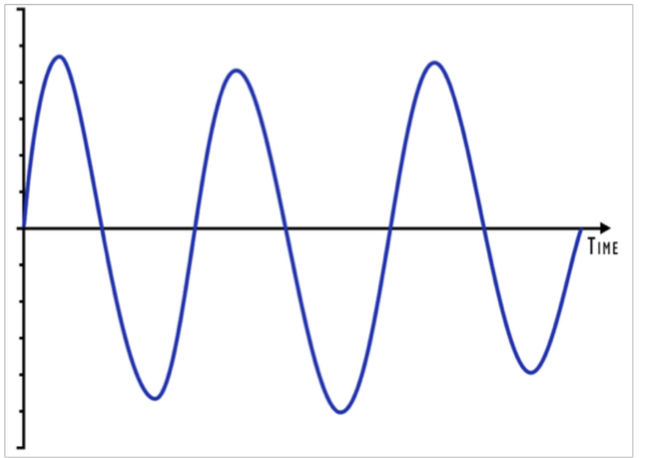


Why?

데이터의 대부분은 숫자 배열로 볼 수 있다

20	54	32	56
63	56	79	43
62	66	77	53
63	6	5	3

23	44	52	56	42	38



Python list?

파이썬 리스트에 비해,

빠른 연산을 지원하고

메모리를 효율적으로 사용

배열 만들기

```
list(range(10))
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]


import numpy as np
np.array([1, 2, 3, 4, 5])
# array([1, 2, 3, 4, 5])
```

배열 만들기

```
np.array([1, 2, 3, 4, 5])
# array([1, 2, 3, 4, 5])
np.array([3, 1.4, 2, 3, 4])
# array([3. , 1.4, 2. , 3. , 4. ])
np.array([[1, 2],
          [3, 4]])
# array([[1, 2],
#        [3, 4]])
np.array([1, 2, 3, 4], dtype='float')
# array([1., 2., 3., 4.])
```

배열 데이터 타입 dtype

```
arr = np.array([1, 2, 3, 4], dtype=float)
arr  # array([1., 2., 3., 4.])
arr.dtype
# dtype('float64')
arr.astype(int)
# array([1, 2, 3, 4])
```

Python List와 다르게 array는 단일타입으로 구성

배열 데이터 타입 dtype

dtype	설명	다양한 표현
int	정수형 타입	i, int_, int32, int64, i8
float	실수형 타입	f, float_, float32, float64, f8
str	문자열 타입	str, U, U32
bool	부울 타입	?, bool_

다양한 배열 만들기

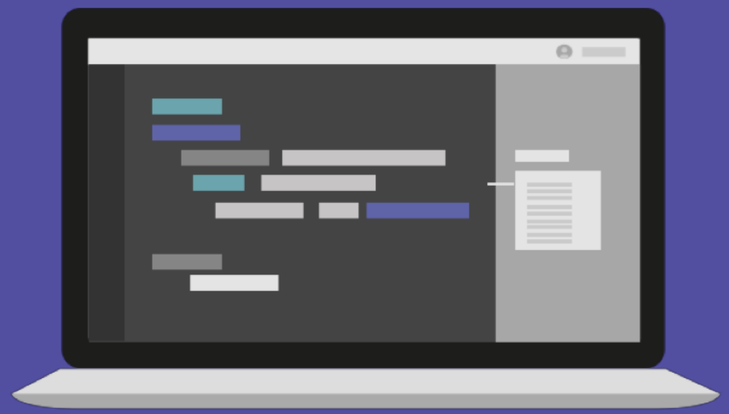
```
np.zeros(10, dtype=int)
# array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

np.ones((3, 5), dtype=float)
# array([[1., 1., 1., 1., 1.],
#        [1., 1., 1., 1., 1.],
#        [1., 1., 1., 1., 1.]])
np.arange(0, 20, 2)
# array([ 0, 2, 4, 6, 8, 10, 12, 14, 16, 18])
np.linspace(0, 1, 5)
# array([0. , 0.25, 0.5 , 0.75, 1. ])
```

난수로 채워진 배열 만들기

```
np.random.random((2, 2))
# array([[0.30986539, 0.85863508],
#        [0.89151021, 0.19304196]])

np.random.normal(0, 1, (2, 2))
# array([[ 0.44050683,  0.04912487],
#        [-1.67023947, -0.70982067]])

np.random.randint(0, 10, (2, 2))
# array([[3, 9],
#         [3, 2]])
```

[실습1]

배열 만들기



배열의 기초

```
x2 = np.random.randint(10, size=(3, 4))
# array([[2, 2, 9, 0],
         [4, 2, 1, 0],
         [1, 8, 7, 3]])


x2.ndim  # 2
x2.shape  # (3, 4)
x2.size  # 12
x2.dtype  # dtype('int64')
```

$$\begin{bmatrix} 2 & 2 & 9 & 0 \\ 4 & 2 & 1 & 0 \\ 1 & 8 & 7 & 3 \end{bmatrix}$$

Indexing / Slicing

찾고 잘라내기

Indexing: 인덱스로 값을 찾아냄


```
x = np.arange(7)
x[3]
# 3
x[7]
# IndexError: index 7 is out of bounds
x[0] = 10
# array([10, 1, 2, 3, 4, 5, 6])
```

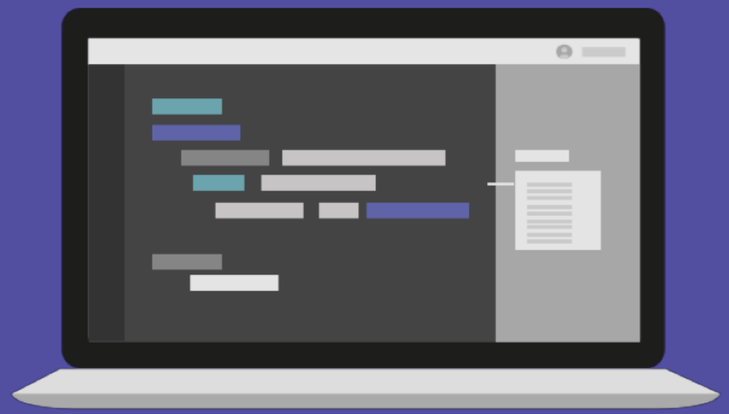
찾고 잘라내기

Slicing: 인덱스 값으로 배열의 부분을 가져옴


```
x = np.arange(7)
x[1:4]
# array([1, 2, 3])
x[1:]
# array([1, 2, 3, 4, 5, 6])
x[:4]
# array([0, 1, 2, 3])
x[::2]
array([0, 2, 4, 6])
```

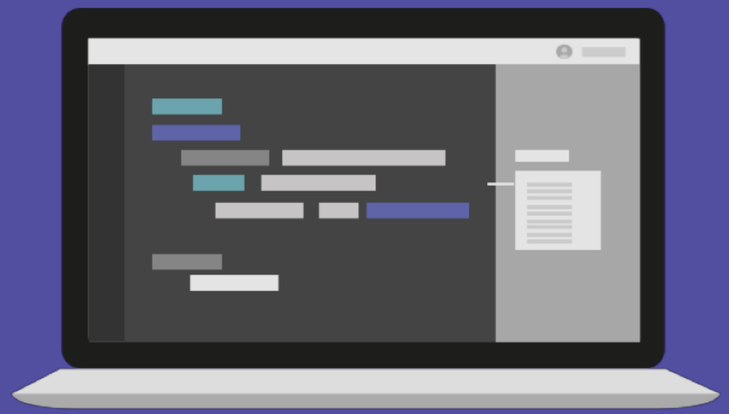
[실습2]

배열의 기초 (1)



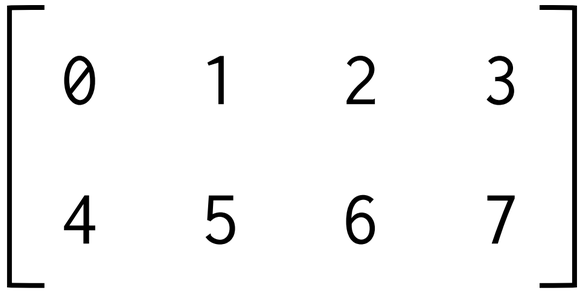
[실습2]

배열의 기초 (2)



모양 바꾸기

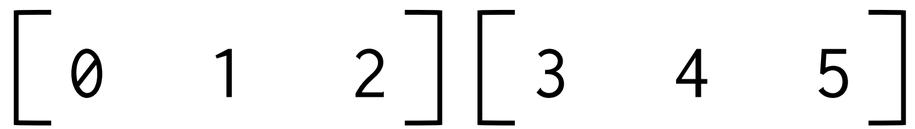
reshape: array의 shape를 변경




```
x = np.arange(8)
x.shape
# (8,)
x2 = x.reshape((2, 4))
# array([[0, 1, 2, 3],
#        [4, 5, 6, 7]])
x2.shape
# (2, 4)
```

이어 붙이고 나누고

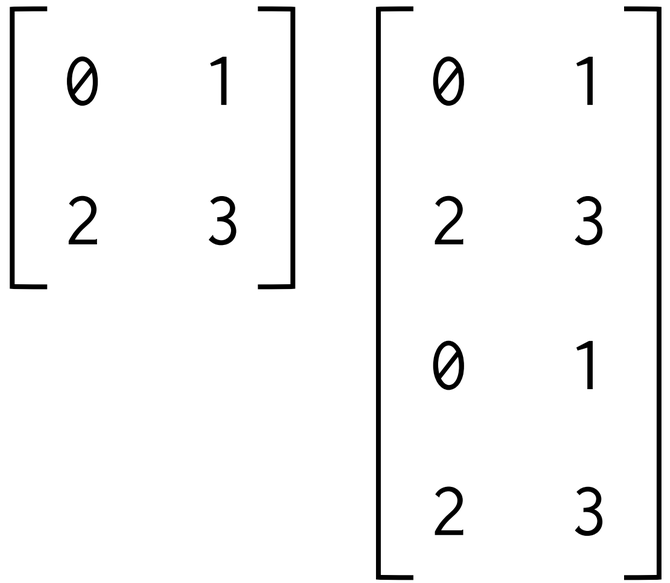
concatenate: array를 이어 붙임




```
x = np.array([0, 1, 2])
y = np.array([3, 4, 5])
np.concatenate([x, y])
# array([0, 1, 2, 3, 4, 5])
```

이어 붙이고 나누고

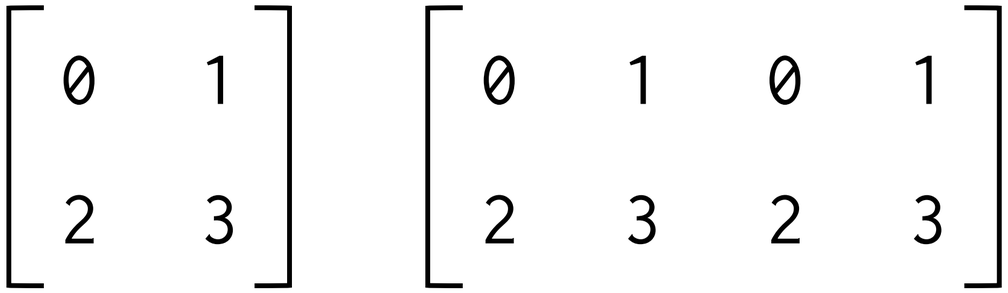
np.concatenate: axis 축을 기준으로 이어붙임




```
matrix = np.arange(4).reshape(2, 2)
np.concatenate([matrix, matrix], axis=0)
```

이어 붙이고 나누고

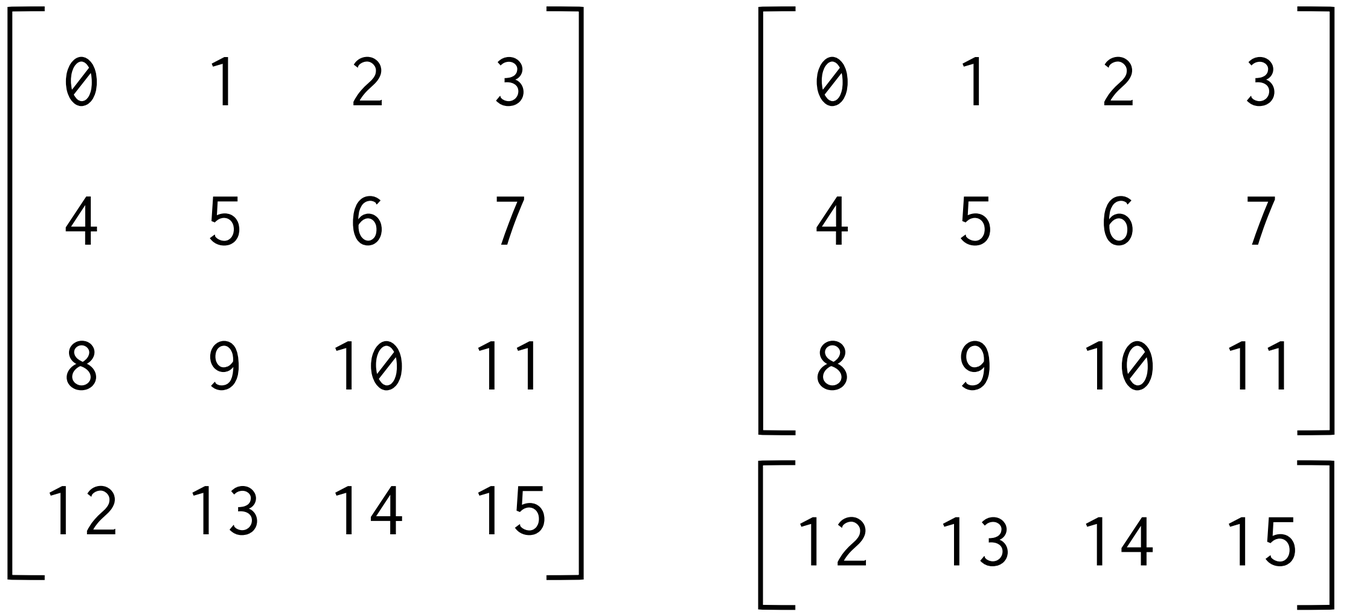
np.concatenate: axis 축을 기준으로 이어붙임




```
matrix = np.arange(4).reshape(2, 2)
np.concatenate([matrix, matrix], axis=1)
```

이어 붙이고 나누고

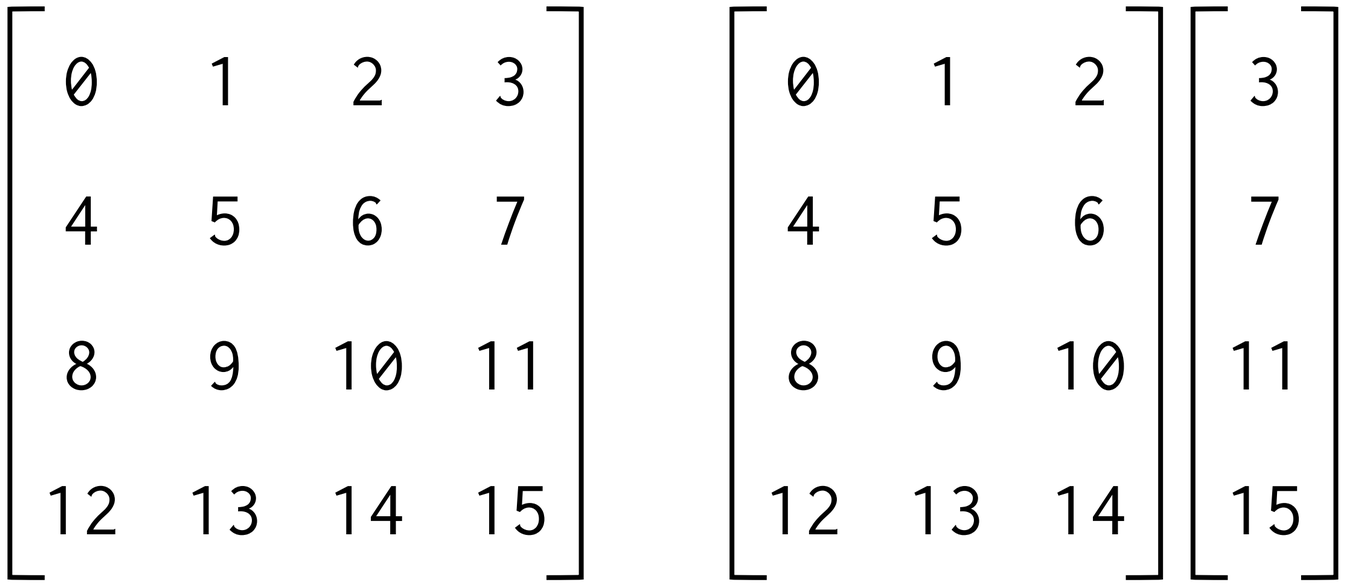
np.split: axis 축을 기준으로 분할




```
matrix = np.arange(16).reshape(4, 4)
upper, lower = np.split(matrix, [3], axis=0)
```

이어 붙이고 나누고

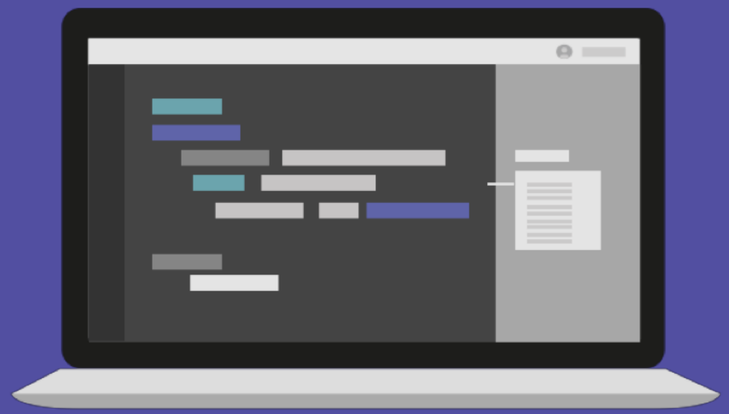
np.split: axis 축을 기준으로 분할




```
matrix = np.arange(16).reshape(4, 4)
left, right = np.split(matrix, [3], axis=1)
```

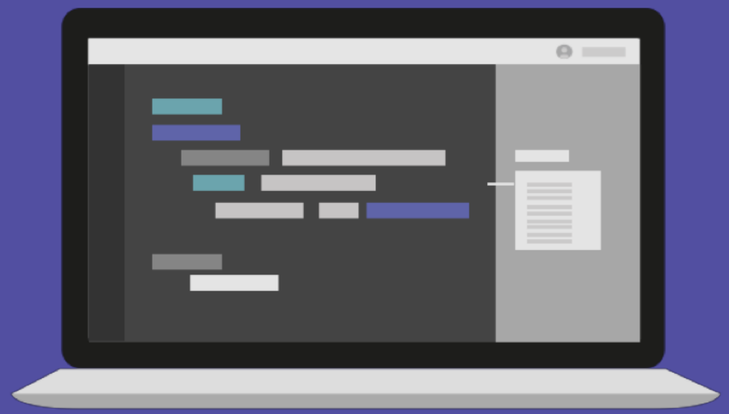
[실습3]

Reshape & 이어붙이고 나누기 (1)



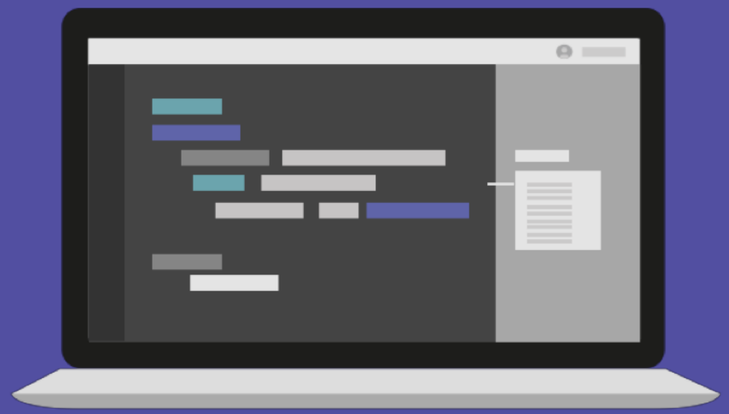
[실습3]

Reshape & 이어붙이고 나누기 (2)



[실습3]

Reshape & 이어붙이고 나누기 (3)



NumPy 연산

루프는 느리다

array의 모든 원소에 5를 더해서 만드는 함수

```
def add_five_to_array(values):
    output = np.empty(len(values))
    for i in range(len(values)):
        output[i] = values[i] + 5
    return output

values = np.random.randint(1, 10, size=5)
add_five_to_array(values)
```

루프는 느리다

만약 array의 크기가 크다면..?

```
big_array = np.random.randint(1, 100, size=10000000)


add_five_to_array(big_array)
# 5.3 s ± 286 ms per loop (mean ± std. dev. of 7 runs,
5 loops each)


big_array + 5
# 33.5 ms ± 1.94 ms per loop (mean ± std. dev. of 7
runs, 5 loops each)
```

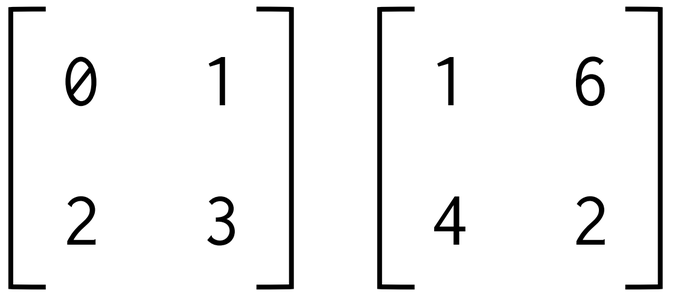
기본 연산

array는 +, - *, / 에 대한 기본 연산을 지원

```
x = np.arange(4)
# array([0, 1, 2, 3])
x + 5
# array([5, 6, 7, 8])
x - 5
# array([-5, -4, -3, -2])
x * 5
# array([ 0,  5, 10, 15])
x / 5
# array([0. , 0.2, 0.4, 0.6])
```

행렬간 연산

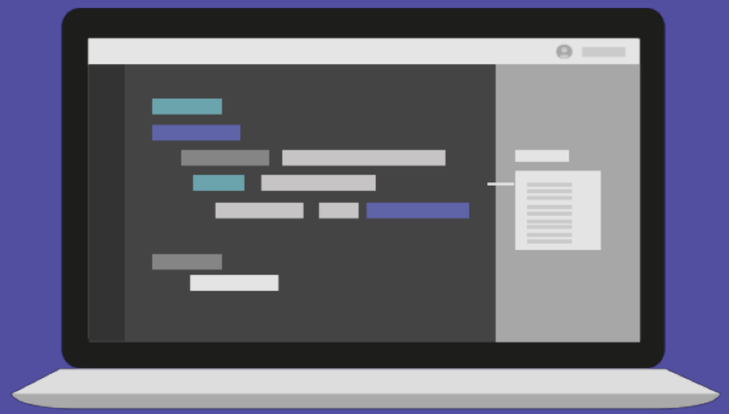
다차원 행렬에서도 적용 가능




```
x = np.arange(4).reshape((2, 2))
y = np.random.randint(10, size=(2, 2))


x + y
# array([[1, 7],
#        [6, 5]])
x - y
# array([[-1, -5],
#        [-2,  1]])
```

[실습4]

기본 연산



브로드캐스팅

Broadcasting: shape이 다른 array끼리 연산

```
matrix + 5
```

2	4	2
6	5	9
9	4	7

5

7	9	7
11	10	14
14	9	12

브로드캐스팅

Broadcasting: shape이 다른 array끼리 연산

```
matrix + np.array([1, 2, 3])
```

2	4	2
6	5	9
9	4	7

1	2	3

3	6	5
7	7	12
10	6	10

브로드캐스팅

Broadcasting: shape이 다른 array끼리 연산

```
np.arange(3).reshape((3,1)) + np.arange(3)
```

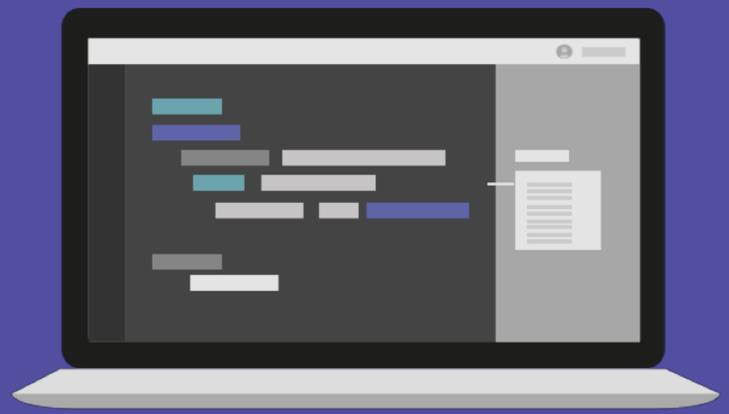
0
1
2

0	1	2

0	1	2
1	2	3
2	3	4

[실습5]

브로드캐스팅



집계함수 & 마스킹 연산

집계함수

집계: 데이터에 대한 요약 통계

```
x = np.arange(8).reshape((2, 4))
np.sum(x)
# 28
np.min(x)
# 0
np.max(x)
# 7
np.mean(x)
# 3.5
```

집계함수

집계: 데이터에 대한 요약 통계

```
x = np.arange(8).reshape((2, 4))

np.sum(x, axis=0)

# array([ 4,  6,  8, 10])

np.sum(x, axis=1)

# array([ 6, 22])
```

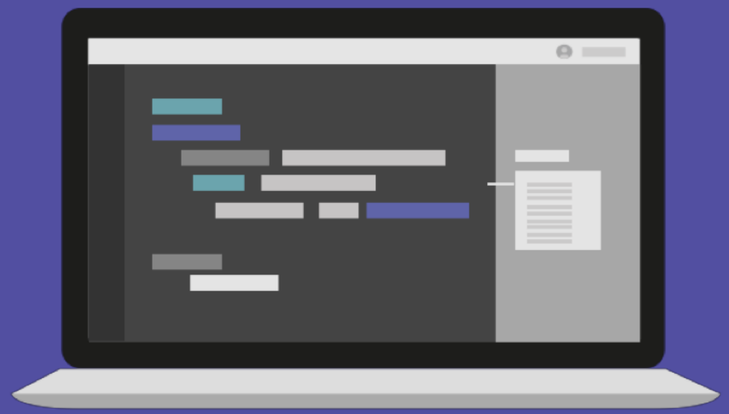
마스킹 연산

마스킹 연산: True, False array를 통해서
특정 값들을 뽑아내는 방법

```
x = np.arange(5)
# array([0, 1, 2, 3, 4])
x < 3
# array([ True,  True,  True, False, False])
x > 5
# array([False, False, False, False, False])
x[x < 3]
# array([0, 1, 2])
```

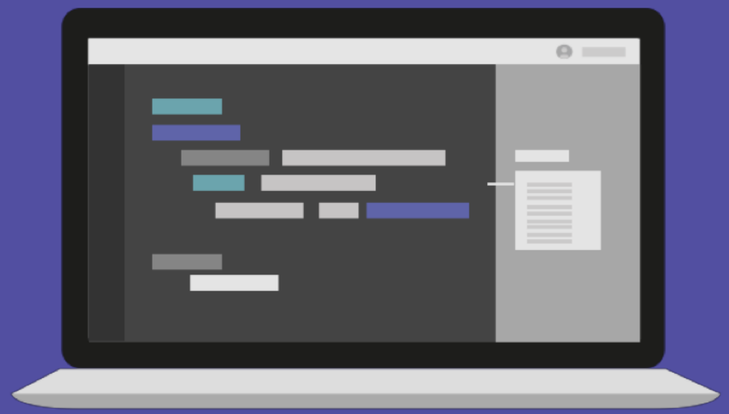
[실습6]

집계함수 & 마스킹연산



[실습7]

양치기 소년의 거짓말 횟수 구하기!



/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice

/* elice */

파이썬으로 시작하는 데이터 분석

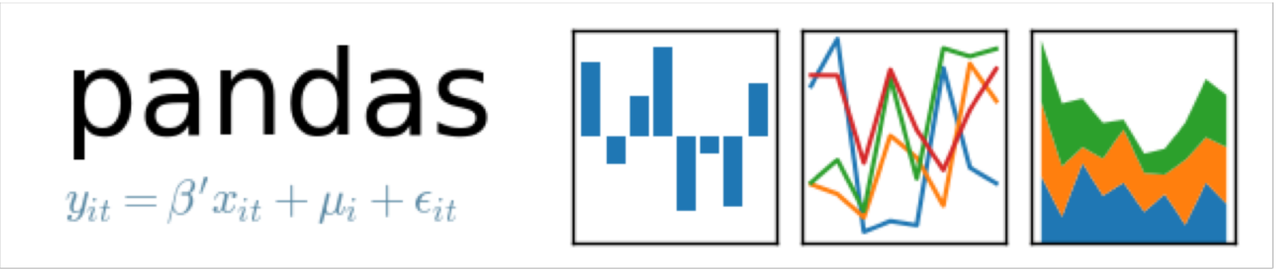
Pandas 기본 알아보기

임원균 선생님

목차

Series 데이터

Pandas



: 파이썬 라이브러리

: 구조화된 데이터를 효과적으로 처리하고 저장

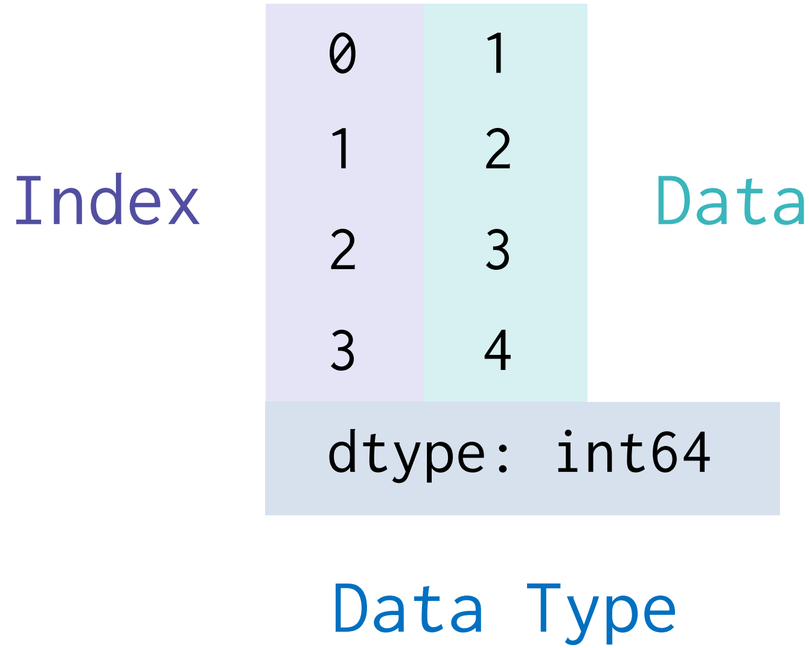
: Array 계산에 특화된 NumPy를 기반으로 설계

Series

numpy array가 보강된 형태

Data와 Index를 가지고 있음

```
import pandas as pd


data = pd.Series([1, 2, 3, 4])

data
```



Series

인덱스를 가지고 있고 인덱스로 접근 가능

```
data = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
data['b']
# 2
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

Series

name 인자로 이름을 지정할 수 있음

```
data = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'], name="Title")
data['c'] = 5
```

```
a    1
b    2
c    5
d    4
Name: Title, dtype: int64
```

Series

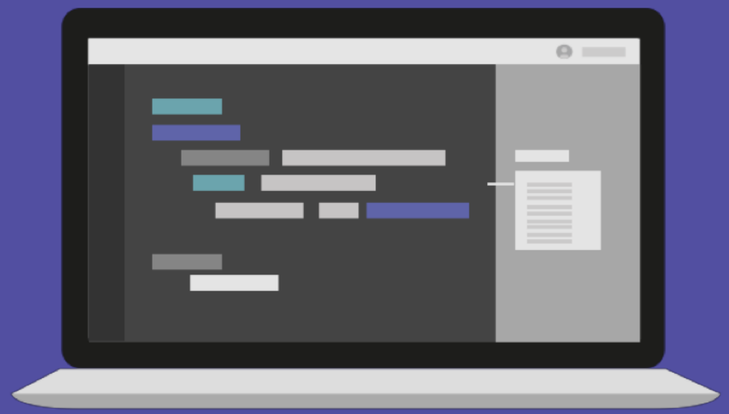
딕셔너리로 변환

```
population_dict = {
    'korea': 5180,
    'japan': 12718,
    'china': 141500,
    'usa': 32676
}
population = pd.Series(population_dict)
```

```
china     141500
japan      12718
korea       5180
usa        32676
dtype: int64
```

[실습1]

Series 데이터



DataFrame

DataFrame

여러 개의 Series가 모여서 행과 열을 이룬 데이터

```
gdp_dict = {
    'korea': 169320000,
    'japan': 516700000,
    'china': 1409250000,
    'usa': 2041280000,
}
gdp = pd.Series(gdp_dict)
country = pd.DataFrame({
    'population': population,
    'gdp': gdp
})
```

	gdp	population
china	1409250000	141500
japan	516700000	12718
korea	169320000	5180
usa	2041280000	32676

DataFrame

딕셔너리로 변환할 수 있음

```
country.index
# Index(['china', 'japan', 'korea', 'usa'],
dtype='object')
country.columns
# Index(['gdp', 'population'], dtype='object')


country['gdp']
type(country['gdp'])
# pandas.core.series.Series
```

	gdp	population
china	1409250000	141500
japan	516700000	12718
korea	169320000	5180
usa	2041280000	32676

```
china     1409250000
japan      516700000
korea      169320000
usa       2041280000
Name: gdp, dtype: int64
```

DataFrame

Series도 numpy array처럼 연산자를 활용

```
gdp_per_capita = country['gdp'] / country['population']
country['gdp per capita'] = gdp_per_capita
```

```
china     9959.363958
japan    40627.457147
korea    32687.258687
usa      62470.314604
dtype: float64
```

	gdp	population	gdp per capita
china	1409250000	141500	9959.363958
japan	516700000	12718	40627.457147
korea	169320000	5180	32687.258687
usa	2041280000	32676	62470.314604

저장과 불러오기

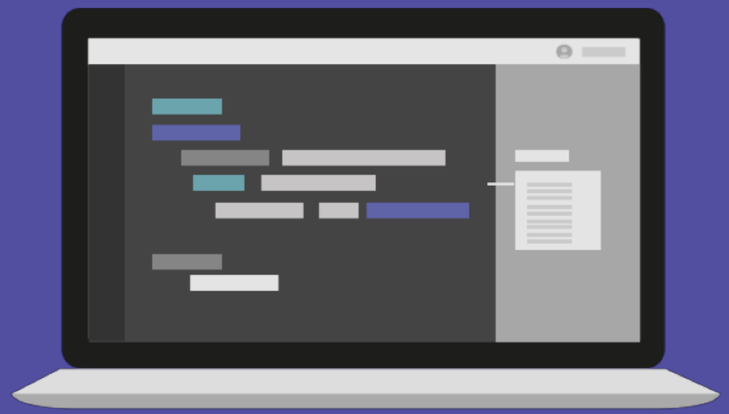
데이터 프레임을 저장

```
country.to_csv(“./country.csv”)
country.to_excel(“country.xlsx”)


country = pd.read_csv(“./country.csv”)
country = pd.read_excel(“country.xlsx”)
```

	A	B	C	D
1		**gdp**	**population**	**gdp per capita**
2	**china**	1409250000	141500	9959.363958
3	**japan**	516700000	12718	40627.45715
4	**korea**	169320000	5180	32687.25869
5	**usa**	2041280000	32676	62470.3146

[실습2]

DataFrame



Indexing / Slicing

Indexing / Slicing

.loc : 명시적인 인덱스를 참조하는 인덱싱/슬라이싱

```
country.loc['china']


country.loc['japan':'korea', :'population']
```

gdp	1.409250e+09
population	1.415000e+05
gdp per capita	9.959364e+03

Name: china, dtype: float64

	gdp	population	gdp per capita
china	1409250000	141500	9959.363958
japan	516700000	12718	40627.457147
korea	169320000	5180	32687.258687
usa	2041280000	32676	62470.314604

	gdp	population
japan	516700000	12718
korea	169320000	5180

Indexing / Slicing

.iloc : 파이썬 스타일 정수 인덱스 인덱싱/슬라이싱

```
country.iloc[0]


country.iloc[1:3, :2]
```

gdp	1.409250e+09
population	1.415000e+05
gdp per capita	9.959364e+03

Name: china, dtype: float64

	gdp	population	gdp per capita
china	1409250000	141500	9959.363958
japan	516700000	12718	40627.457147
korea	169320000	5180	32687.258687
usa	2041280000	32676	62470.314604

	gdp	population
japan	516700000	12718
korea	169320000	5180

DataFrame 새 데이터 추가/수정

리스트로 추가

or 딕셔너리로 추가

```
dataframe = pd.DataFrame(columns=['이름','나이','주소'])
dataframe.loc[0] = ['임원균', '26', '서울']
dataframe.loc[1] = {'이름':'철수', '나이':'25', '주소':'인천'}


dataframe.loc[1, '이름'] = '영희'
```

	이름	나이	주소
0	임원균	26	서울
1	철수	25	인천

	이름	나이	주소
0	임원균	26	서울
1	영희	25	인천

DataFrame 새 컬럼 추가

새로운 컬럼 추가

```
dataframe['전화번호'] = np.nan
dataframe.loc[0, '전화번호'] = '01012341234'


len(dataframe)
# 2
```

	이름	나이	주소	전화번호
0	임원균	26	서울	NaN
1	영희	25	인천	NaN

	이름	나이	주소	전화번호
0	임원균	26	서울	01012341234
1	영희	25	인천	NaN

컬럼 선택하기

컬럼 이름이 하나만 있다면 Series

리스트로 들어가 있다면 DataFrame

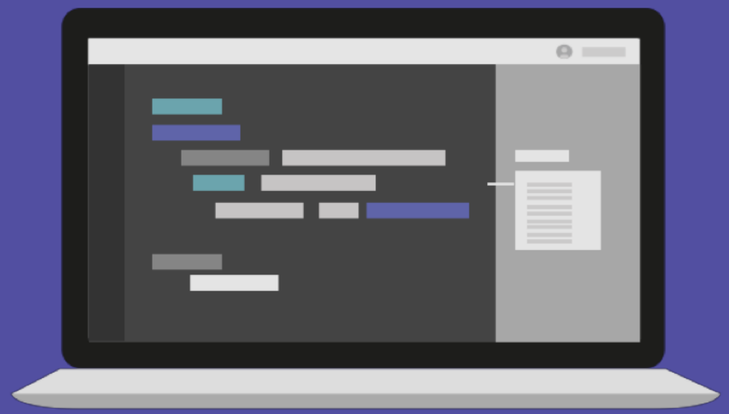
```
dataframe["이름"]
dataframe[["이름", "주소", "나이"]]
```

```
0    임원균
1    영희
Name: 이름, dtype: object
```

	이름	주소	나이
0	임원균	서울	26
1	영희	인천	25

[실습3]

Indexing & Slicing



Pandas 연산과 함수

누락된 데이터 체크

튜토리얼의 데이터와 다르게

현실 데이터는 일부 누락되어 있는 형태가 많음

```
dataframe.isnull()

dataframe.notnull()
```

	이름	나이	주소	전화번호
0	False	False	False	False
1	False	False	False	True

	이름	나이	주소	전화번호
0	True	True	True	True
1	True	True	True	False

누락된 데이터 체크

튜토리얼의 데이터와 다르게

현실 데이터는 일부 누락되어 있는 형태가 많음

```
dataframe.dropna()
dataframe['전화번호'] = dataframe['전화번호'].fillna('전화번호 없음')
```

	이름	나이	주소	전화번호
0	임원균	26	서울	01012341234

	이름	나이	주소	전화번호
0	임원균	26	서울	01012341234
1	영희	25	인천	전화번호 없음

Series 연산

numpy array에서 사용했던 Series연산자들을
동일하게 활용할 수 있음

```
A = pd.Series([2, 4, 6], index=[0, 1, 2])
B = pd.Series([1, 3, 5], index=[1, 2, 3])
A + B
A.add(B, fill_value=0)
```

```
0    2
1    4
2    6
dtype: int64
```

```
1    1
2    3
3    5
dtype: int64
```

```
0    NaN
1    5.0
2    9.0
3    NaN
dtype: float64
```

```
0    2.0
1    5.0
2    9.0
3    5.0
dtype: float64
```

DataFrame 연산

add (+), sub (-), mul (*), div (/)

```
A = pd.DataFrame(np.random.randint(0, 10, (2, 2)), columns=list("AB"))
B = pd.DataFrame(np.random.randint(0, 10, (3, 3)), columns=list("BAC"))
A + B
A.add(B, fill_value=0)
```

	A	B
0	2	4
1	0	1

	B	A	C
0	0	2	3
1	0	4	1
2	6	0	4

	A	B	C
0	4.0	4.0	NaN
1	4.0	1.0	NaN
2	NaN	NaN	NaN

	A	B	C
0	4.0	4.0	3.0
1	4.0	1.0	1.0
2	0.0	6.0	4.0

집계함수

numpy array에서 사용했던 sum, mean 등의 집계함수를 동일하게 사용할 수 있음

```
data = {
    'A': [ i+5 for i in range(3) ],
    'B': [ i**2 for i in range(3) ]
}
df = pd.DataFrame(data)
df['A'].sum() # 18
df.sum()
df.mean()
```

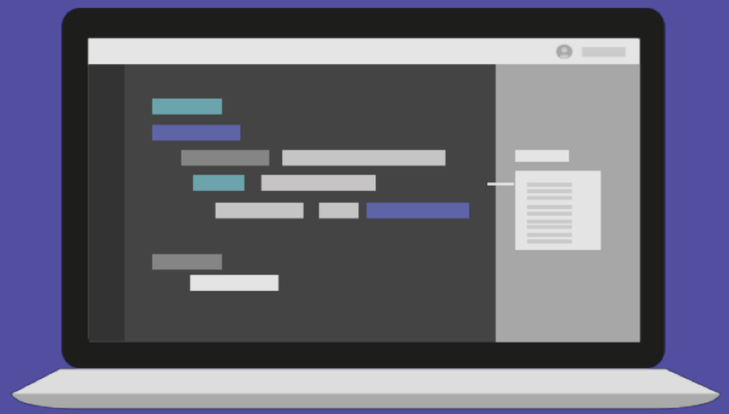
	A	B
0	5	0
1	6	1
2	7	4

```
A    18
B     5
dtype: int64
```

```
A    6.000000
B    1.666667
dtype: float64
```

[실습4]

Pandas 연산과 함수



Dataframe 정렬하기

값으로 정렬하기

`sort_values()`

```
df = pd.DataFrame({
    'col1' : [2, 1, 9, 8, 7, 4],
    'col2' : ['A', 'A', 'B', np.nan, 'D', 'C'],
    'col3': [0, 1, 9, 4, 2, 3],
})
```

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

값으로 정렬하기

```
df.sort_values('col1')
```

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

	col1	col2	col3
1	1	A	1
0	2	A	0
5	4	C	3
4	7	D	2
3	8	NaN	4
2	9	B	9

값으로 정렬하기

```
df.sort_values('col1', ascending=False)
```

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

	col1	col2	col3
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3
0	2	A	0
1	1	A	1

값으로 정렬하기

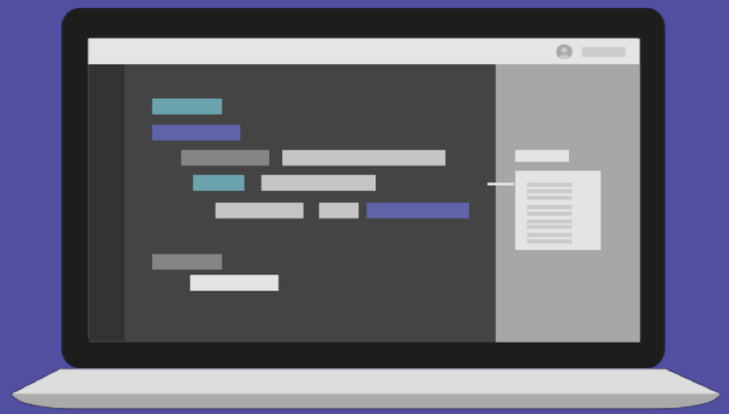
```
df.sort_values(['col2', 'col1'])
```

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

	col1	col2	col3
1	1	A	1
0	2	A	0
2	9	B	9
5	4	C	3
4	7	D	2
3	8	NaN	4

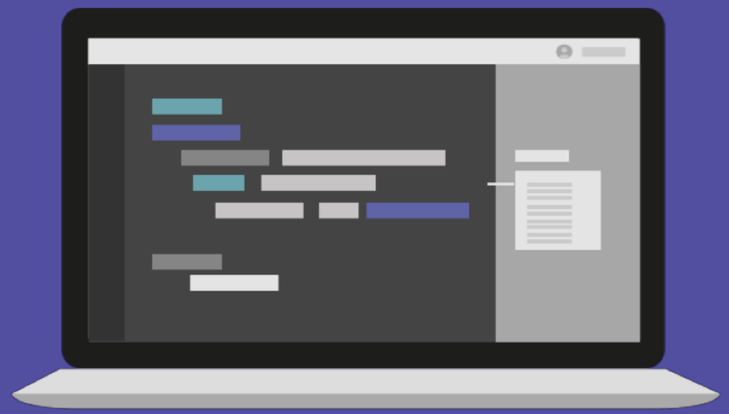
[실습5]

Dataframe 정렬하기



[실습6]

잭이 심은 콩나무 데이터 정렬하기



/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice

/* elice */

파이썬으로 시작하는 데이터 분석

Pandas 심화 알아보기

임원균 선생님

목차

조건으로 검색하기

조건으로 검색하기

numpy array와 마찬가지로 masking 연산이 가능하다

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randint(5, 2), columns=["A", "B"])
df[“A”] < 0.5
```

	A	B
0	0.416760	0.417993
1	0.417333	0.010951
2	0.490884	0.335433
3	0.942838	0.114225
4	0.909844	0.214219

```
0     True
1     True
2     True
3    False
4    False
Name: A, dtype: bool
```

조건으로 검색하기

조건에 맞는 DataFrame row를 추출 가능하다

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randint(5, 2), columns=["A", "B"])
df[(df["A"] < 0.5) & (df["B"] > 0.3)]
df.query("A < 0.5 and B > 0.3")
```

```
0     True
1     True
2     True
3    False
4    False
Name: A, dtype: bool
```

```
0     True
1    False
2     True
3    False
4    False
Name: B, dtype: bool
```

	A	B
0	0.416760	0.417993
2	0.490884	0.335433

조건으로 검색하기

문자열이라면 다른 방식으로도 조건 검색이 가능하다

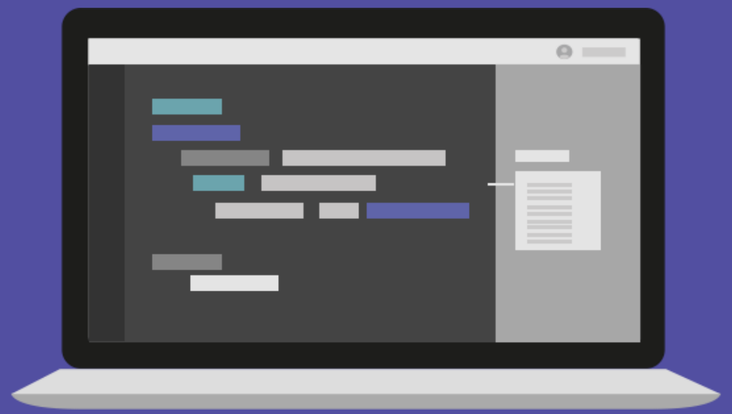
```
df["Animal"].str.contains("Cat")

df.Animal.str.match("Cat")
```

	Animal	Name
0	Dog	Happy
1	Cat	Sam
2	Cat	Toby
3	Pig	Mini
4	Cat	Rocky

```
0    False
1     True
2     True
3    False
4     True
Name: Animal, dtype: bool
```

[실습1]

조건으로 검색하기



함수로 데이터 처리하기

함수로 데이터 처리하기

apply를 통해서 함수로 데이터를 다룰 수 있다

```
df = pd.DataFrame(np.arange(5), columns=["Num"])
def square(x):
    return x**2
df["Num"].apply(square)
df["Square"] = df.Num.apply(lambda x: x ** 2)
```

	Num	Square
0	0	0
1	1	1
2	2	4
3	3	9
4	4	16

```
0     0
1     1
2     4
3     9
4    16
Name: Num, dtype: int64
```

함수로 데이터 처리하기

apply를 통해서 함수로 데이터를 다룰 수 있다

```
df = pd.DataFrame(columns=["phone"])
df.loc[0] = "010-1234-1235"
df.loc[1] = "공일공-일이삼사-1235"
df.loc[2] = "010.1234.일이삼오"
df.loc[3] = "공1공-1234.1이3오"
df["preprocess_phone"] = ''
```

	phone	preprocess_phone
0	010-1234-1235	
1	공일공-일이삼사-1235	
2	010.1234.일이삼오	
3	공1공-1234.1이3오	

함수로 데이터 처리하기

```
def get_preprocess_phone(phone):
    mapping_dict = {
        "공": "0",
        "일": "1",
        "이": "2",
        "삼": "3",
        "사": "4",
        "오": "5",
         "-": "",
        ".": "",
    }
    for key, value in mapping_dict.items():
        phone = phone.replace(key, value)
    return phone
df["preprocess_phone"] = df["phone"].apply(
      get_preprocessed_phonenumber)
```

	phone	preprocess_phone
0	010-1234-1235	01012341235
1	공일공-일이삼사-1235	01012341235
2	010.1234.일이삼오	01012341235
3	공1공-1234.1이3오	01012341235

함수로 데이터 처리하기

replace: apply 기능에서 데이터 값만 대체 하고 싶을때

```
df.Sex.replace({"Male": 0, "Female": 1})
df.Sex.replace({"Male": 0, "Female": 1}, inplace=True)
```

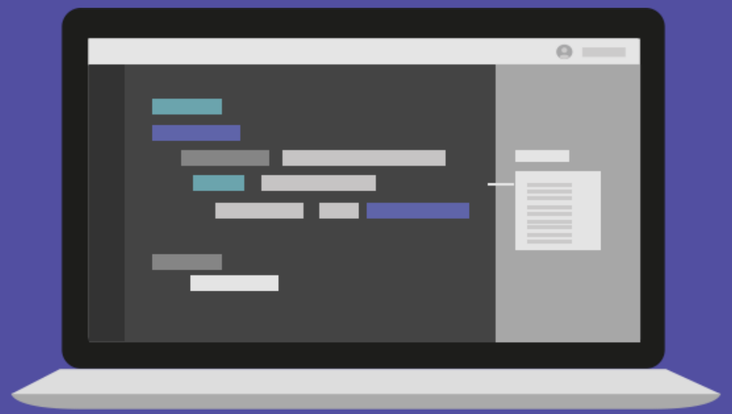
	Sex
0	Male
1	Male
2	Female
3	Female
4	Male

```
0    0
1    0
2    1
3    1
4    0
Name: Sex, dtype: int64
```

	Sex
0	0
1	0
2	1
3	1
4	0

[실습2]

함수로 데이터 처리하기



그룹으로 묶기

그룹으로 묶기

간단한 집계를 넘어서서 조건부로 집계하고 싶은 경우

```
df = pd.DataFrame({'key': ['A', 'B', 'C', 'A', 'B', 'C'],
  'data1':[1, 2, 3, 1, 2, 3], 'data2' : np.random.randint(0, 6, 6)})
df.groupby('key')
# <pandas.core.groupby.groupby.DataFrameGroupBy object at 0x10e3588>
df.groupby('key').sum()
df.groupby(['key','data1']).sum()
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

	data
key	
A	3
B	5
C	7

		data2
key	**data1**	
A	**0**	4
	3	0
B	**1**	4
	4	6
C	**2**	6
	5	1

aggregate

groupby를 통해서 집계를 한번에 계산하는 방법

```
df.groupby('key').aggregate(['min', np.median, max])
df.groupby('key').aggregate({'data1': 'min', 'data2': np.sum})
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

	data1			data2		
	min	median	max	min	median	max
key						
A	0	1.5	3	0	2.0	4
B	1	2.5	4	4	5.0	6
C	2	3.5	5	1	3.5	6

	data1	data2
key		
A	0	4
B	1	10
C	2	7

filter

groupby를 통해서 그룹 속성을 기준으로 데이터 필터링

```
def filter_by_mean(x):
    return x['data2'].mean() > 3
df.groupby('key').mean()
df.groupby('key').filter(filter_by_mean)
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

key	data1	data2
A	1.5	2.0
B	2.5	5.0
C	3.5	3.5

	data1	data2	key
1	1	4	B
2	2	6	C
4	4	6	B
5	5	1	C

apply

groupby를 통해서 묶인 데이터에 함수 적용

```
df.groupby('key').apply(lambda x: x.max() - x.min())
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

key	data1	data2
A	3	4
B	3	2
C	3	5

get_group

groupby로 묶인 데이터에서 key값으로 데이터를 가져올 수 있다

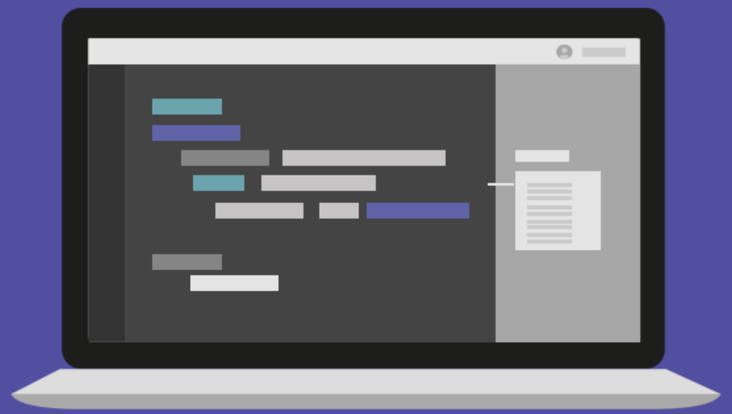
```
df = pd.read_csv("./univ.csv")
df.head()
df.groupby("시도").get_group("충남")
len(df.groupby("시도").get_group("충남"))
# 94
```

	시도	학교명
0	충남	충남도립청양대학
1	경기	한국복지대학교
2	경북	가톨릭상지대학교
3	전북	군산간호대학교
4	경남	거제대학교

	시도	학교명
0	충남	충남도립청양대학
44	충남	신성대학교
60	충남	백석문화대학교
67	충남	혜전대학교
92	충남	아주자동차대학
112	충남	천안연암대학

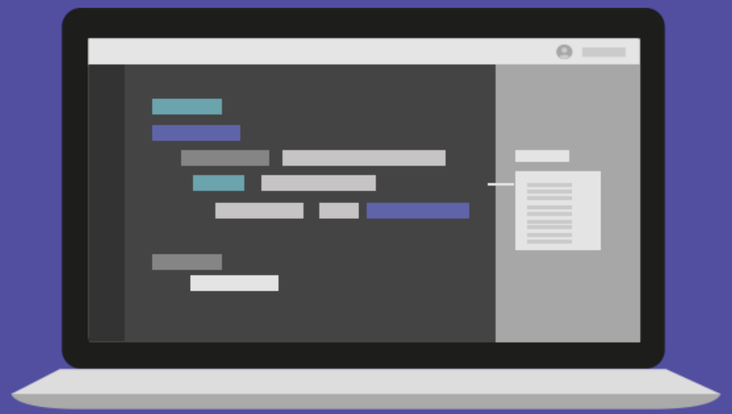
[실습3]

그룹으로 묶기 (1)

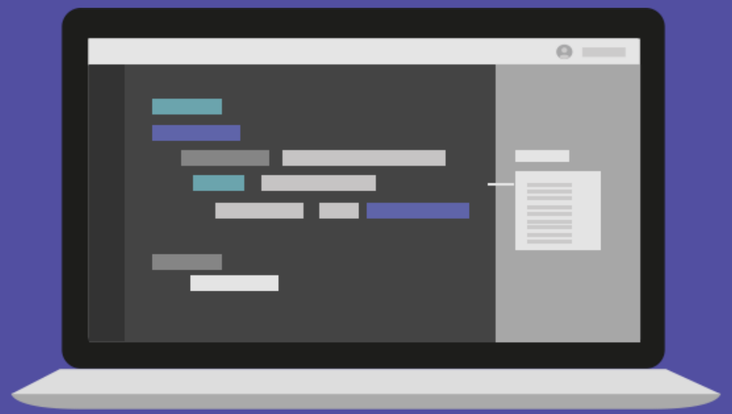


[실습3]

그룹으로 묶기 (2)



[실습3]
그룹으로 묶기 (3)



MultiIndex & pivot_table

MultiIndex

인덱스를 계층적으로 만들 수 있다

```
df = pd.DataFrame(
    np.random.randn(4, 2),
    index=[['A', 'A', 'B', 'B'], [1, 2, 1, 2]],
    columns=['data1', 'data2']
)
```

		data1	data2
A	**1**	1.374474	0.883503
	2	1.471934	-0.004420
B	**1**	0.749019	1.263473
	2	-1.302791	-0.969855

MultiIndex

열 인덱스도 계층적으로 만들 수 있다

```
df = pd.DataFrame(
    np.random.randn(4, 4),
    columns=[["A", "A", "B", "B"], ["1", "2", "1", "2"]]
)
```

	A		B	
	1	2	1	2
0	0.779620	0.089044	1.620036	-0.619421
1	-1.135626	0.219745	1.092230	-0.981231
2	0.181091	-1.089722	-0.323383	-0.586702
3	-0.171893	1.688616	-0.861637	0.156536

MultiIndex

다중 인덱스 컬럼의 경우 인덱싱은 계층적으로 한다

인덱스 탐색의 경우에는 loc, iloc를 사용가능하다

```
df[“A”]
df[“A”][“1”]
```

	A		B	
	1	2	1	2
0	0.779620	0.089044	1.620036	-0.619421
1	-1.135626	0.219745	1.092230	-0.981231
2	0.181091	-1.089722	-0.323383	-0.586702
3	-0.171893	1.688616	-0.861637	0.156536

	1	2
0	1.424545	0.810491
1	0.463098	-0.304480
2	-0.561584	-0.902295
3	-0.219135	0.235004

pivot_table

데이터에서 필요한 자료만 뽑아서 새롭게 요약,

분석 할 수 있는 기능 엑셀에서의 피봇 테이블과 같다

- Index : 행 인덱스로 들어갈 key
- Column : 열 인덱스로 라벨링될 값
- Value : 분석할 데이터

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.0	1	0	7.2500	S	Third	man
1	1	1	female	38.0	1	0	71.2833	C	First	woman
2	1	3	female	26.0	0	0	7.9250	S	Third	woman
3	1	1	female	35.0	1	0	53.1000	S	First	woman
4	0	3	male	35.0	0	0	8.0500	S	Third	man

pivot_table

타이타닉 데이터에서 성별과 좌석별 생존률 구하기

```
df.pivot_table(
    index='sex', columns='class', values='survived’,
    aggfunc=np.mean
)
```

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

pivot_table

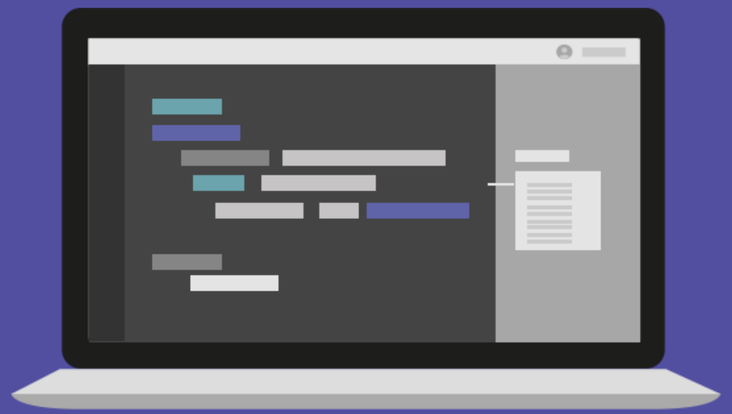
```
df.pivot_table(
    index="월별", columns='내역', values=["수입", '지출'])
```

	월별	내역	지출	수입
0	201805	관리비	200000	0
1	201805	교통비	50000	0
2	201805	월급	0	400000
3	201806	관리비	300000	0
4	201806	교통비	100000	0
5	201806	월급	0	500000
6	201807	관리비	250000	0
7	201807	교통비	150000	0
8	201807	월급	0	600000

	수입			지출		
내역	관리비	교통비	월급	관리비	교통비	월급
월별						
201805	0	0	400000	200000	50000	0
201806	0	0	500000	300000	100000	0
201807	0	0	600000	250000	150000	0

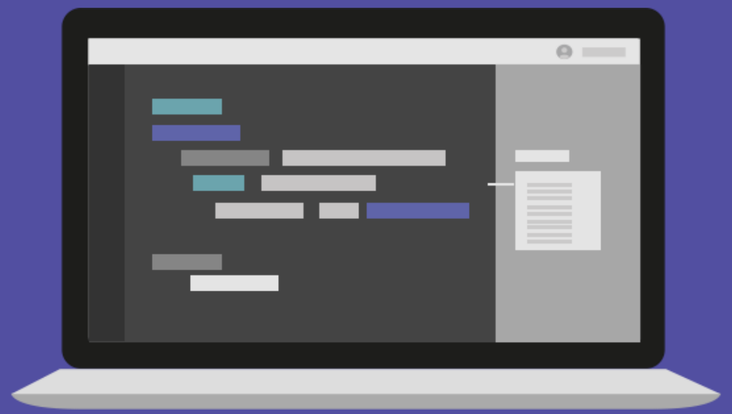
[실습4]

MultiIndex & pivot_table



[실습5]

피리 부는 사나이를 따라가는 아이들



/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice

/* elice */

파이썬으로 시작하는 데이터 분석

Matplotlib 데이터 시각화

임원균 선생님

목차

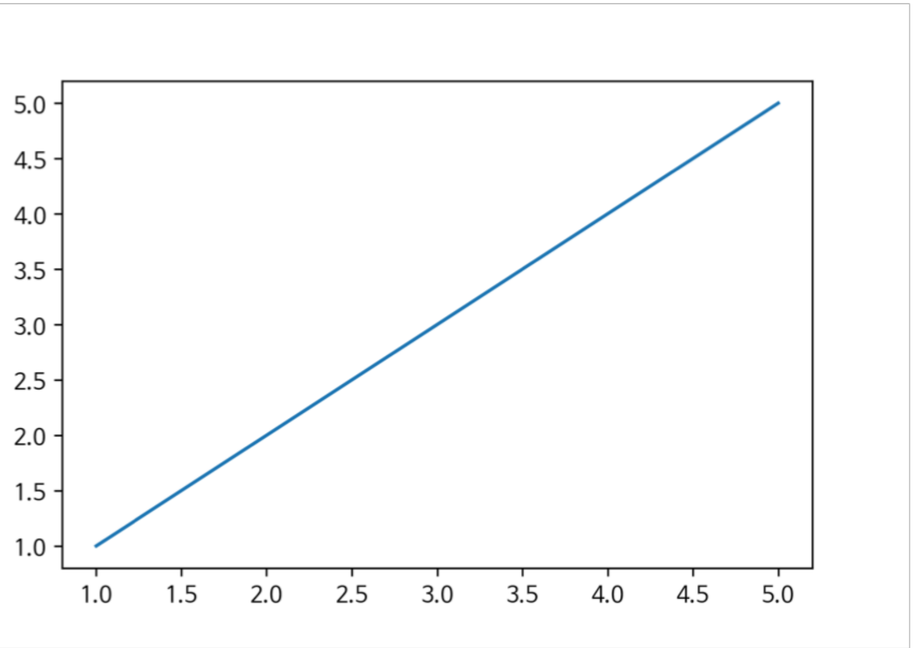
Matplotlib 소개

Matplotlib

파이썬에서 데이터를 그래프나 차트로
시각화할 수 있는 라이브러리

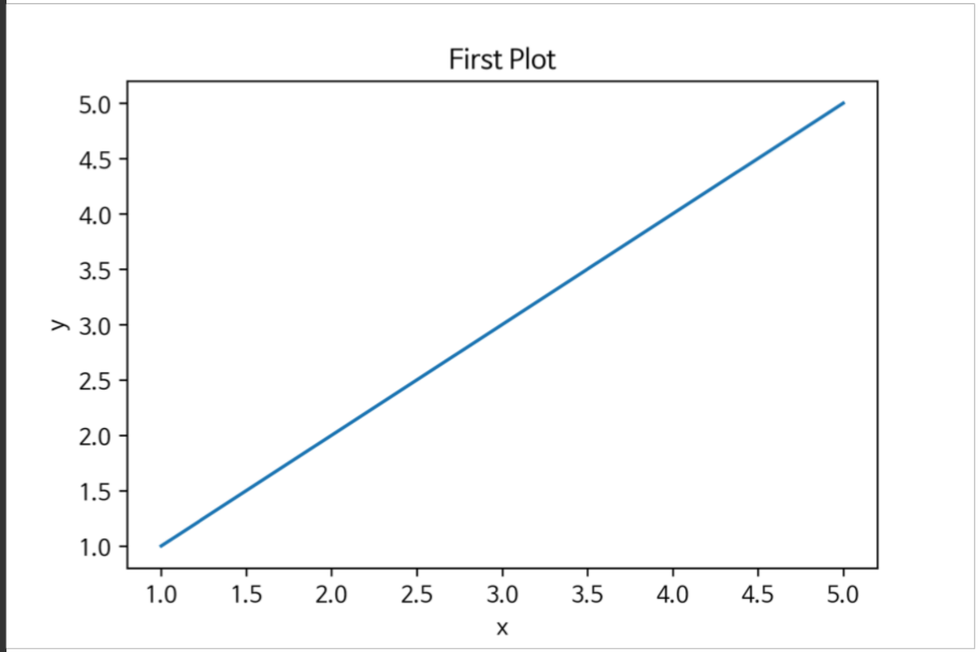
그래프 그려보기

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



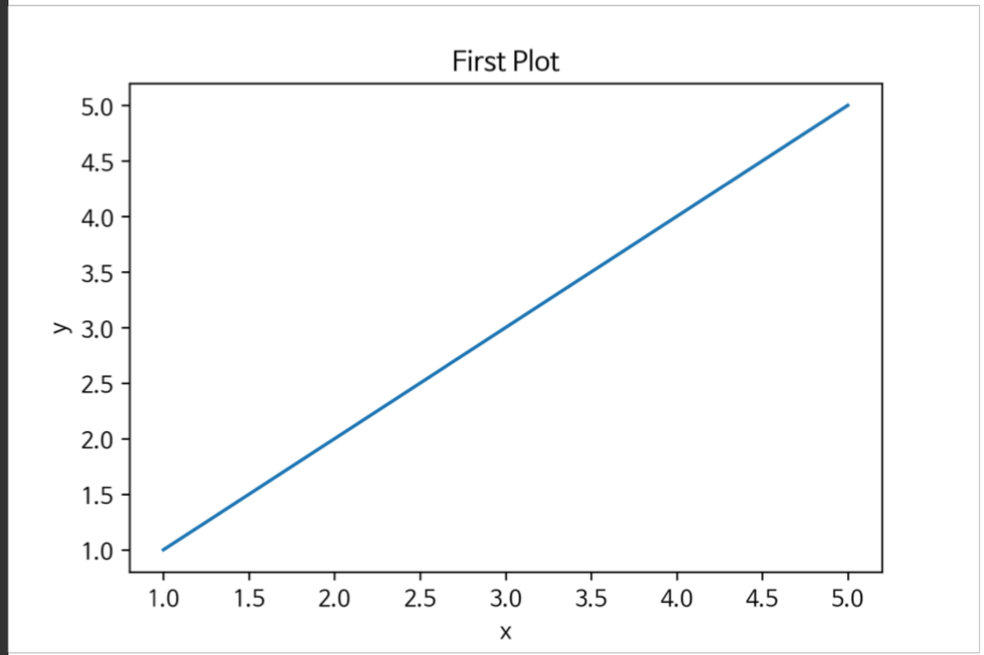
그래프 그려보기

```
x = [1, 2, 3, 4, 5]
y = [1, 2, 3, 4, 5]
plt.plot(x, y)
plt.title("First Plot")
plt.xlabel("x")
plt.ylabel("y")
```

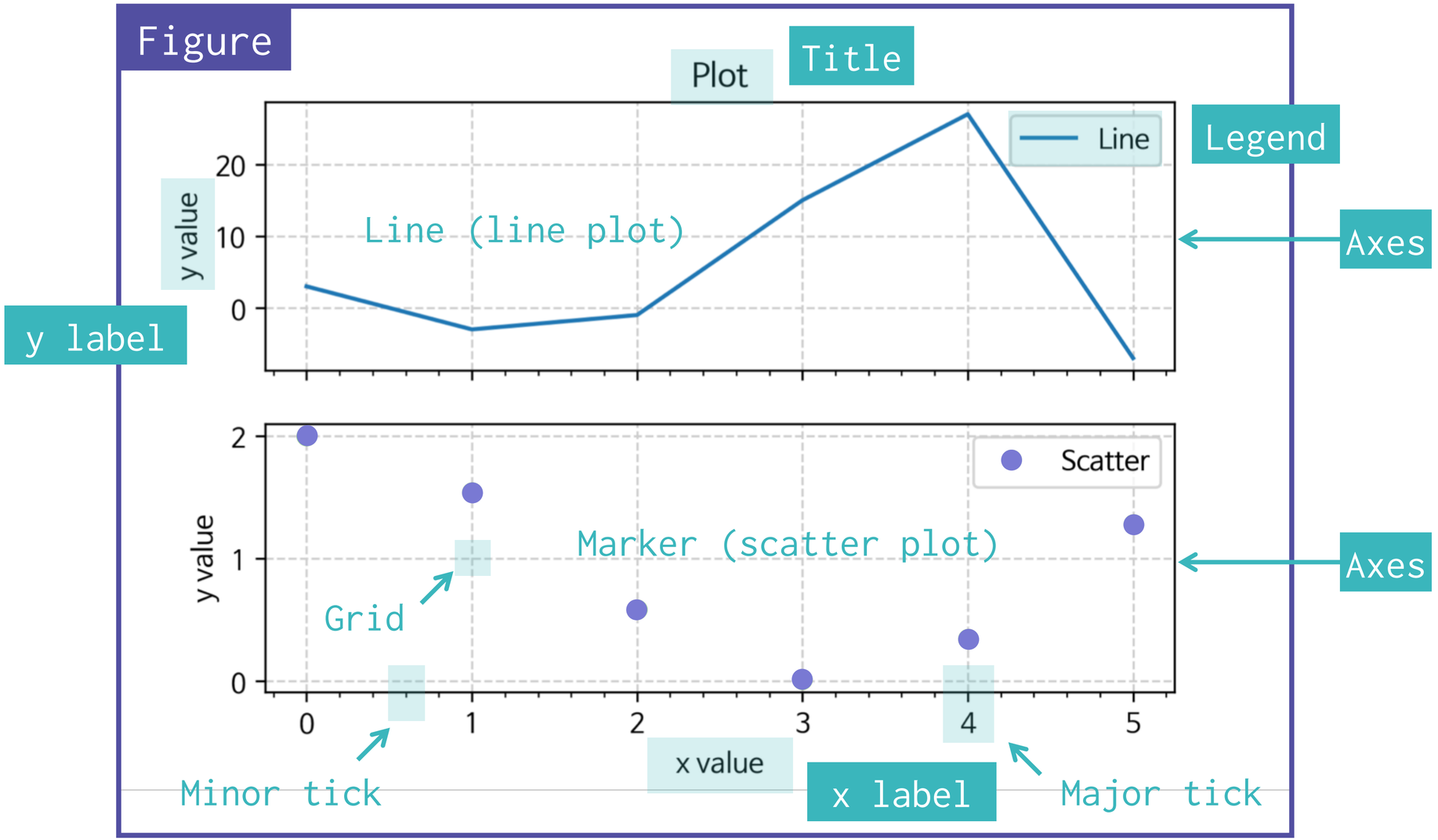


그래프 그려보기

```
x = [1, 2, 3, 4, 5]
y = [1, 2, 3, 4, 5]
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title("First Plot")
ax.set_xlabel("x")
ax.set_ylabel("y")
```

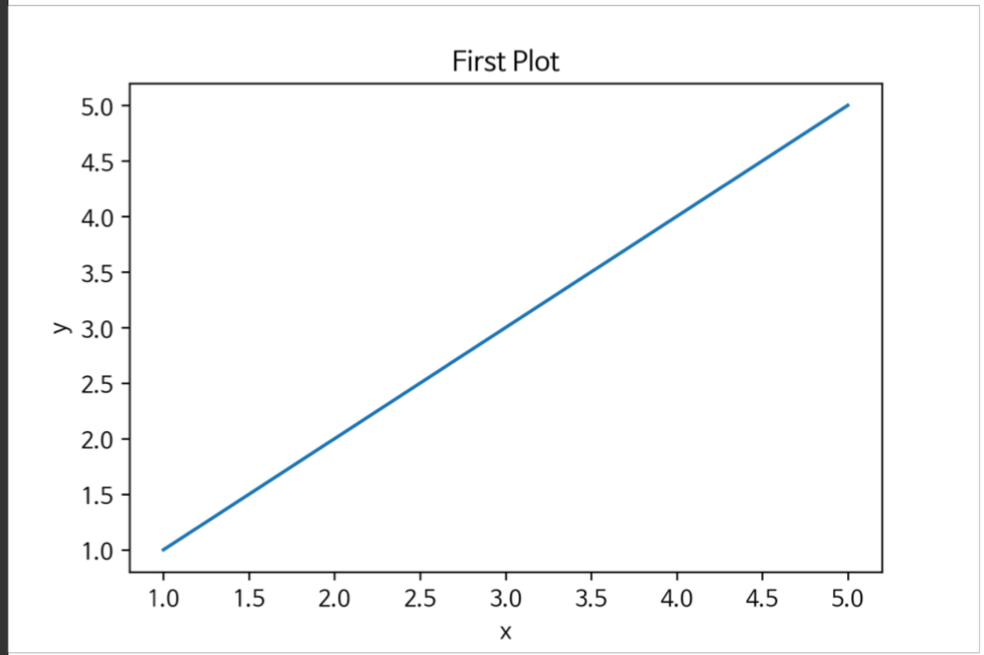


Matplotlib 구조



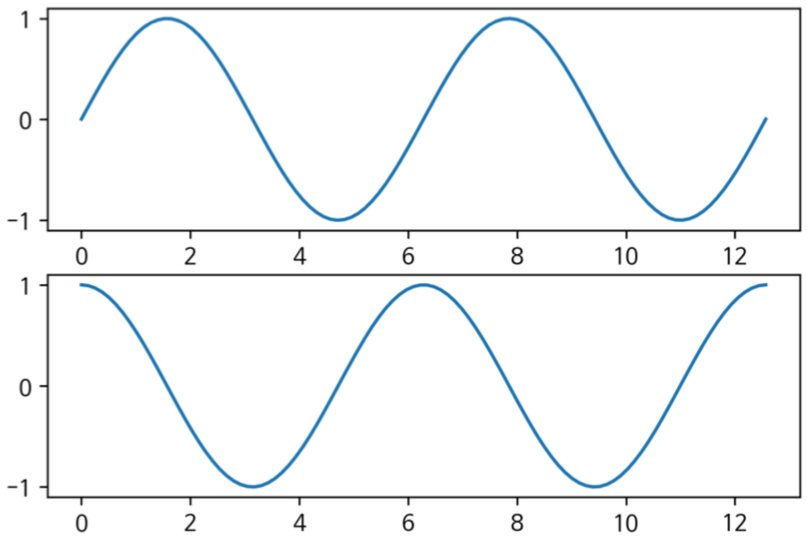
저장하기

```
x = [1, 2, 3, 4, 5]
y = [1, 2, 3, 4, 5]
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title("First Plot")
ax.set_xlabel("x")
ax.set_ylabel("y")
fig.set_dip(300)
fig.savefig(”first_plot.png”)
```



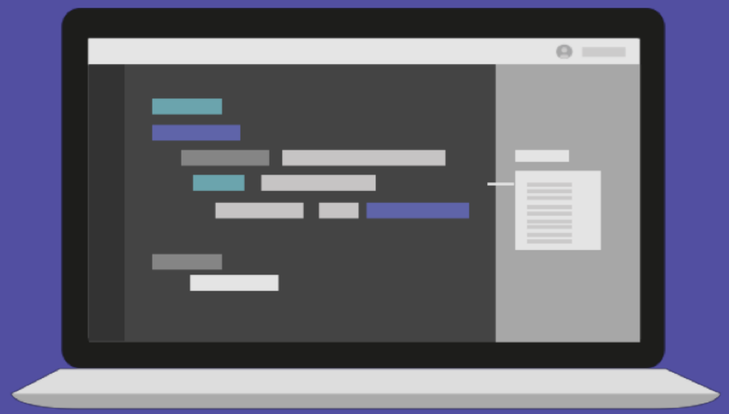
여러개 그래프 그리기

```
x = np.linspace(0, np.pi*4, 100)
fig, axes = plt.subplots(2, 1)
axes[0].plot(x, np.sin(x))
axes[1].plot(x, np.cos(x))
```



[실습1]

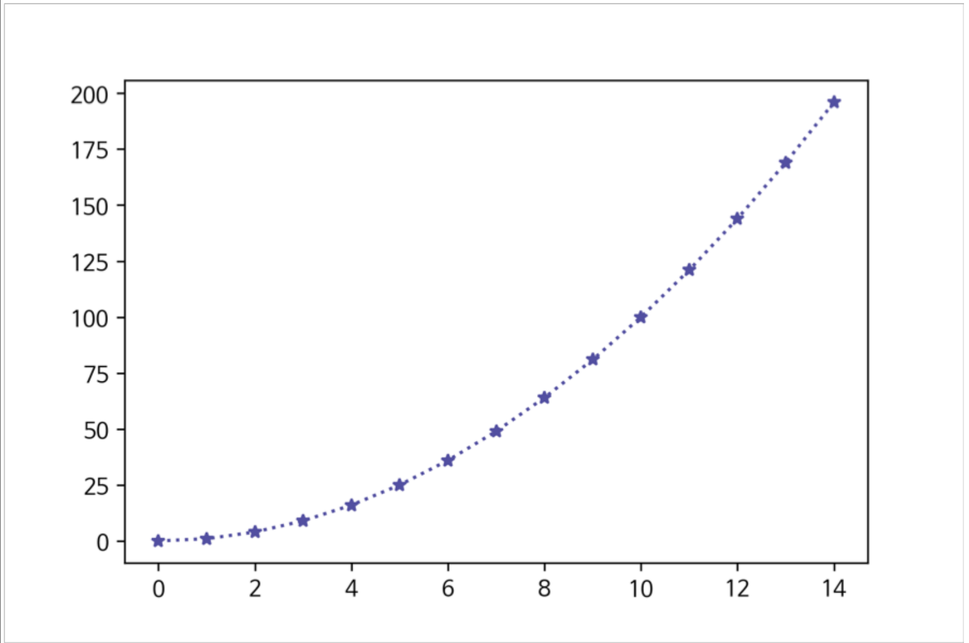
Matplotlib 기초



Matplotlib 그래프

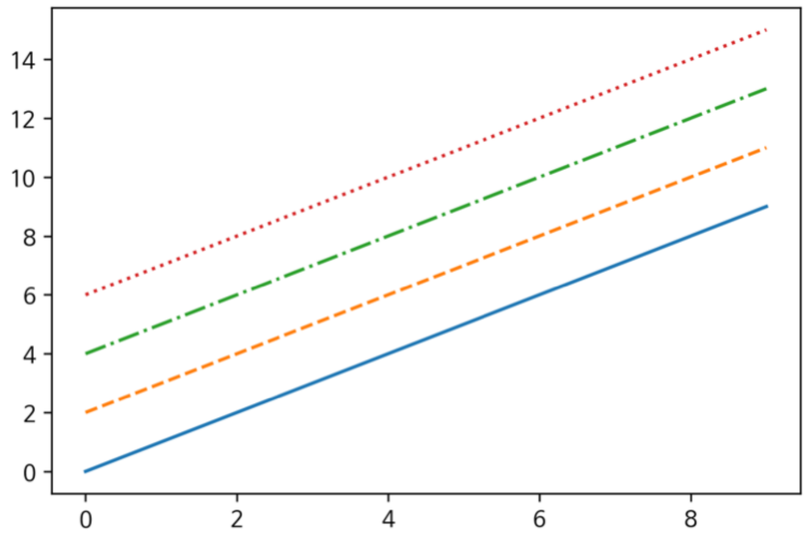
Line plot

```
fig, ax = plt.subplots()
x = np.arange(15)
y = x ** 2
ax.plot(
    x, y,
    linestyle=":",
    marker="*",
    color="#524FA1"
)
```



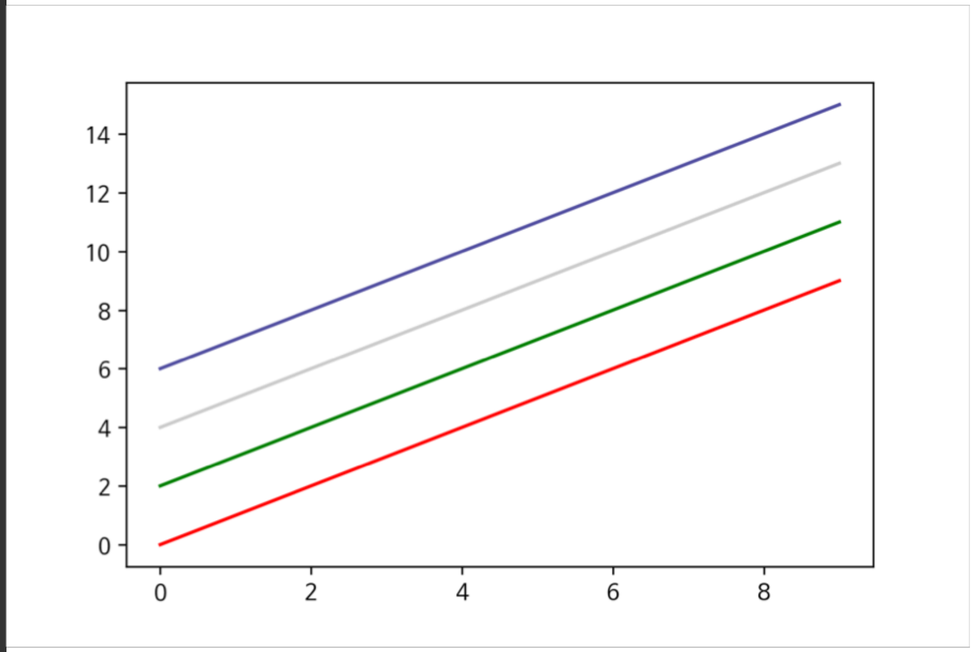
Line style

```
x = np.arange(10)
fig, ax = plt.subplots()
ax.plot(x, x, linestyle="-")
# solid
ax.plot(x, x+2, linestyle="--")
# dashed
ax.plot(x, x+4, linestyle="-.")
# dashdot
ax.plot(x, x+6, linestyle=":")
# dotted
```



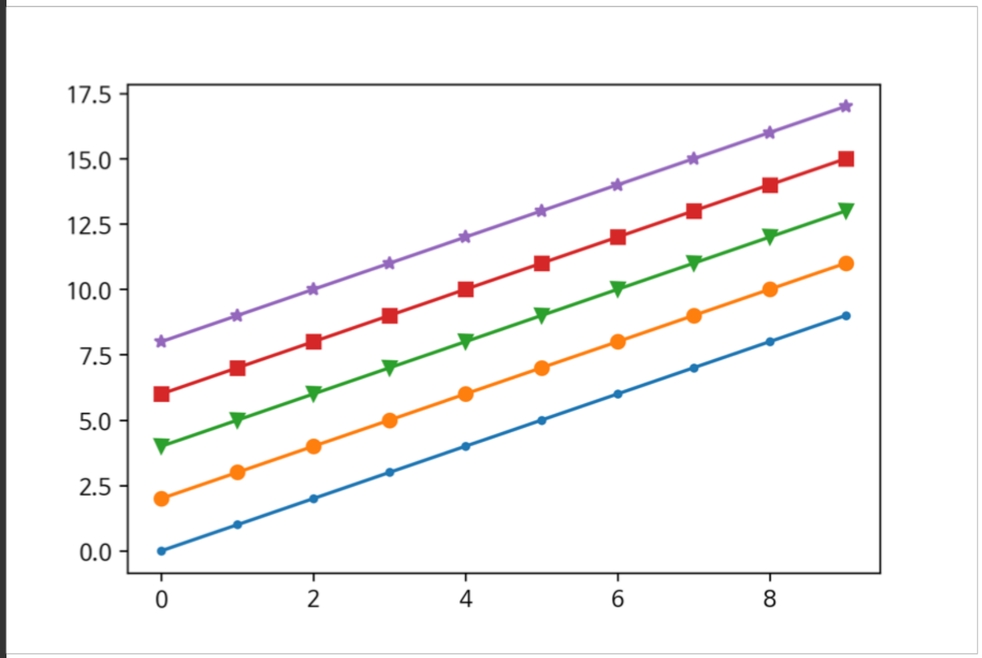
Color

```
x = np.arange(10)
fig, ax = plt.subplots()
ax.plot(x, x, color="r")
ax.plot(x, x+2, color="green")
ax.plot(x, x+4, color='0.8')
ax.plot(x, x+6, color="#524FA1")
```



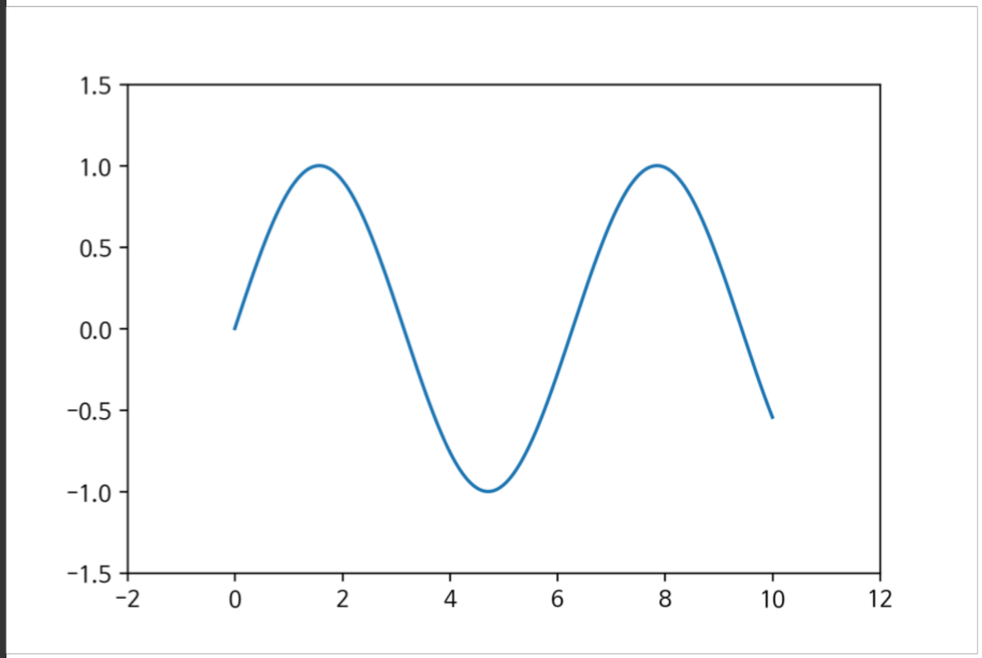
Marker

```
x = np.arange(10)
fig, ax = plt.subplots()
ax.plot(x, x, marker=".")
ax.plot(x, x+2, marker="o")
ax.plot(x, x+4, marker='v')
ax.plot(x, x+6, marker="s")
ax.plot(x, x+8, marker="*")
```



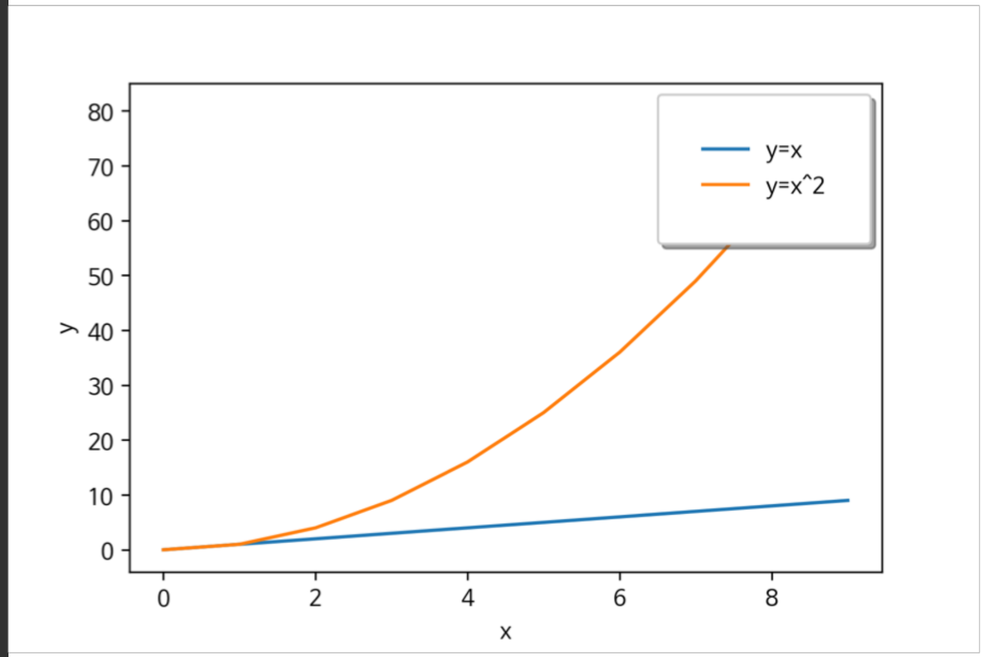
축 경계 조정하기

```
x = np.linspace(0, 10, 1000)
fig, ax = plt.subplots()
ax.plot(x, np.sin(x))
ax.set_xlim(-2, 12)
ax.set_ylim(-1.5, 1.5)
```



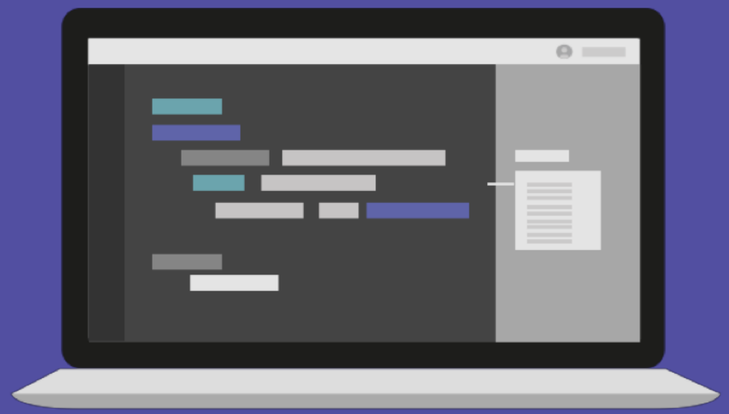
범례

```
fig, ax = plt.subplots()
ax.plot(x, x, label='y=x')
ax.plot(x, x**2, label='y=x^2')
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.legend(
    loc='upper right',
    shadow=True,
    fancybox=True,
    borderpad=2
)
```



[실습2]

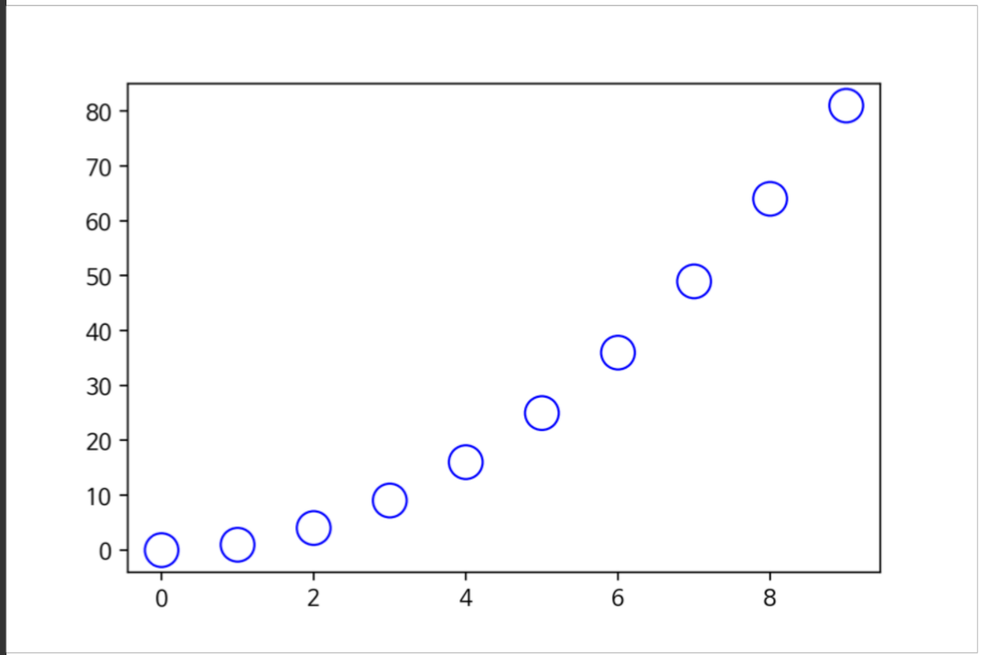
Line Graph



Scatter

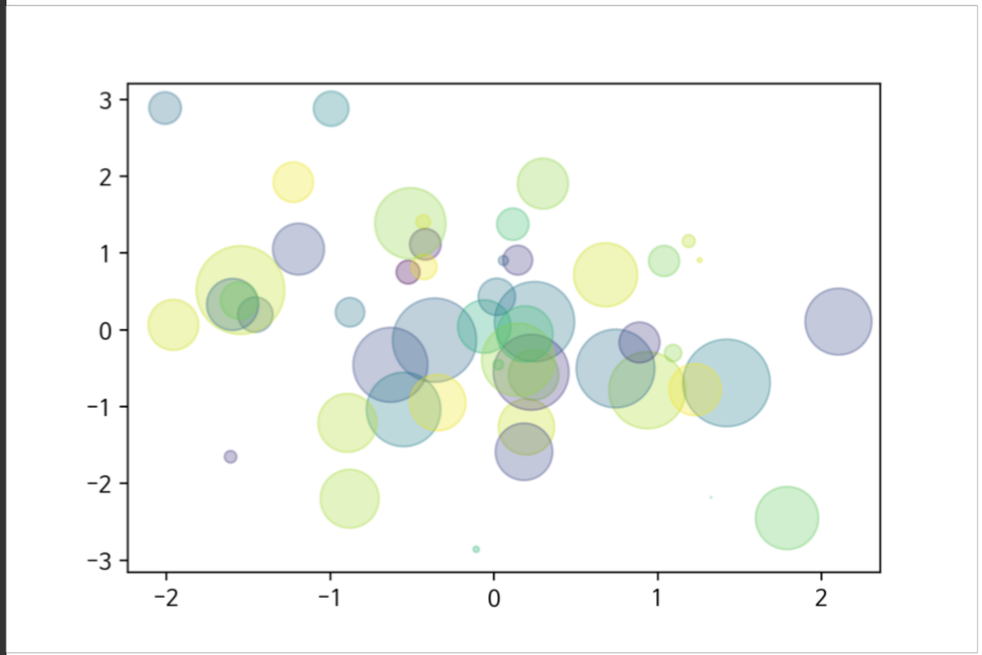
Scatter

```
fig, ax = plt.subplots()
x = np.arange(10)
ax.plot(
    x, x**2, "o",
    markersize=15,
    markerfacecolor='white',
    markeredgecolor="blue"
)
```



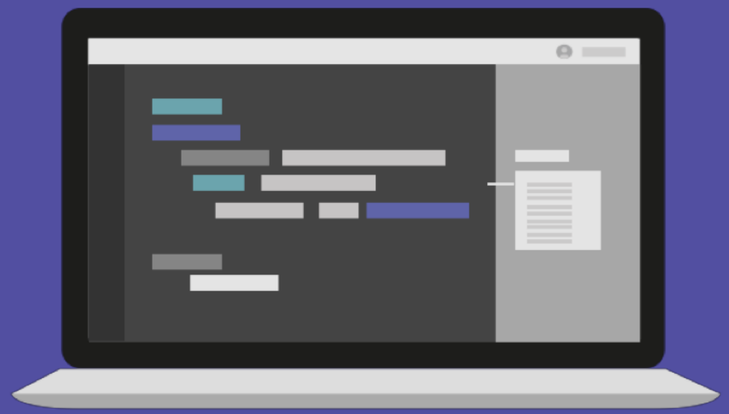
Scatter

```
  fig, ax = plt.subplots()
  x = np.random.randn(50)
  y = np.random.randn(50)
  colors = np.random.randint(0, 100,
50)
  sizes = 500 * np.pi *
np.random.rand(50) ** 2
  ax.scatter(
      x, y, c=colors, s=sizes,
alpha=0.3
  )
```



[실습3]

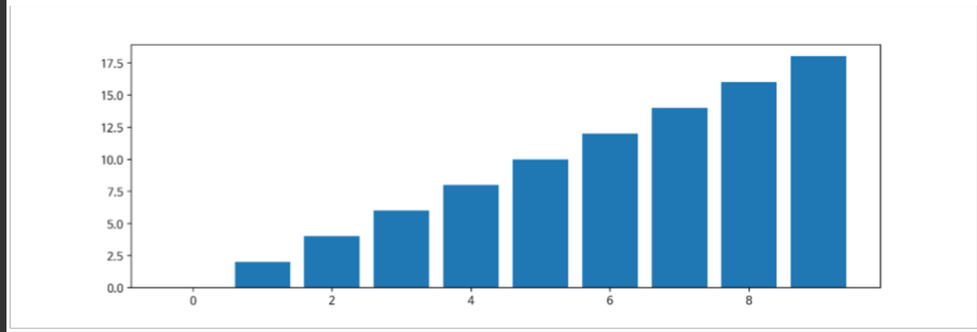
Scatter



Bar & Histogram

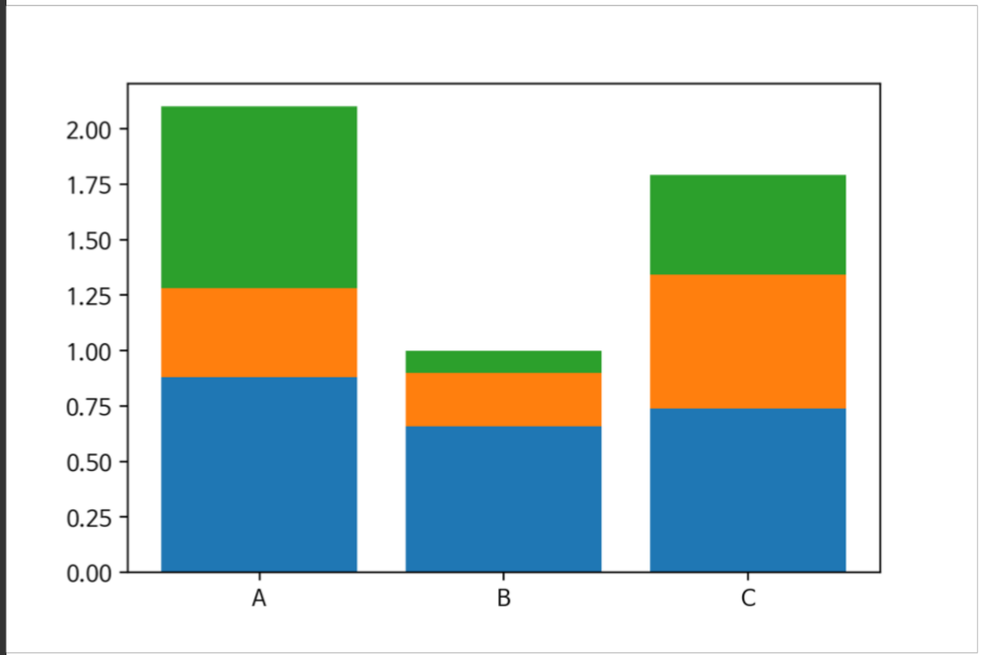
Bar plot

```
  # bar
  x = np.arange(10)
  fig, ax =
plt.subplots(figsize=(12, 4))
  ax.bar(x, x*2)
```



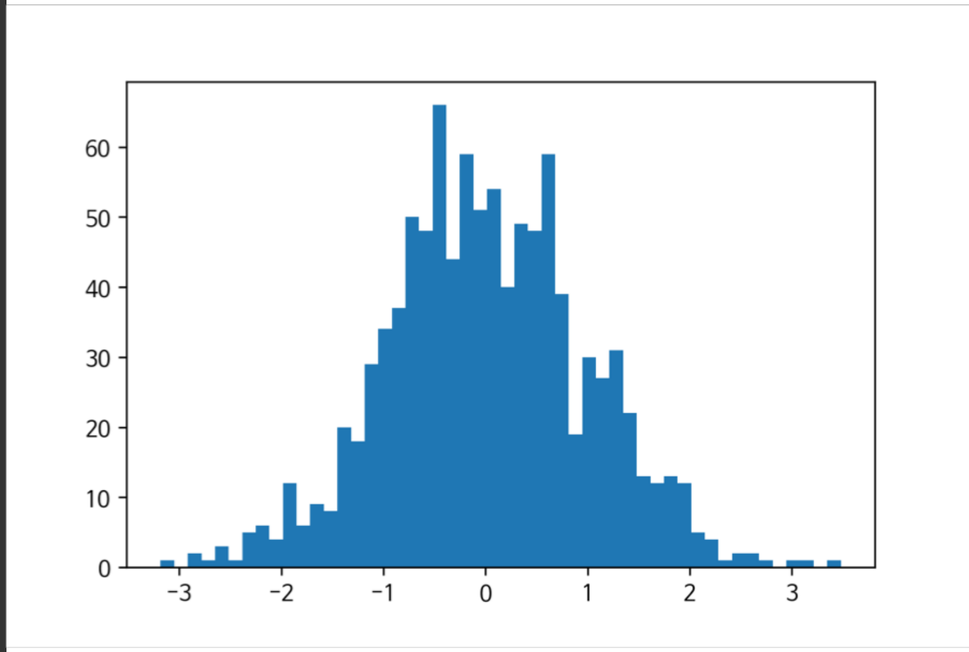
Bar plot

```
x = np.random.rand(3)
y = np.random.rand(3)
z = np.random.rand(3)
data = [x, y, z]

fig, ax = plt.subplots()
x_ax = np.arange(3)
for i in x_ax:
    ax.bar(x_ax, data[i],
    bottom=np.sum(data[:i], axis=0))
ax.set_xticks(x_ax)
ax.set_xticklabels(["A", "B", "C"])
```



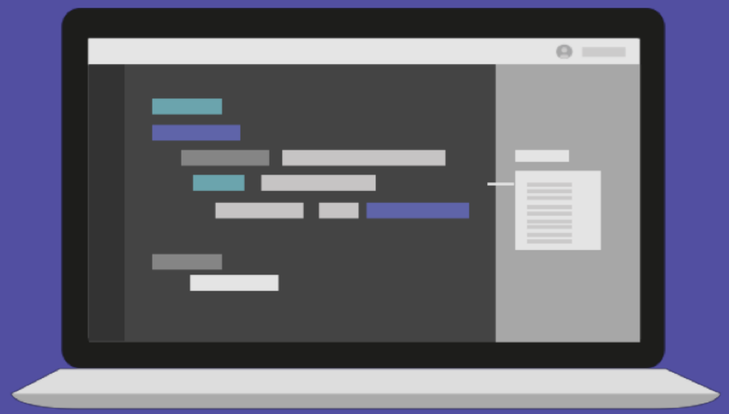
Histogram

```
fig, ax = plt.subplots()

data = np.random.randn(1000)

ax.hist(data, bins=50)
```



[실습4]

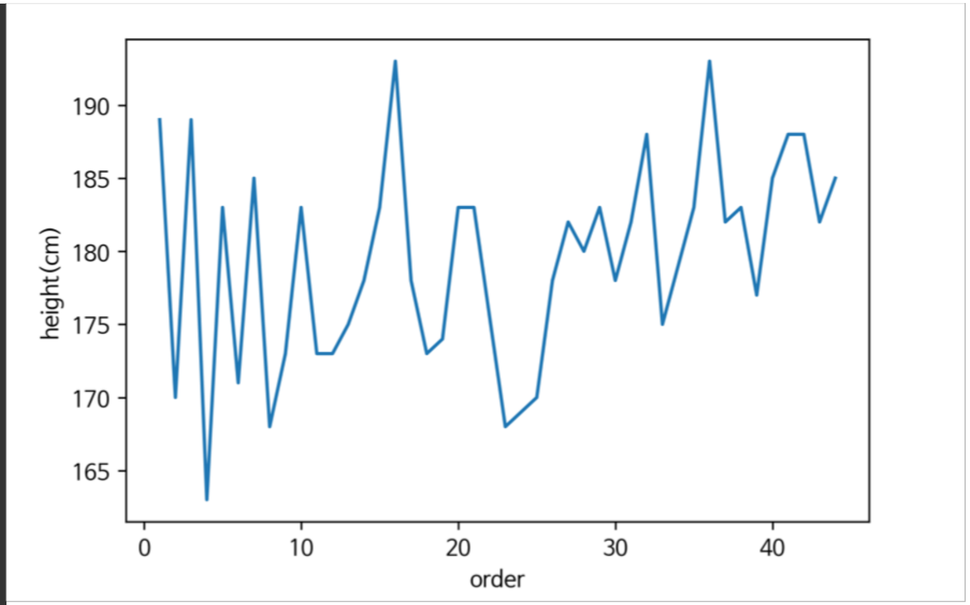
Bar & Histogram



Matplotlib with Pandas

Matplotlib with pandas

```
df = pd.read_csv(
"./president_heights.csv")
fig, ax = plt.subplots()
ax.plot(df["order"],
df["height(cm)"], label="height")
ax.set_xlabel("order")
ax.set_ylabel("height(cm)")
```



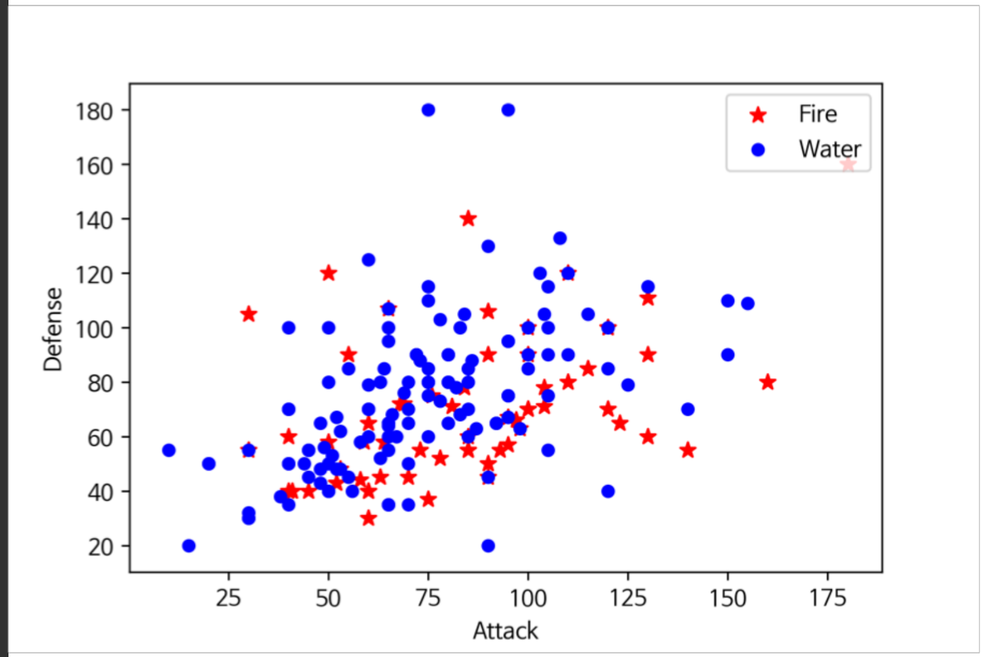
	order	name	height(cm)
0	1	George Washington	189
1	2	John Adams	170
2	3	Thomas Jefferson	189
3	4	James Madison	163
4	5	James Monroe	183

Matplotlib with pandas

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
5	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	1	False
6	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
7	6	CharizardMega Charizard X	Fire	Dragon	634	78	130	111	130	85	100	1	False

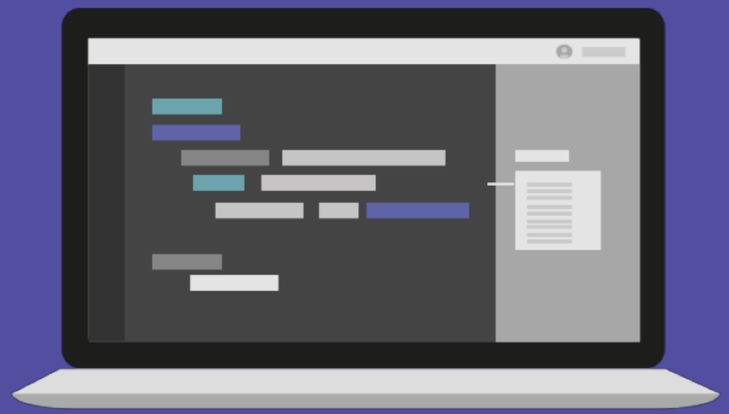
Matplotlib with pandas

```
    fire = df[
          (df['Type 1']=='Fire') | ((df['Type
2'])=="Fire")
    ]
    water = df[
          (df['Type 1']=='Water') | ((df['Type
2'])=="Water")
    ]
    fig, ax = plt.subplots()
    ax.scatter(fire['Attack'], fire['Defense'],
          color='R', label='Fire', marker="*", s=50)
    ax.scatter(water['Attack'], water['Defense'],
          color='B', label="Water", s=25)
    ax.set_xlabel("Attack")
    ax.set_ylabel("Defense")
    ax.legend(loc="upper right")
```



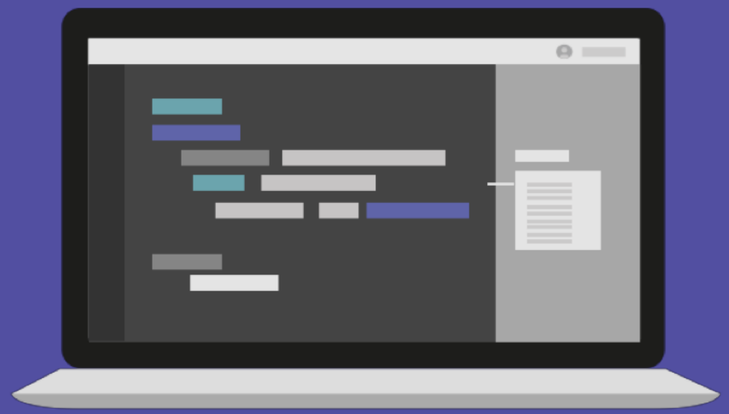
[실습5]

Matplotlib with Pandas



[실습6]

토끼와 거북이 경주 결과 시각화



/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice