머신러닝을 위한 수학

머신러닝을 위한 수학 툴 익히기

/* elice */

목차

커리큘럼

1. 머신러닝이란?

머신러닝의 정의와 개요에 대해 학습합니다.

2. 머신러닝을 위한 수학이 필요한 이유?

머신러닝의 기초와 이를 위한 수학에 대해 학습합니다.

3. Python 데이터 처리

Python의 여러 자료형 데이터의 처리에 대해 학습합니다.

커리큘럼

4. Numpy 사용하기

Python의 Numpy 사용에 대해 학습합니다.

5. Pandas 사용하기

Python의 Pandas 사용에 대해 학습합니다.

6. Matplotlib 사용하기

Python의 Matplotlib 사용에 대해 학습합니다.

💡 추천대상

1. 머신러닝 입문자

머신러닝을 얼핏 알지만, 이해는 못하는 사람

2. 데이터 분석 입문자

파이썬 라이브러리를 실용적으로 활용해보고 싶은 사람

3. 벡터 행렬을 모르는 사람

머신러닝의 이해에 필수적인 벡터와 행렬에 대해 모르는 사람

🎯 수강목표

1. 머신러닝의 전반에 대해 이해합니다.

인공지능과 머신러닝의 차이를 알고, 일반적인 머신러닝의 구조를 이해합니다.

2. 머신러닝 속의 본질적 수학 지식을 이해합니다.

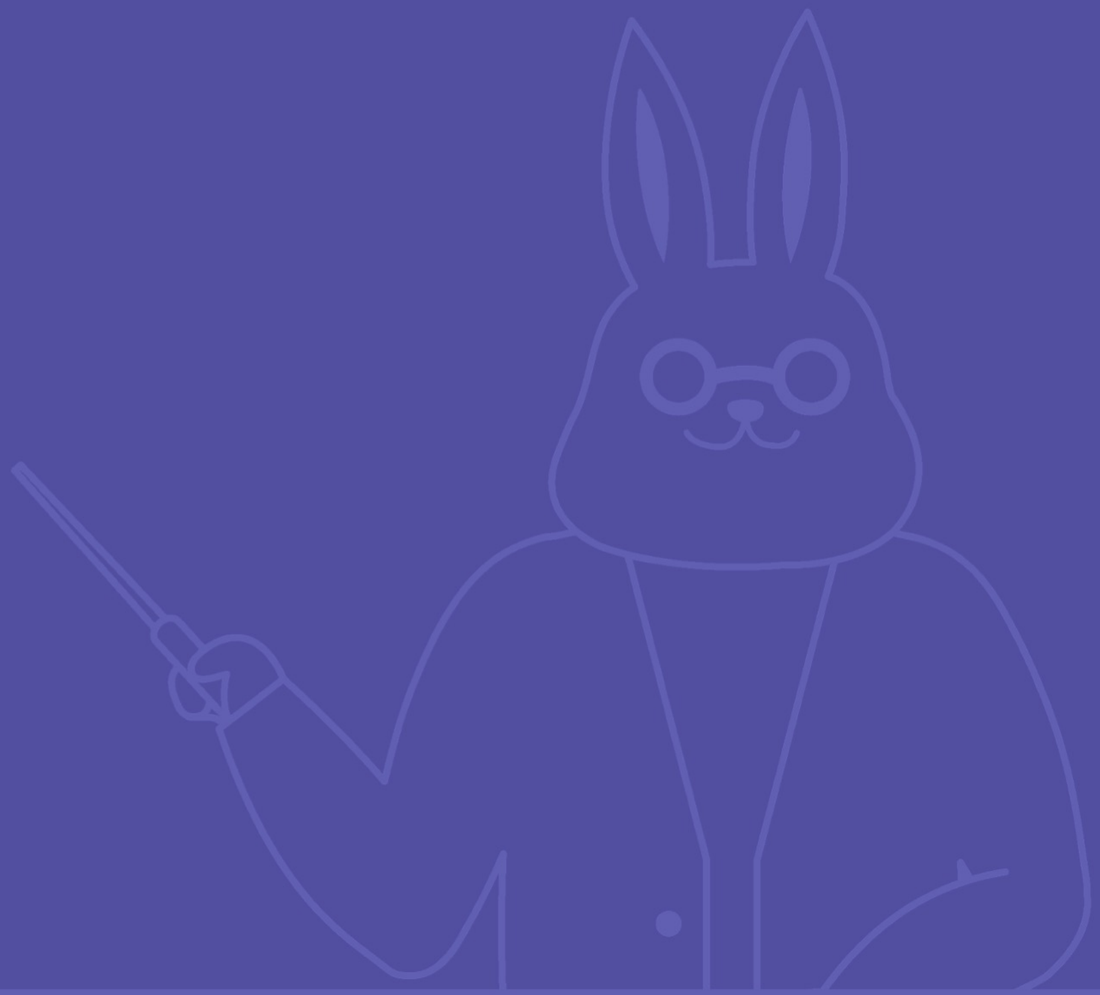
막연하고 이해하기 어려웠던 지식을 체계적으로 배우며 익힙니다.

3. 어떤 머신러닝 기법을 마주하더라도 두렵지 않습니다.

익힌 수학 지식으로 머신러닝 속의 여러 기법들을 접해도 어렵지 않습니다.

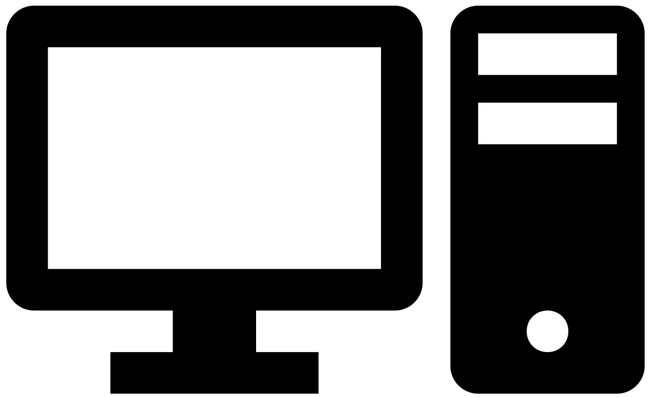
01

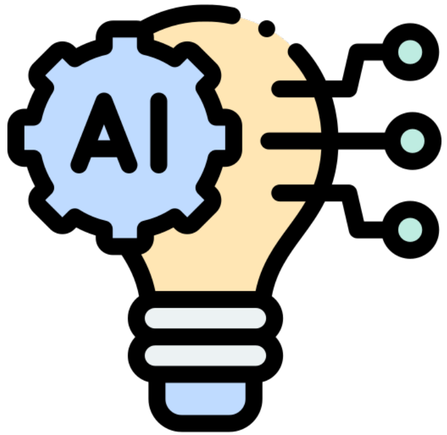
머신러닝이란?



인공지능(Artificial Intelligence; AI)

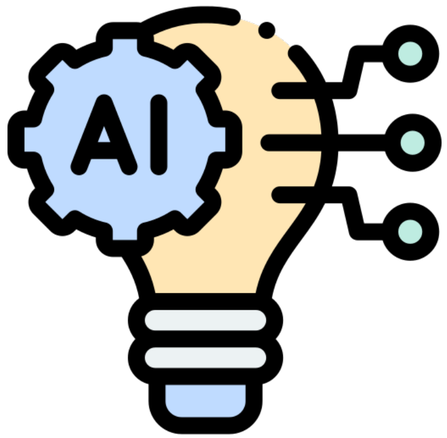
- 지능을 갖고 있는 기능을 갖춘 컴퓨터 시스템
- 인간의 지능을 기계 등에 인공적으로 구현한 것. 예) 컴퓨터





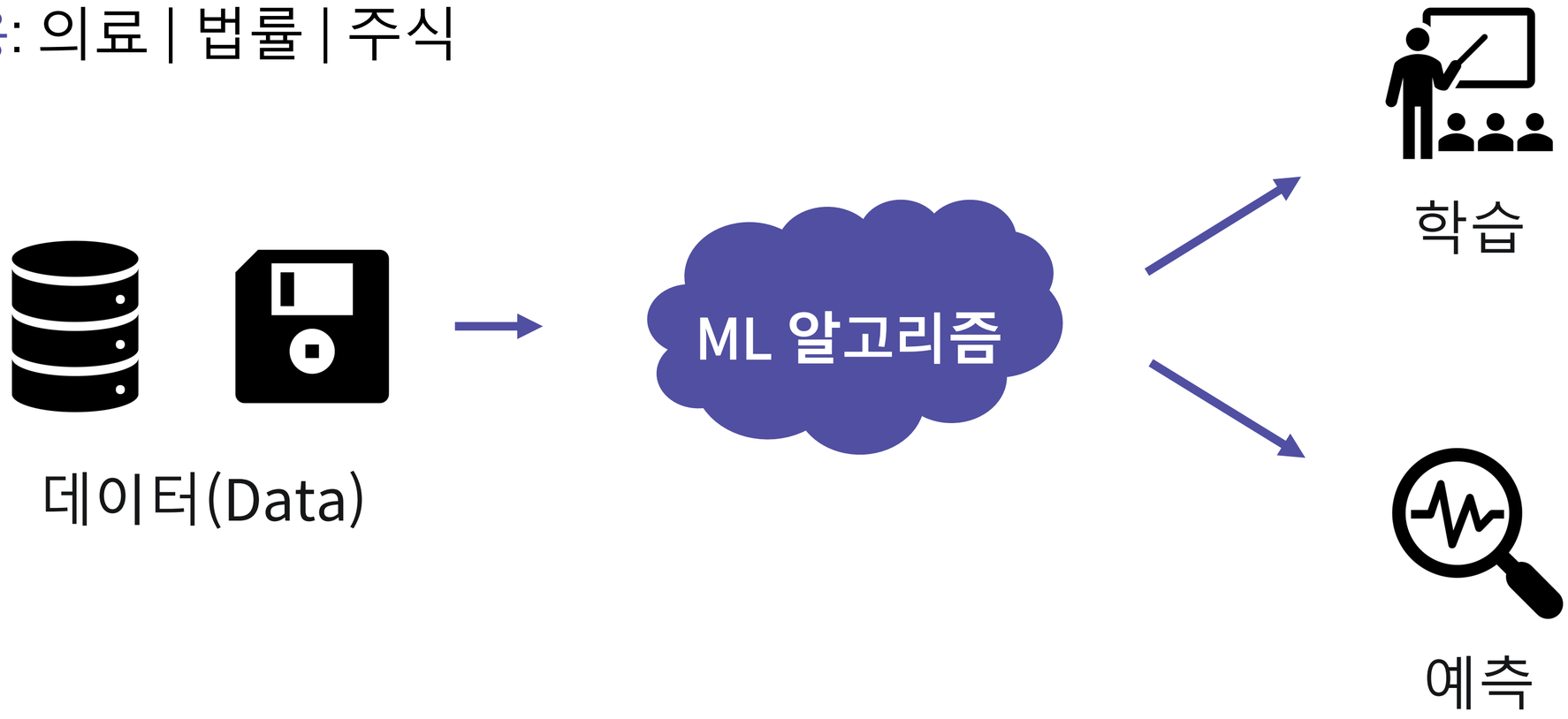
머신러닝(Machine Learning; ML)

- **경험**을 통해 자동으로 개선하는 컴퓨터 알고리즘의 연구임.
- **기계학습**이라고도 부름.
- **훈련 데이터**를 통해 학습해 경험을 쌓아 예측하는 방법들임.



ML 파이프라인

- **데이터 – 알고리즘(모델) – 학습/예측**.
- 전문가의 개입 없이 기계가 자동으로 학습함.
- 응용: 의료 | 법률 | 주식

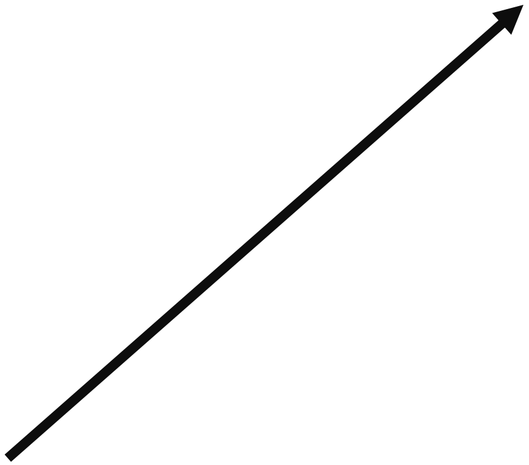


02

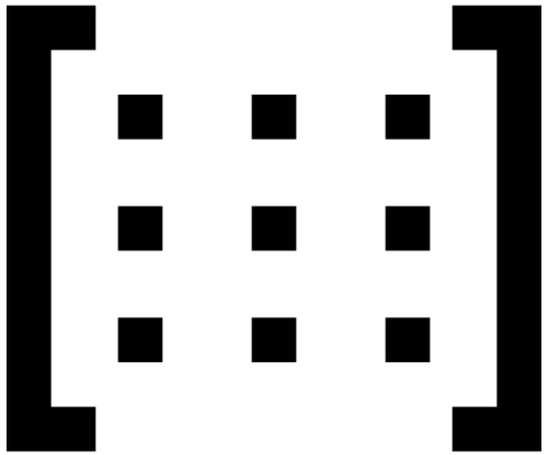
머신러닝을 위한 수학이 필요한 이유?

머신러닝은 **데이터**(data) 속의 **표현**(representation)을 기계가 학습하는 것이다.

데이터는 대부분 **고차원**(high dimension)이다. 즉, **벡터**(vector)와 **행렬**(matrix)로 구성되어 있다.



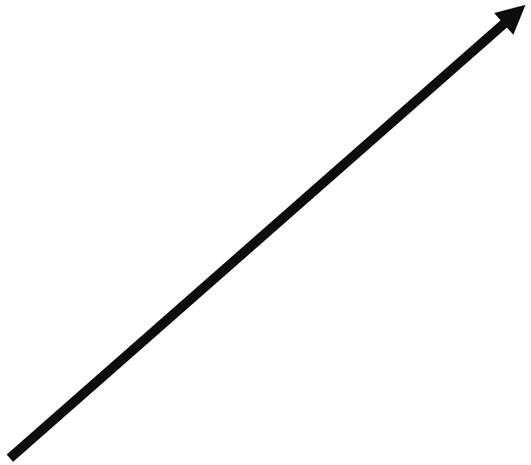
벡터(Vector)



행렬(Matrix)

의료 데이터를 통해 예를 들어보자.

병원은 환자 A에 대한 정보를 “성별, 나이, 몸무게, 키, 심장질환 유무”로 저장한다고 하자. 그렇다면?



벡터(Vector)

환자 A

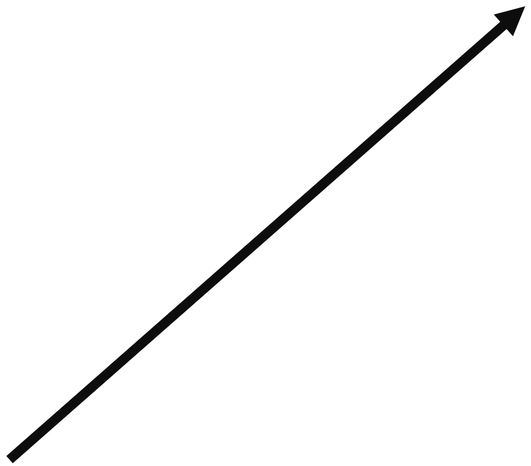
남자
29
65
174
심장질환 없음

5차원

여러 명이 있다면?

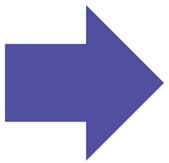
5 x N 의 행렬!

ML 알고리즘은 고차원 데이터의 표현을 학습 목적에 맞게 뽑아내는 함수(function)을 학습함.



벡터(Vector)

X



예측

~~**고혈압 X**~~

고혈압 O

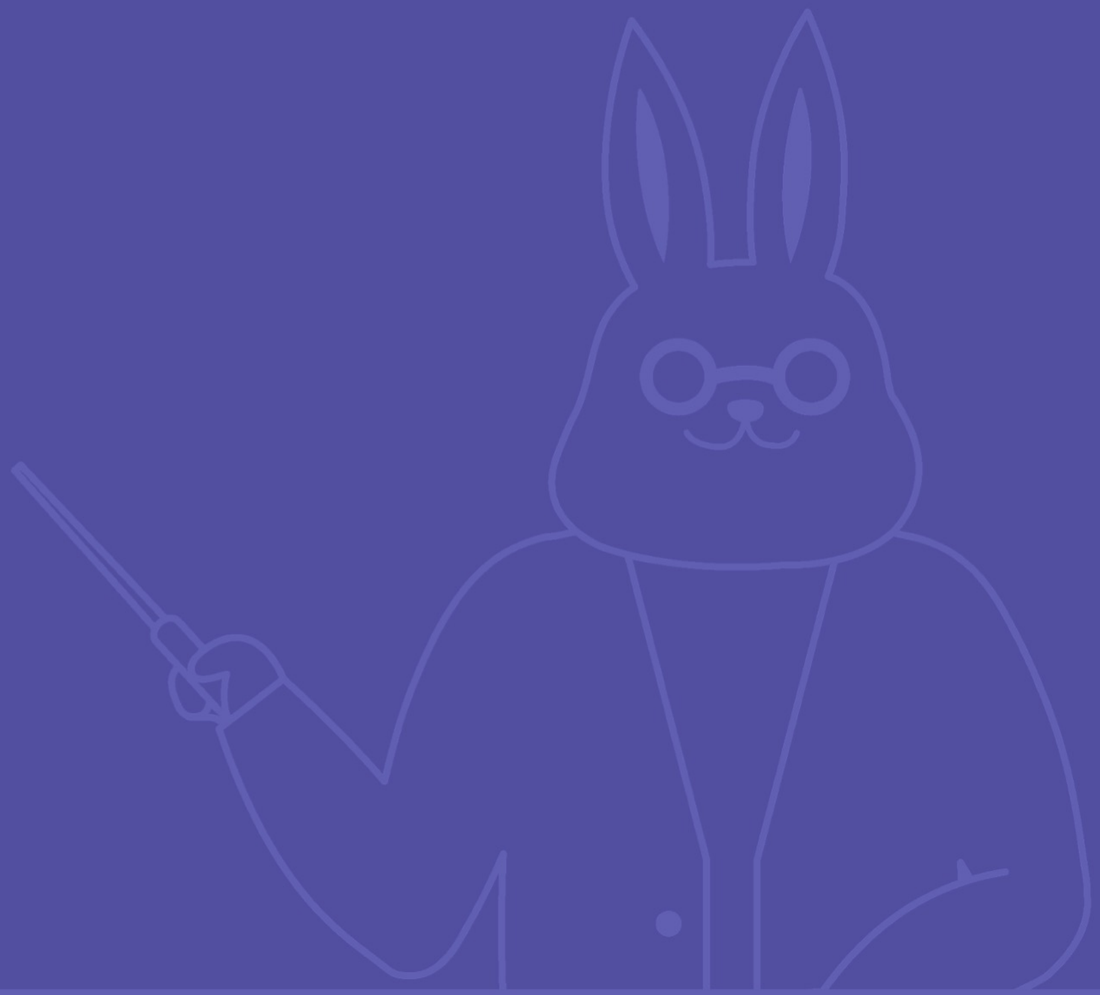
- 어떻게 a, b를 결정할까?
- Vector 함수의 사칙연산?

예시

$F(X) \dashrightarrow F(X) = aX + b$

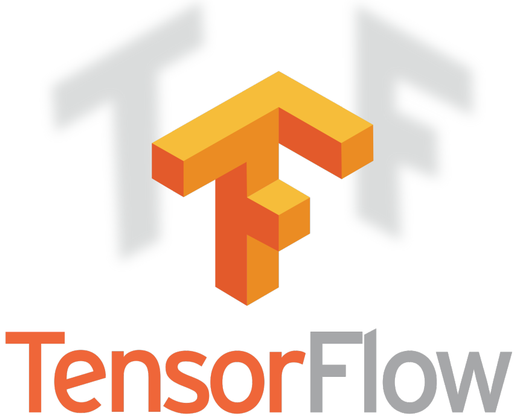
03

Python 데이터 처리



/* elice */

Python은 Data Science, Machine Learning 등 다양한 분야에서 강력하게 활용됨.



[복습] Python의 특징

1. 들여쓰기 + 세미콜론 없음

2. 별도의 컴파일 필요 없음. Interpreter 방식으로 결과 바로 확인 가능

이외

오픈소스 라이브러리: 다양하게 공개적으로 공유됨- GitHub
- 최근에는 학회에서 발표되는 ML 알고리즘의 코드가 대부분 Python으로 작성되고 있으며, 이는 GitHub 등을 통하여 새롭게 공개되고 있다.

여러가지 자료형이 있음.

1. 문자열(String) : “Hello world”
2. **리스트(List) : [1, 0, -0.2, 1]**
3. 튜플(Tuple) : (1, 3, 3)
4. 집합(Set) : {1, 3}
5. 딕셔너리(Dictionary) : {“1”: “one”, “3”: “three”}

리스트 자료형

- 정수나 실수 등의 여러 값을 요소로 가져 하나의 변수에 많은 데이터를 담은 형태임.
- 대괄호 [] 안에 콤마로 구분해 원소(element) 나열
- 어떠한 자료형도 포함이 가능하고 중복될 수 있음.
- 순서를 가짐.

예시: **리스트 자**료형

```
weight1 = 76
weight2 = 54
weight3 = 61
weight4 = 59


weight = [76, 54, 61, 59]
# 4차원의 data
```

예시: 리스트 자료형

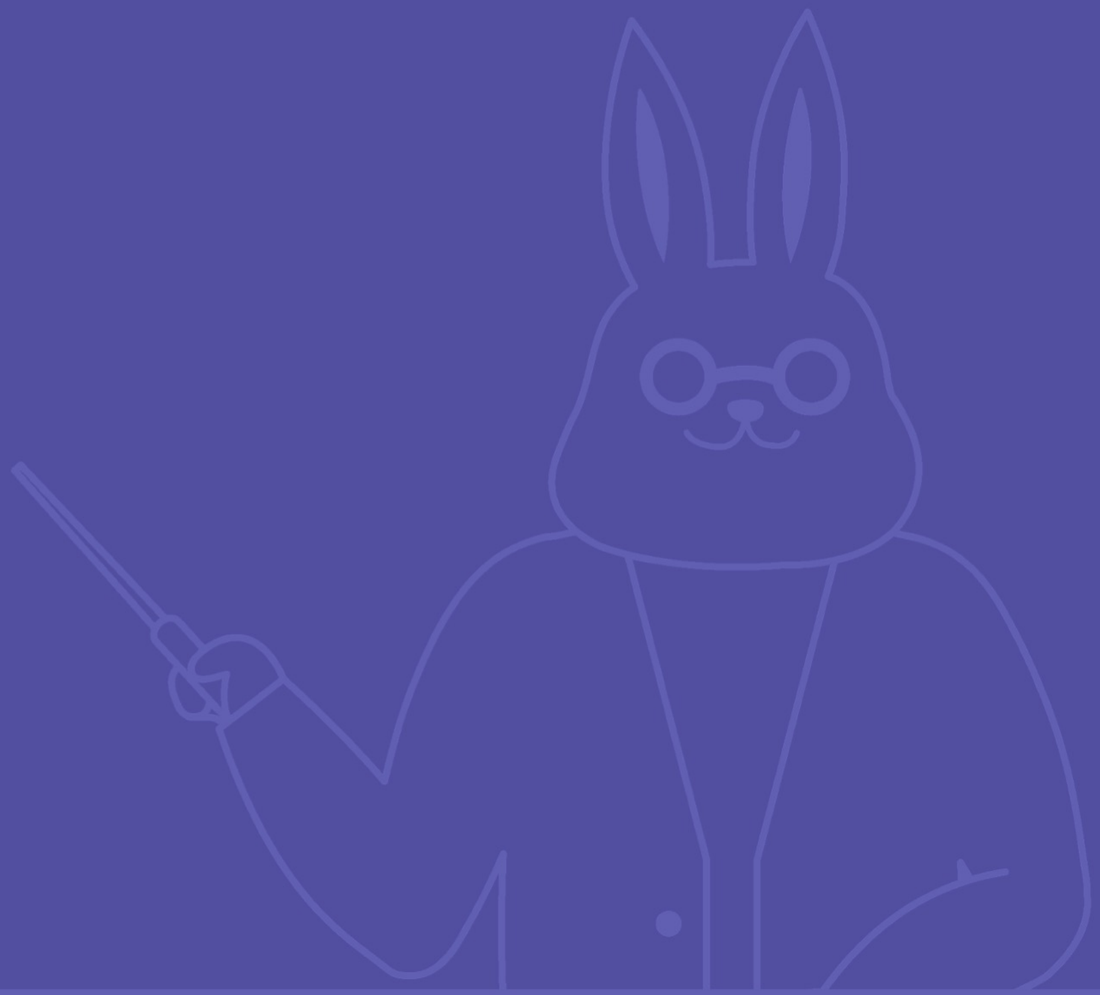
```
lst1 = [1 , 7, 5, 3, 2]
print ('first element: {}'.format(lst1[0]))
print ('last element: {}'.format(lst1[len(lst1)-1]))
print ('last element: {}'.format(lst1[-1]))
print ('first to third elements: {}'.format(lst1[0:3]))
print ('sorted lst: {}\n'.format(sorted(lst1)))
```

결과

```
first element: 1
last element: 2
last element: 2
first to third elements: [1, 7, 5]
sorted lst: [1, 2, 3, 5, 7]
```

04

Numpy 사용하기



Numpy는 **행렬이나 일반적으로 대규모 다차원 배열**을 쉽게 처리 할 수 있도록 지원하는 파이썬의 라이브러리임.

주로, 벡터와 행렬의 **효율적인 계산을** 다루는 라이브러리임.

벡터, 행렬 및 대규모 다차원 배열(Tensor)를 쉽게 처리하도록 지원함.

Numpy 예시

```
import numpy as np

a = np.array([1, 2, 3])
print(type(a), a.shape, a[0], a[1], a[2])
a[0] = 5
print(a)
```

numpy를 np로 import함

3차원의 데이터를 a로 생성함
<class 'numpy.ndarray'> (3,) 1 2 3
a의 첫번째 값을 5로 대체함
[5 2 3]

Numpy 예시

```
import numpy as np

a = np.zeros((2,2))
print(a, a.shape)
```

0으로 구성된 행렬
[[0. 0.] [0. 0.]] , (2, 2)

Numpy 예시

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
print(x + y)
print(np.add(x, y))
```

```
[[ 6. 8.] [10. 12.]]
 [[ 6. 8.] [10. 12.]]
```

Numpy의 효율성

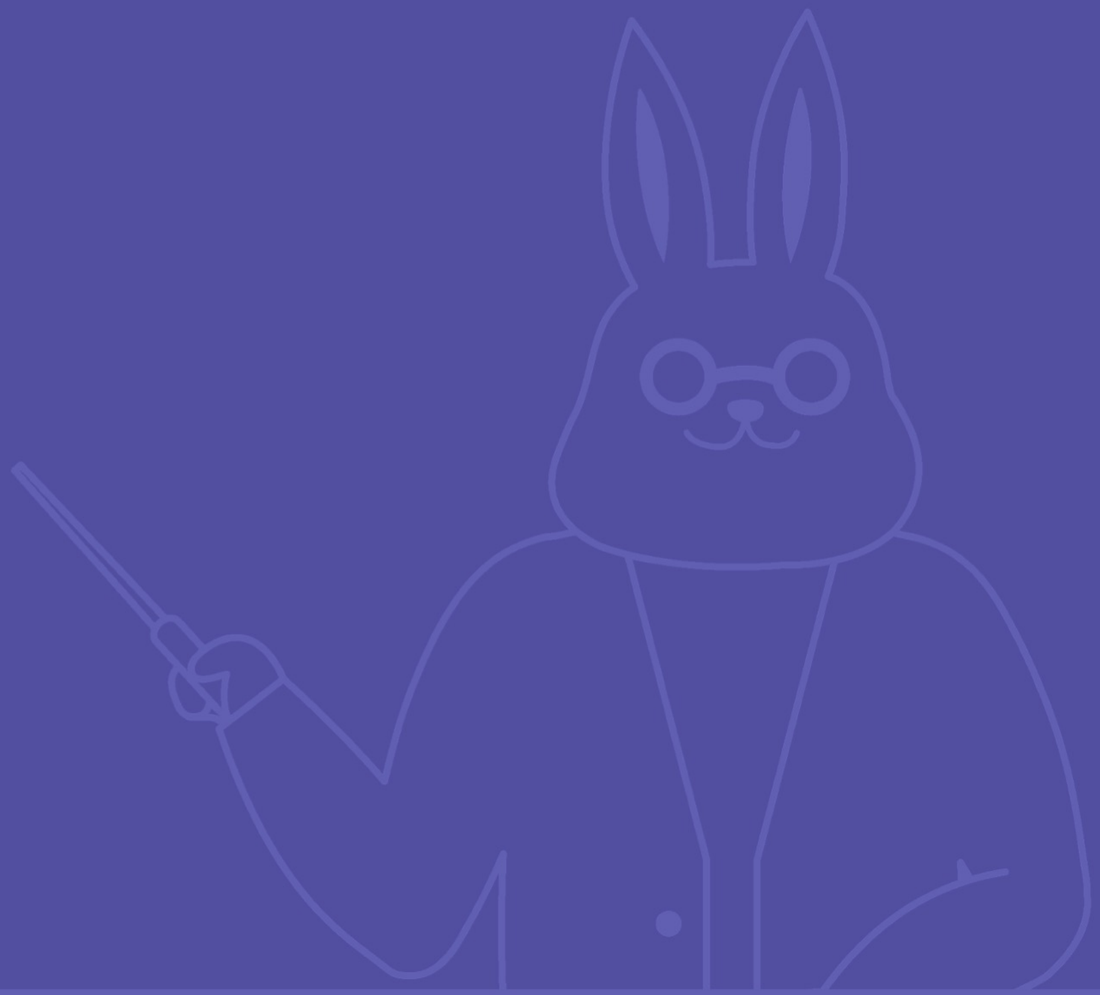
Numpy를 이용해서 벡터 연산을 하는 것이 빠를까?

Vs

Python에 내장된 사칙연산으로 벡터 계산을 하는 것이 빠를까?

05

Pandas 사용하기



Pandas는 Data Science에서 데이터 분석을 위해 주로 사용되는 라이브러리임.

주로 행렬과 같은 테이블 형식(Table, csv)으로 이루어진 데이터를 읽는데 사용됨.

Pandas 예시

```
import pandas as pd

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

a = [1, 2, 3, 4]

print(pd.Series(a))
```

```
0    1
1    2
2    3
3    4
dtype: int64
0    1
1    2
2    3
3    4
dtype: int64
```

Pandas 예시

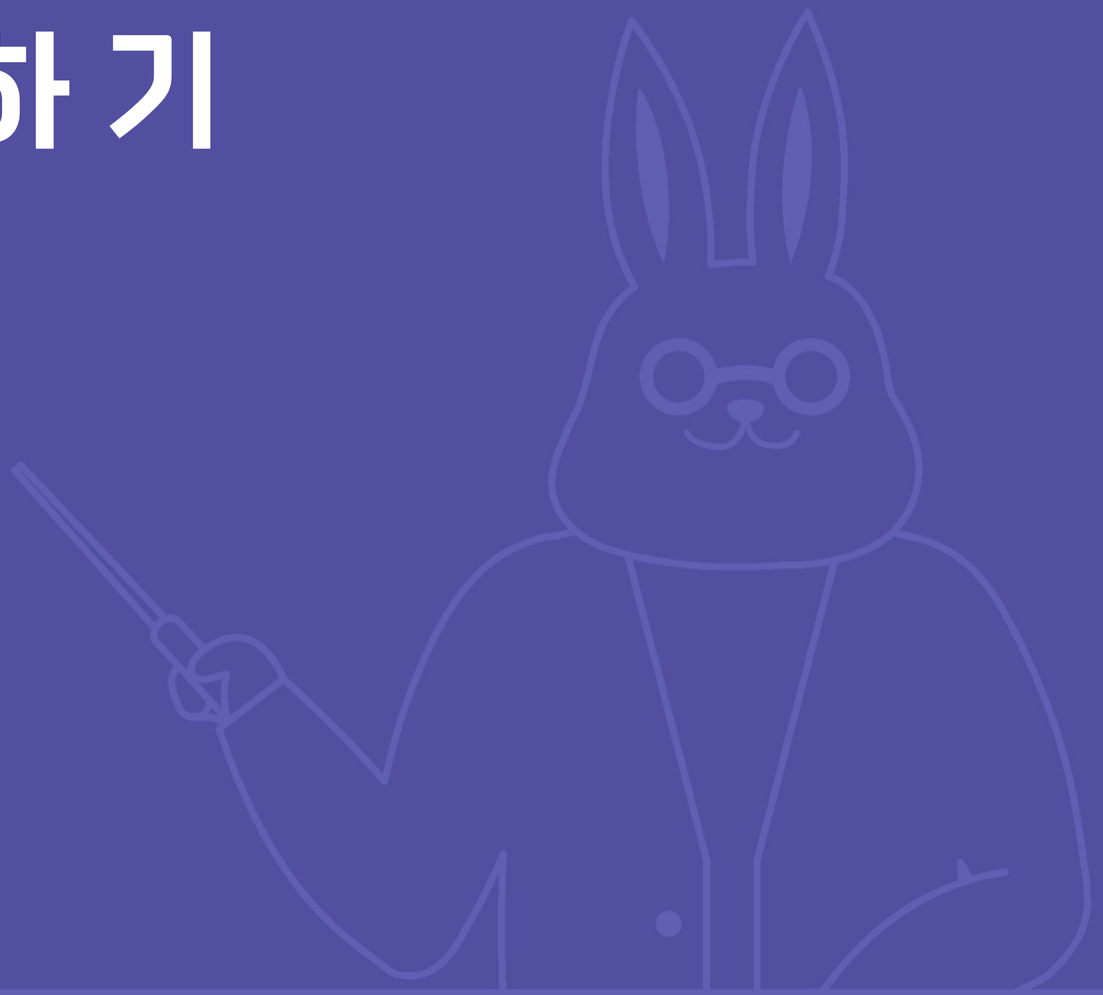
각 Key값과 Value(1차원데이터)가
DataFrame의 하나의 컬럼과 2차원 데이터가 됩니다.

```
company1 = [['삼성', 1000000, '갤럭시노트'],
 ['애플', 1200000, '아이폰'], ['엘지', 900000, '윙']]
print(pd.DataFrame(company1))


df1 = pd.DataFrame(company1)
print(df1)

df1.columns = ['회사명', '가격', '제품명’]
print(df1)
```

```
     0        1       2
0   삼성  1000000  갤럭시노트
1   애플  1200000   아이폰
2   엘지   900000     윙
     0        1       2
0   삼성  1000000  갤럭시노트
1   애플  1200000   아이폰
2   엘지   900000     윙
   회사명       가격     제품명
0   삼성  1000000  갤럭시노트
1   애플  1200000   아이폰
2   엘지   900000     윙
```

06

Matplotlib 사용하기



Matplotlib는 함수를 그래프로 시각화하는 도구임.

➔ Data Science에서는 데이터 분석 결과를 시각화하는 것에 활용됨.

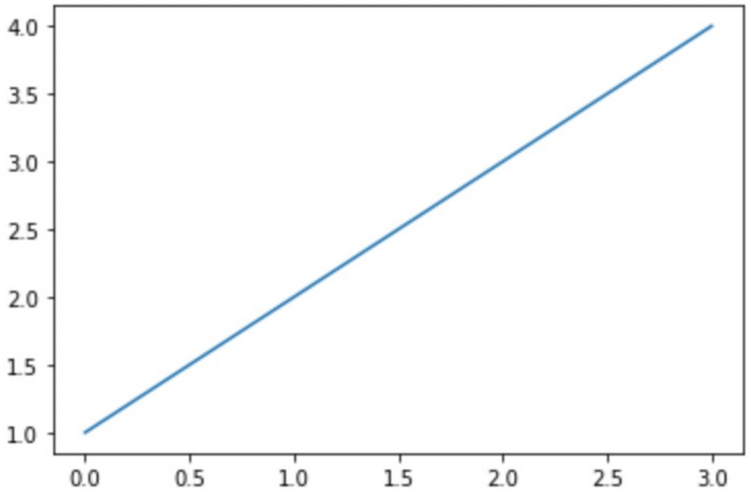
데이터를 꺾은 선 등의 선이나 점의 형태의 그래프를 기본적으로 그림을 제공함.

matplotlib.pyplot을 사용할 것임.

Matplotlib 예시

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4])
plt.show()
```

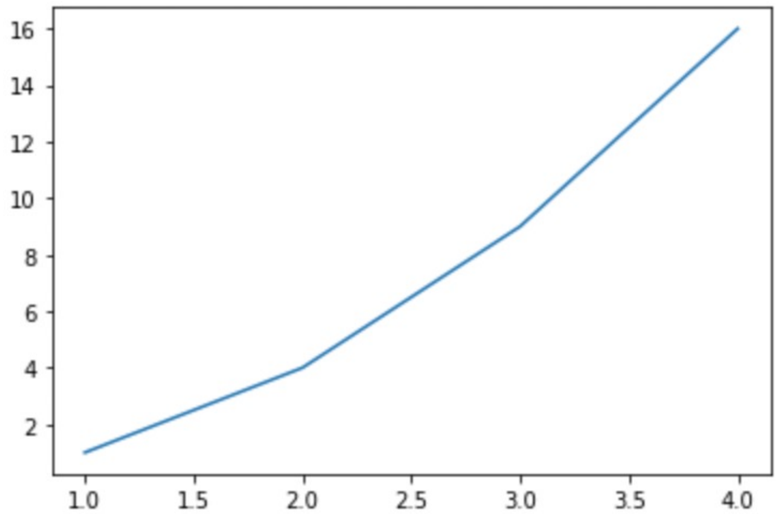
하기의 그림을 보여줌.
plot 속의 입력을 y로 인지하고, 자동으로 x는 [0,1,2,3]으로 간주함.



Matplotlib 예시

```
import matplotlib.pyplot as plt


plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```

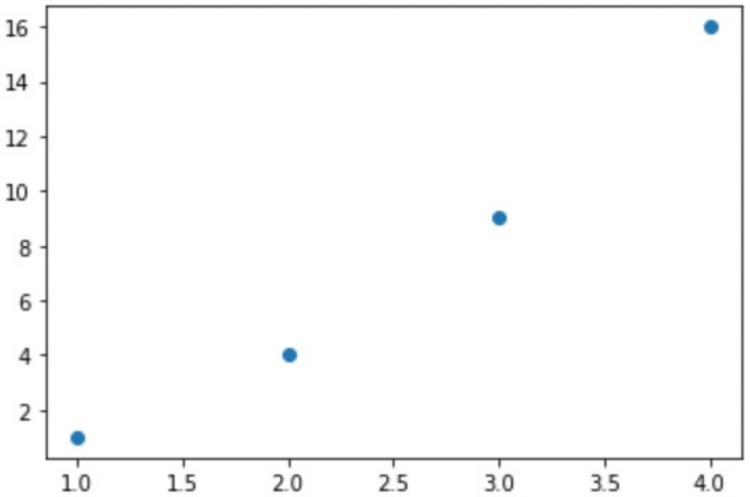
하기의 그림을 보여줌.
x를 [1,2,3,4] 로, y를 [1,4,9,16]으로 간주함.



Matplotlib 예시

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'o')

plt.axis([0.9, 4.1, 0, 16.5])
plt.show()
```

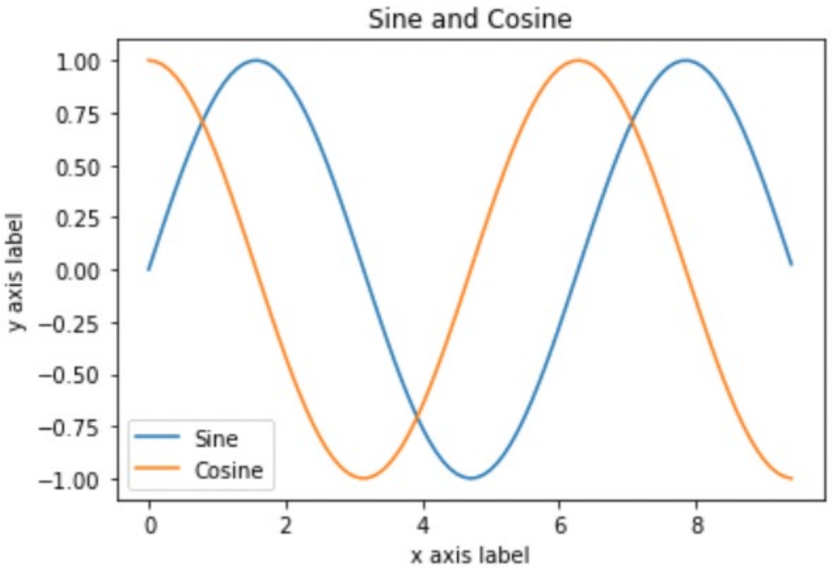
하기의 그림을 보여줌.
‘o’를 통해 점의 형태로 시각화할 수 있음.



Numpy와 Matplotlib의 혼합

```
x = np.arange(0, 3 * np.pi, 0.1)

y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
```

sin과 cos함수 그리기.



크레딧

/* elice */

코스 매니저

콘텐츠 제작자

강사

감수자

디자이너

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io