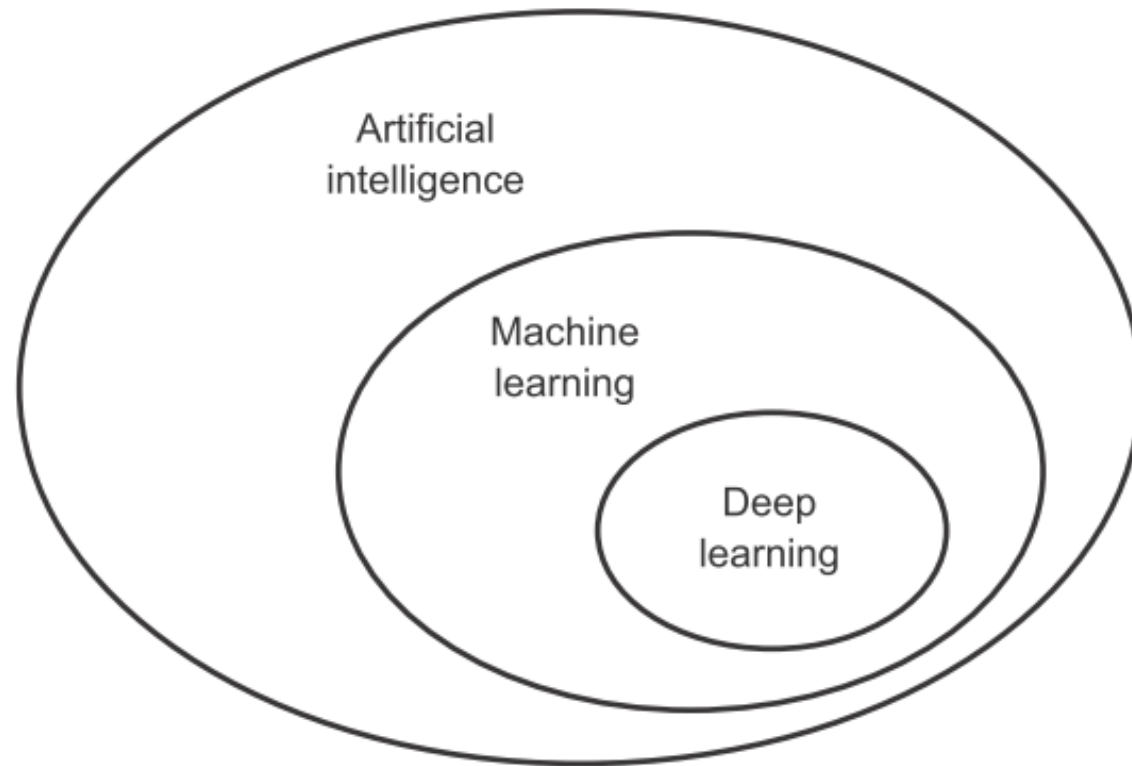


인공지능 기본 용어 및 기술 이해

조용균

인공지능, 머신러닝, 딥러닝

- 인공지능, 머신러닝, 딥러닝의 관계



인공지능

- 보통의 사람이 수행하는 지능적인 작업을 자동화하기 위한 연구 활동
- 1950년대 초부터 연구
- 머신러닝과 딥러닝을 포함하는 종합 분야
- 머신러닝 이전까지는 프로그래머가 규칙을 직접 코딩하는, 학습 과정이 없는 방법론이 대세
 - 전문가 시스템Expert System

머신러닝

- 데이터로부터 학습하도록 컴퓨터를 프로그래밍하는 과학/기술
- “어떤 작업 T에 대한 컴퓨터 프로그램의 성능을 P로 측정했을 때 경험 E로 인해 성능이 향상되었다면 이 컴퓨터 프로그램은 작업 T와 성능 측정 P에 대해 경험 E로 학습한 것이다.”
 - 톰 미첼, 1997
 - 예) 스팸 메일 필터
 - 작업 T: 새 메일이 스팸인지 구분하는 것
 - 경험 E: 훈련 데이터 혹은 훈련 세트
 - 성능 측정 P: 정확도(정확히 분류된 메일의 비율)

머신러닝

- “명시적인 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구 분야이다.”
 - 아서 사무엘, 1959
- 프로그램이 최적의 문제 해결 규칙을 스스로 찾도록 구성된 알고리즘
- 학습
 - 시스템을 문제 해결에 대해 일반화시키는 과정
 - 머신러닝 모델의 오차를 최소화하기 위해 모델 파라미터를 수정해 나가는 과정

왜 머신러닝인가

- 기존 솔루션으로는 많은 수동 조정과 규칙이 필요한 문제
 - 하나의 머신러닝 모델이 코드를 간단하고 더 잘 수행되도록 작성될 수 있음
- 전통적인 방식으로는 전혀 해결할 방법이 없는 복잡한 문제
 - 가장 뛰어난 머신러닝 기법으로 해결 방법을 찾을 수 있음
- 유동적인 환경
 - 머신러닝 시스템은 새로운 데이터에 적응할 수 있음

왜 머신러닝인가

- 머신러닝을 통한 학습

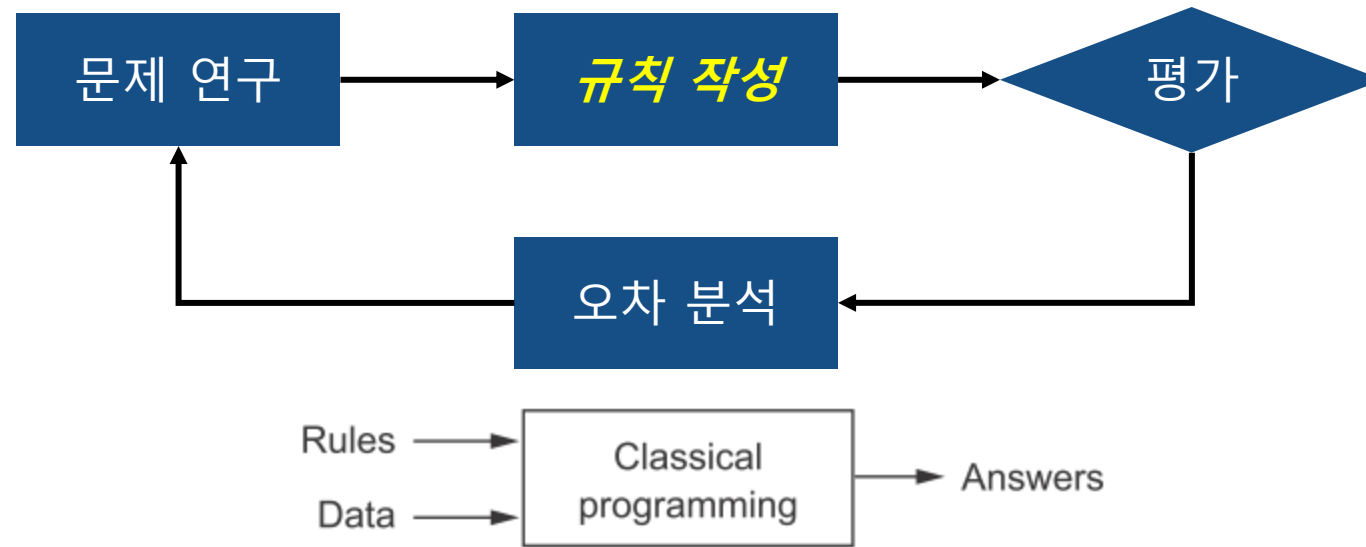
- 머신러닝 모델이 학습한 내용으로부터 문제 해결 방식을 배울 수 있음

- 복잡한 문제와 대량의 데이터로부터 통찰 얻기

- 머신러닝 기술로 대용량 데이터를 분석하면 겉으로 보이지 않던 패턴을 발견할 수 있음

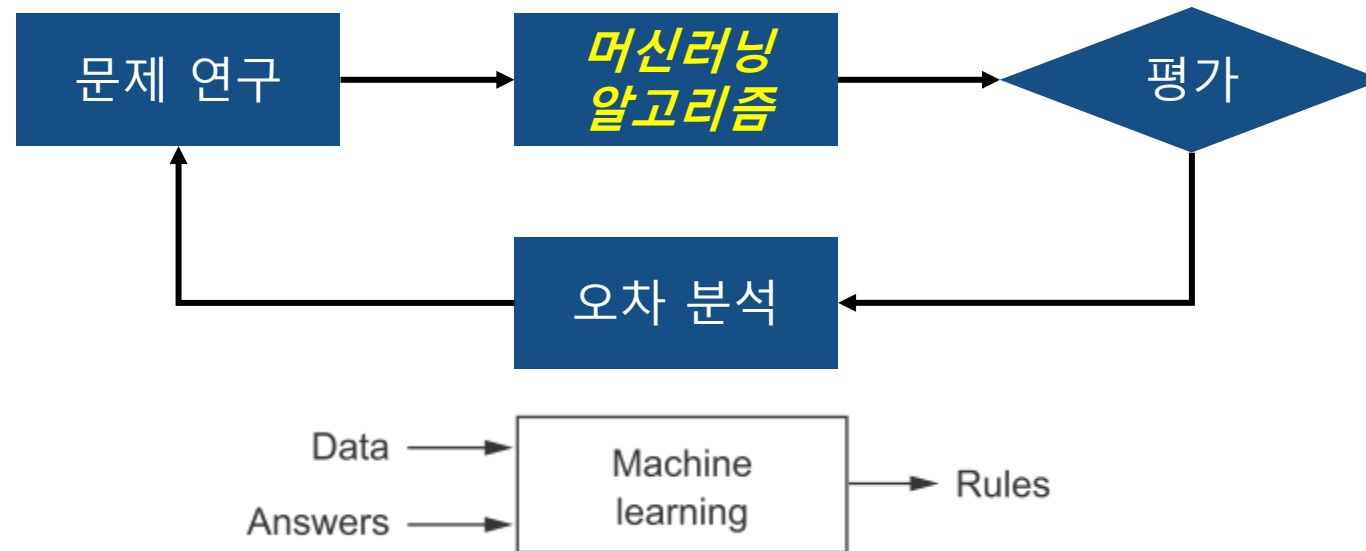
전통적인 문제 해결 방법

- 규칙을 사람(프로그래머)이 직접 찾아내어 작성
 - 규칙이 점점 길고 복잡해짐
 - 유지보수 매우 힘들
 - 변화에 능동적으로 대처하기 어려움



머신러닝의 문제 해결 방법

- 주어진 데이터와 해답으로부터 훈련을 통해 규칙을 생성
 - 규칙을 알 수 없거나 찾기 힘든 문제 해결 가능
 - 훈련, 평가, 론칭하는 전체 과정을 쉽게 자동화 가능



머신러닝 시스템의 종류

- 지도 학습 supervised learning
 - 훈련 데이터와 그에 상응하는 레이블 또는 타겟 포함
- 지도 학습의 종류
 - 분류 classification
 - 데이터가 특정 클래스에 속하는지 예측
 - 회귀 regression
 - 특징 features를 사용해 타겟 수치를 예측

머신러닝 시스템의 종류

▪ 지도 학습의 예

- 편지 봉투에 손으로 쓴 우편번호 숫자 판별
 - 입력: 손글씨 스캔 이미지
 - 기대 출력: 우편번호 숫자
- 의료 영상 이미지에 기반한 종양 판단
 - 입력: 영상 이미지
 - 기대 출력: 종양이 양성인지 여부
- 의심되는 신용카드 거래 감지
 - 입력: 신용카드 거래 내역
 - 기대 출력: 부정 거래인지 여부
- 특정 지역의 부동산 가격 예측
 - 입력: 부동산 거래 내역
 - 기대 출력: 부동산 가격

머신러닝 시스템의 종류

- 대표적인 지도 학습 알고리즘(모델)
 - k-최근접 이웃 (k-Nearest Neighborhood)
 - 선형 회귀(Linear Regression)
 - 로지스틱 회귀(Logistic Regression)
 - 서포트 벡터 머신(SVM, Support Vector Machine)
 - 결정 트리(Decision Tree)와 랜덤 포레스트(Random Forest)
 - 신경망(Neural Networks)

머신러닝 시스템의 종류

- 비지도 학습unsupervised learning

- 훈련 데이터에 그에 상응하는 레이블이 없음

- 비지도 학습의 종류

- 군집clustering

- 훈련 데이터들을 비슷한 그룹으로 묶음
- k-평균(k-Means) 클러스터링, 병합 클러스터링, DBSCAN 등

- 차원 축소, 시각화

- 레이블 없는 고차원 데이터를 도식화 가능토록 2차원 또는 3차원 데이터로 변환
- 너무 많은 정보를 잃지 않으면서 데이터를 간소화
- 주성분 분석(PCA, Principal Component Analysis), 비음수 행렬 분해 등

- 이상치 탐지

- 정상 샘플로 훈련된 시스템이 새로운 샘플이 정상 데이터인지 이상치인지 판단

머신러닝 시스템의 종류

■ 비지도 학습의 예

- 블로그 글의 주제 구분
 - 사전에 어떤 주제인지 알지 못하고 얼마나 많은 주제가 있는지 모름
 - 기대 출력을 준비할 수 없음
- 고객들을 취향이 비슷한 그룹으로 묶기
 - 어떤 취향의 그룹이 있는지 얼마나 많은 그룹이 있는지 알 수 없음
- 비정상적인 웹사이트 접근 탐지
 - 비정상 패턴은 서로 많이 다를 수 있고
 - 이미 확보한 비정상 패턴이 아닌 새로운 유형의 비정상 패턴이 앞으로 얼마든지 생길 수 있음
 - 단지 웹 트래픽만으로 정상/비정상 여부를 판별해야 함
- 고객들이 함께 구매하는 물품 찾기
 - “기저귀를 구매한 사람이 맥주도 많이 구매한다.”
 - 수 많은 물품들 중에서 서로 관련 있는 물품을 찾으면 마케팅에 유리하게 활용할 수 있음

머신러닝 시스템의 종류

■ 강화 학습 reinforcement learning

- 학습하는 시스템인 에이전트 agent가 환경을 관찰하여 행동을 실행하고 그 결과로 보상 reward 또는 벌점 penalty를 받으며 시간이 지나면서 가장 큰 보상을 얻기 위해 정책 policy이라는 최상의 전략을 스스로 학습
- 정책은 주어진 상황에서 에이전트가 어떤 행동을 선택해야 할지 정의
- 다른 머신러닝 시스템과의 차이점
 - No supervisor, 레이블 대신 보상 신호
 - 피드백이 즉각적이지 않고 딜레이되어 들어옴
 - 시간 혹은 스텝이 중요: 시퀀셜 데이터
 - 에이전트의 행동이 뒤이어 발생하는 데이터(신호)에 영향을 미침

머신러닝 시스템의 종류

- 강화 학습의 예
 - 무선 헬리콥터 스텐트 조종 비행
 - 투자/주식 포트폴리오 관리
 - 발전소 제어
 - 휴머노이드 로봇 걷기
 - 인간보다 컴퓨터 게임을 더 잘하는 시스템 개발
 - 자동 주차 시스템
 - 알파고

머신러닝으로 문제 해결하기 전에

■ 문제와 데이터 이해하기

- 어떤 질문에 대한 답을 원하는가?
- 가지고 있는 데이터가 원하는 답을 줄 수 있는가?
- 내 질문을 머신러닝의 문제로 가장 잘 기술하는 방법은 무엇인가?
- 문제를 풀기에 충분한 데이터를 모았는가?
- 내가 추출한 데이터의 특성은 무엇이며 좋은 예측을 만들어낼 수 있을 것인가?
- 머신러닝 애플리케이션의 성과를 어떻게 측정할 수 있는가?
- 머신러닝 솔루션이 다른 연구나 제품과 어떻게 협력할 수 있는가?

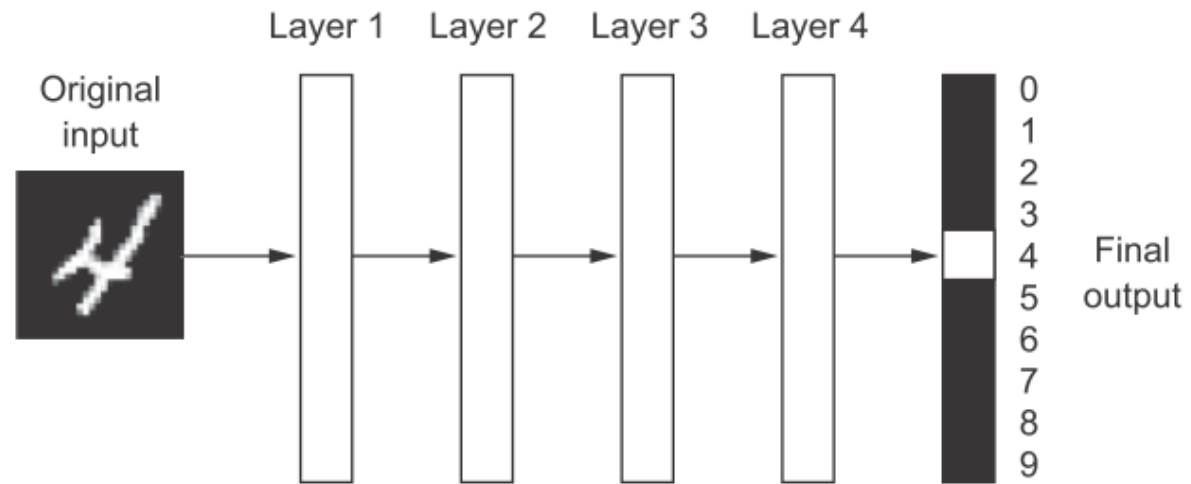
딥러닝

■ 딥러닝이란?

- 머신러닝의 한 분야인 신경망의 한 종류인 심층신경망에서의 학습 메커니즘
- 연속된 층(layer)에서 점진적으로 의미 있는 표현을 배우는데 강점
- 데이터로부터 모델을 만드는데 얼마나 많은 층을 사용했는지가 그 모델의 깊이(depth)
→ 딥러닝이라 불리는 이유
- 일반 머신러닝 모델로 해결하기 어려운 분야에 활용
- 이미지 인식, 동영상 인식, 음성 인식, 사진 검색
- 언어 번역, 자연어 처리
- 기사 작성, 작곡, 이미지 합성, 딥페이크

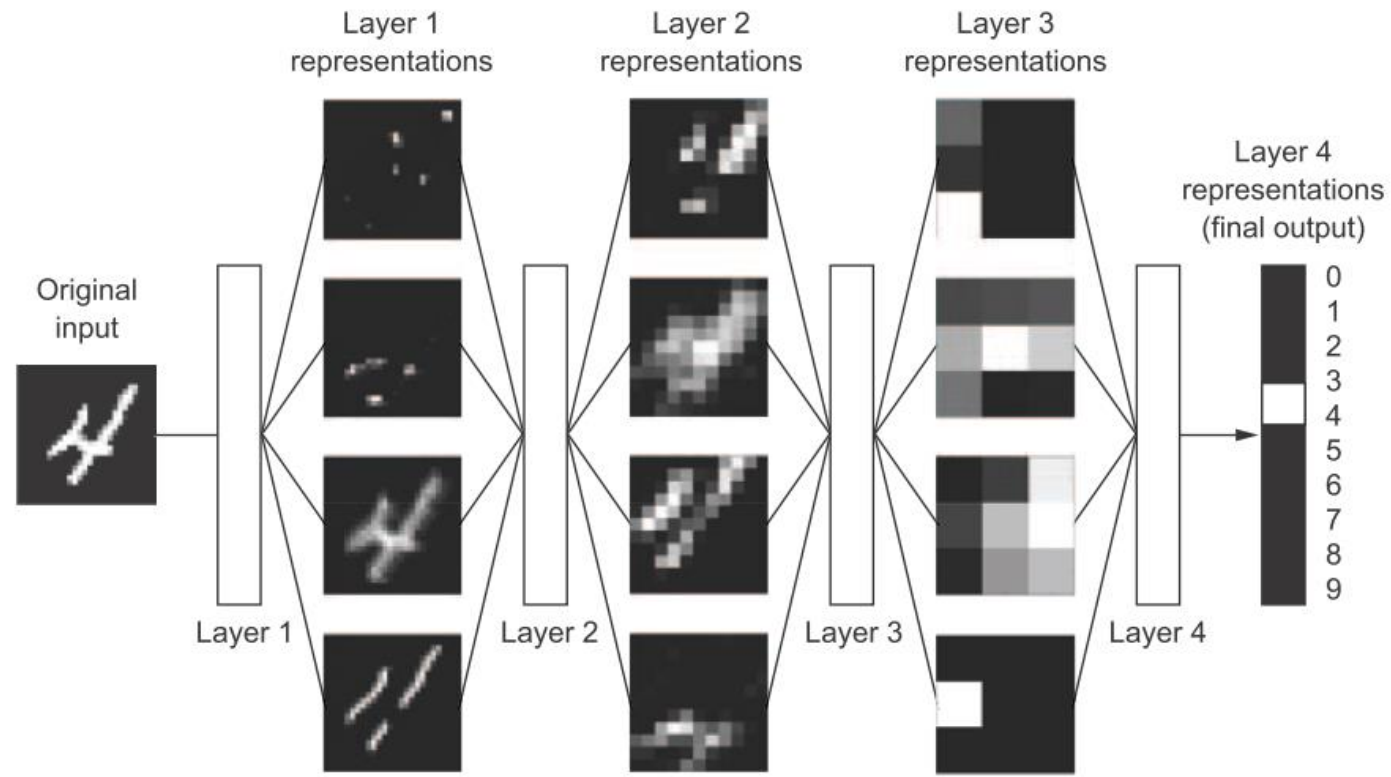
딥러닝

- 손글씨 숫자 분류를 위한 심층신경망 구조



딥러닝

- 손글씨 숫자 분류를 위한 심층신경망의 학습 이후 내부 모습



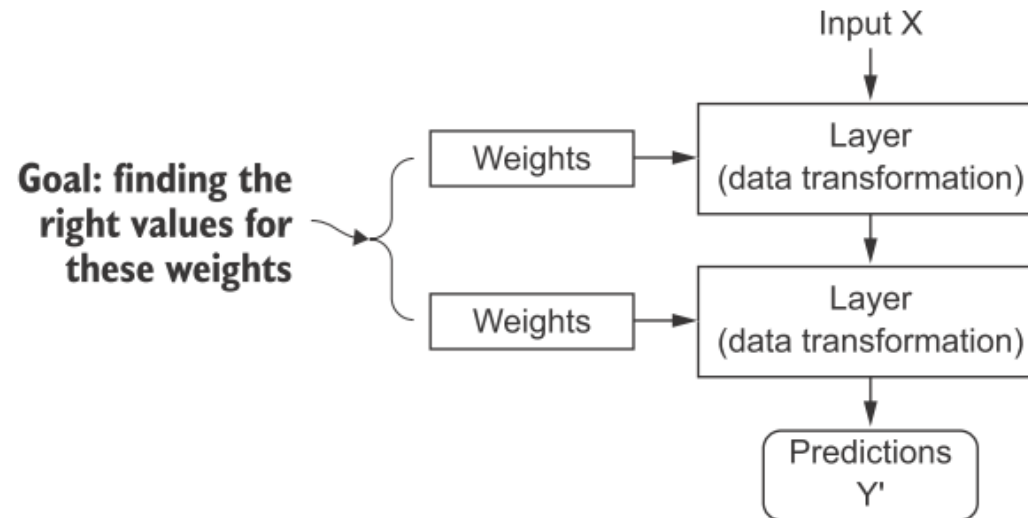
딥러닝 작동 원리

- 가중치

- 입력 데이터는 가중치와 곱해진 후 다음 층으로 출력

- 학습

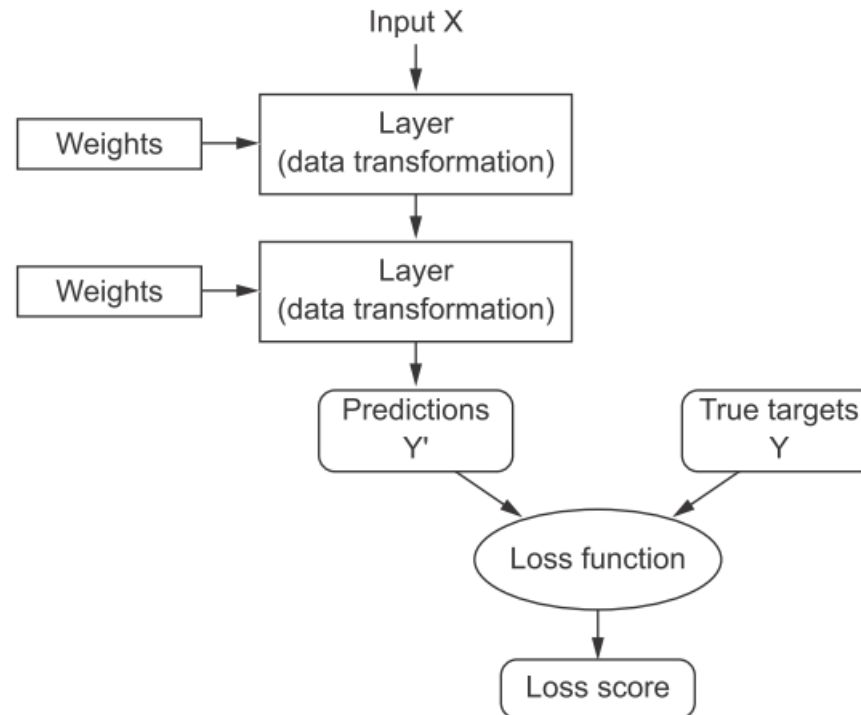
- 주어진 입력을 정확한 타겟에 매핑하기 위해 신경망의 모든 층에 있는 가중치를 찾는 것



딥러닝 작동 원리

- 손실 함수(비용함수, 목적 함수)

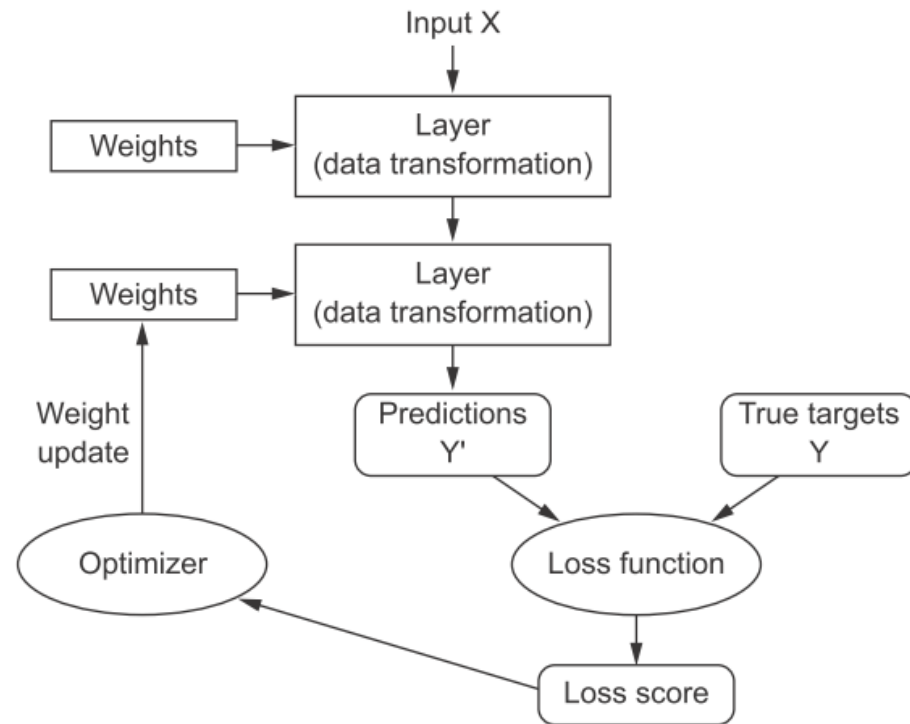
- 층의 출력이 기대하는 것보다 얼마나 벗어났는지를 측정
- 예측과 실제 타겟의 차이를 점수로 계산
- 손실 함수에 대한 미분



딥러닝 작동 원리

■ 옵티마이저

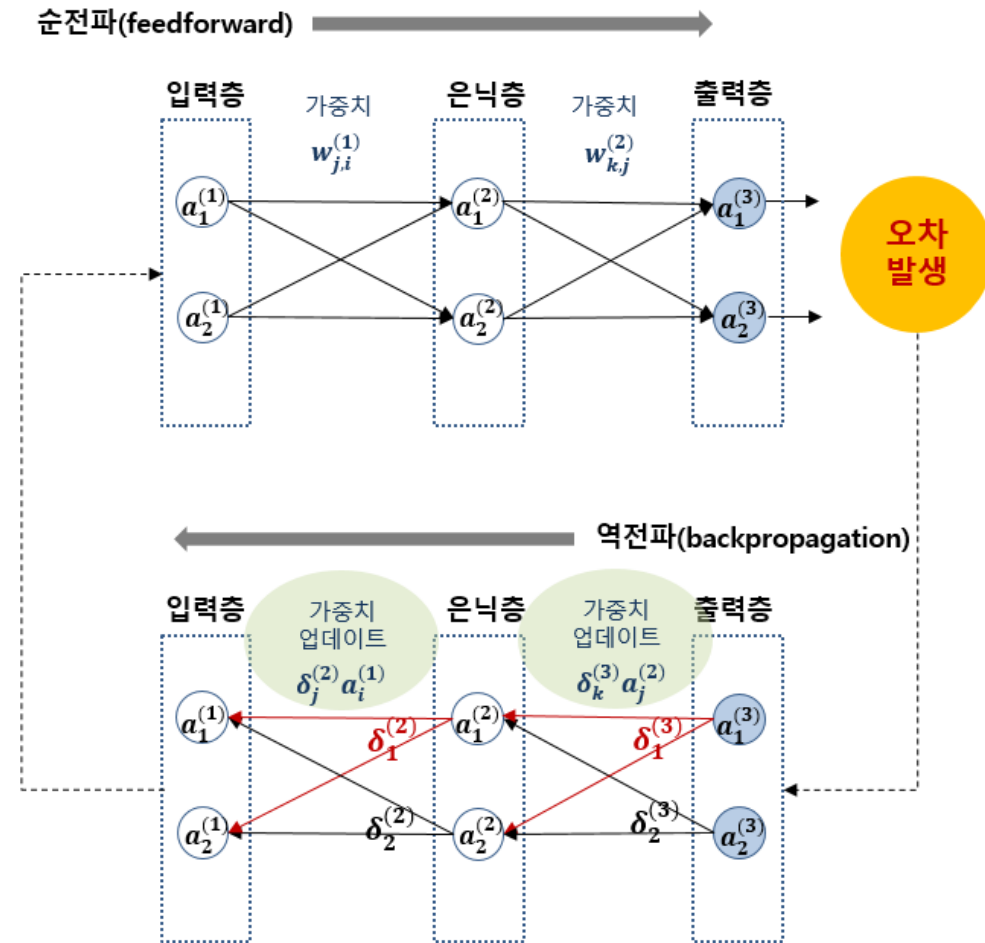
- 앞서 구한 점수를 피드백 신호로 사용하여 현재 샘플이 손실 점수가 감소되는 방향으로 가중치 값을 조금씩 수정



딥러닝 작동 원리

■ 역전파 알고리즘

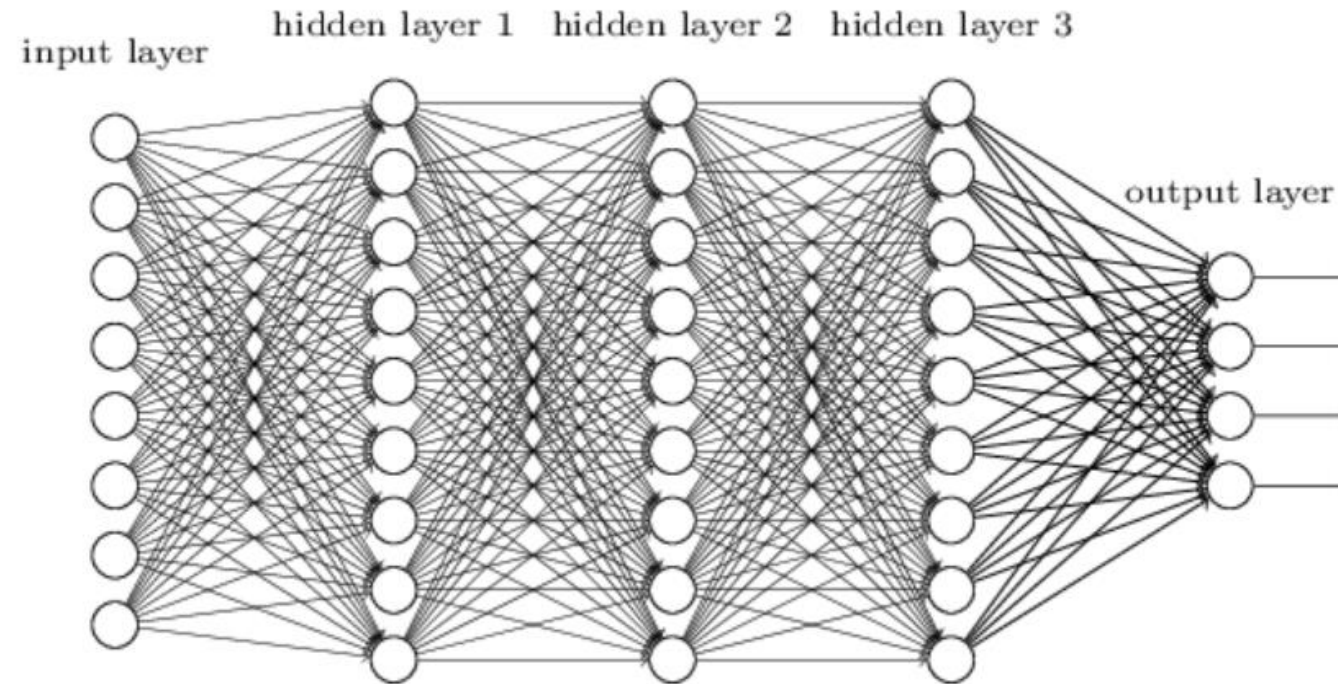
- 해당 층 출력의 오차에 대한 미분을 이전 층으로 넘기면서 최적의 가중치를 구하는 방법



[이미지 출처] <https://blog.naver.com/samsjang/221033626685>

딥러닝 모델

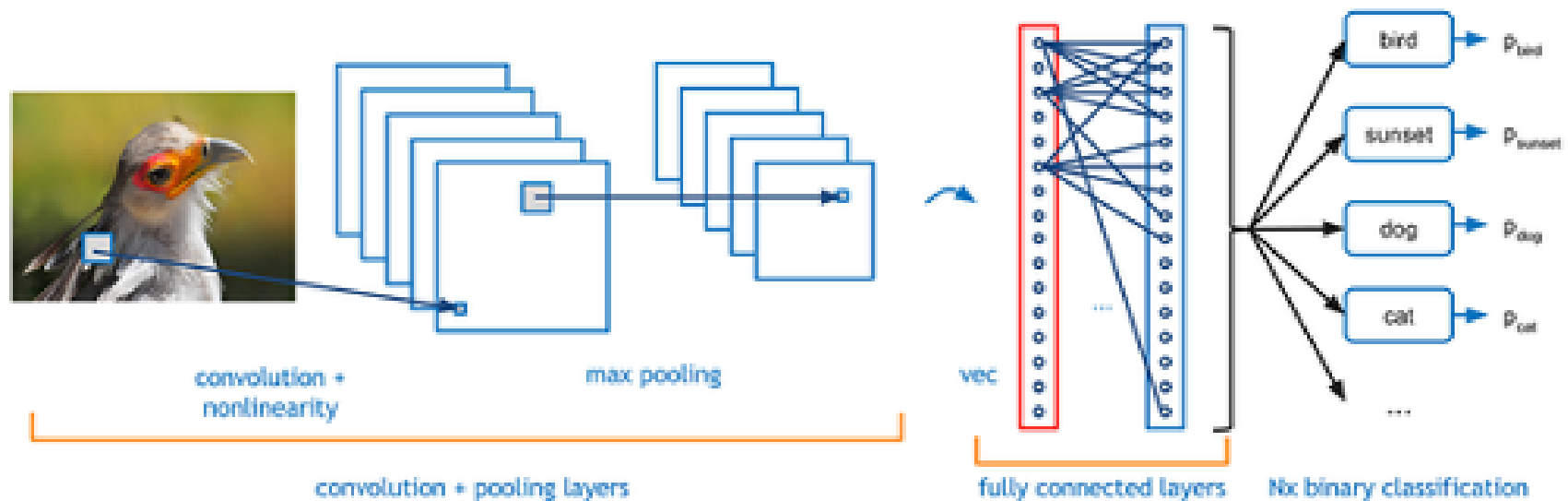
- 다층 퍼셉트론(MLP, Multilayer Perceptron)
 - 일반적인 머신러닝 문제 해결



딥러닝 모델

- 합성곱 신경망(CNN, Convolutionary Neural Network)

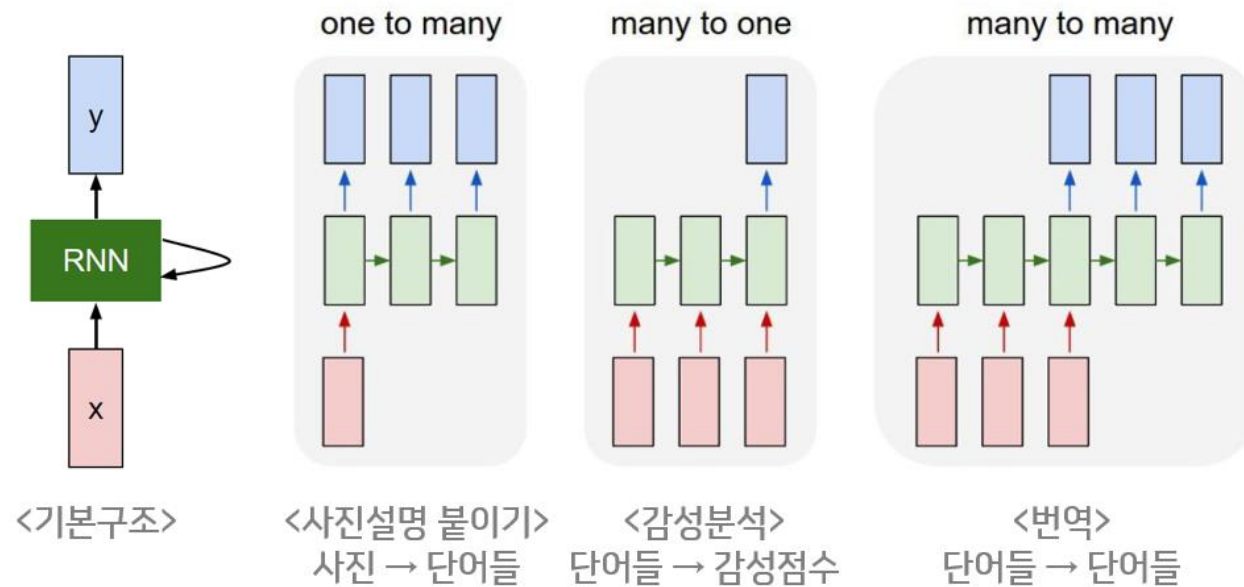
- 이미지 및 동영상 같은 다차원 데이터를 분류, 분할, 생성에 탁월



딥러닝 모델

- 순환 신경망(RNN, Recurrent Neural Network)

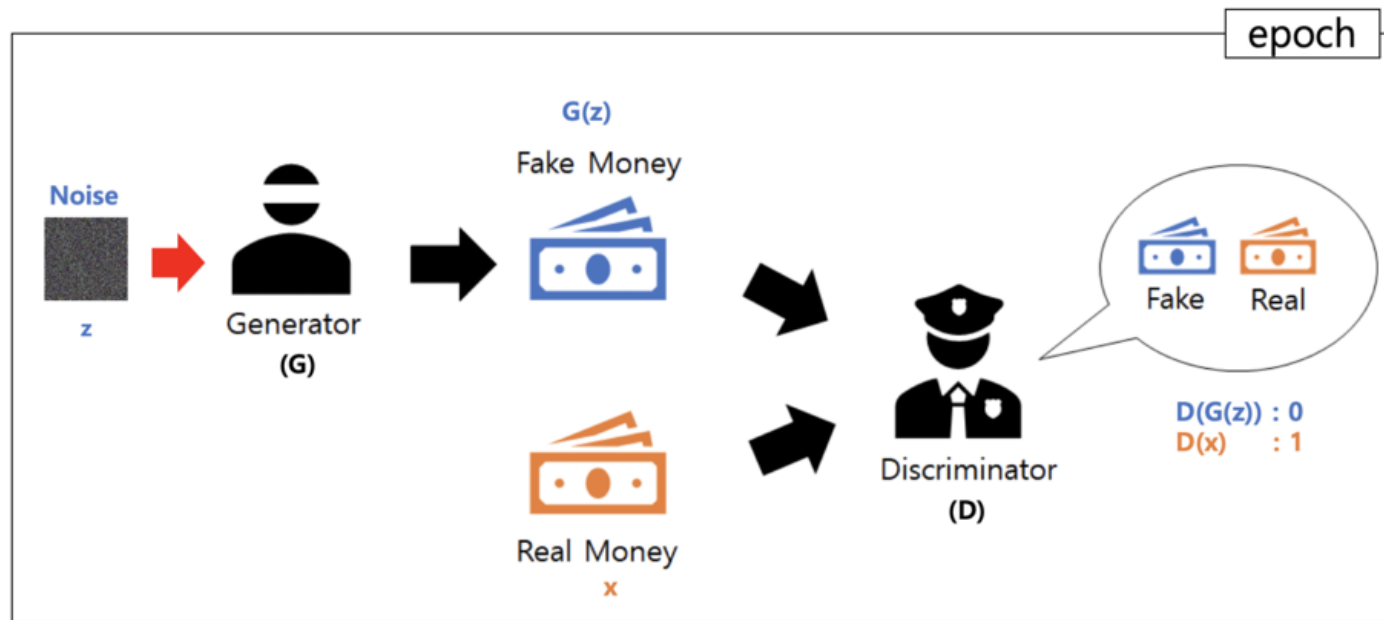
- 자연어 처리(NLP)에서 텍스트나 기기 장치의 센서 데이터 스트림과 같은 순차적 데이터의 표현을 학습하기에 적합



[이미지 출처] <https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/>

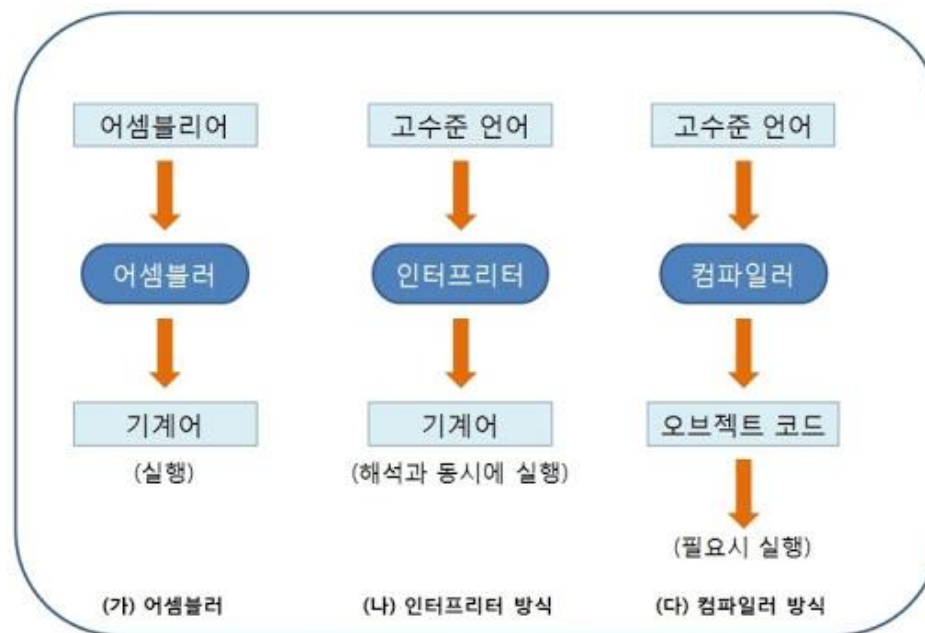
딥러닝 모델

- 생성적 적대 신경망(GAN, Generative Adversarial Network)
 - 생성자 네트워크: 잡음으로부터 원본 데이터셋에서 샘플링한 것처럼 보이는 샘플 생성
 - 판별자 네트워크: 원본 데이터셋에서 추출한 샘플인지 생성자가 만든 가짜인지 구별
 - 두 네트워크 간의 (적대적) 경쟁을 통해 학습



[이미지 출처] <https://blog.naver.com/euleekwon/221557899873>

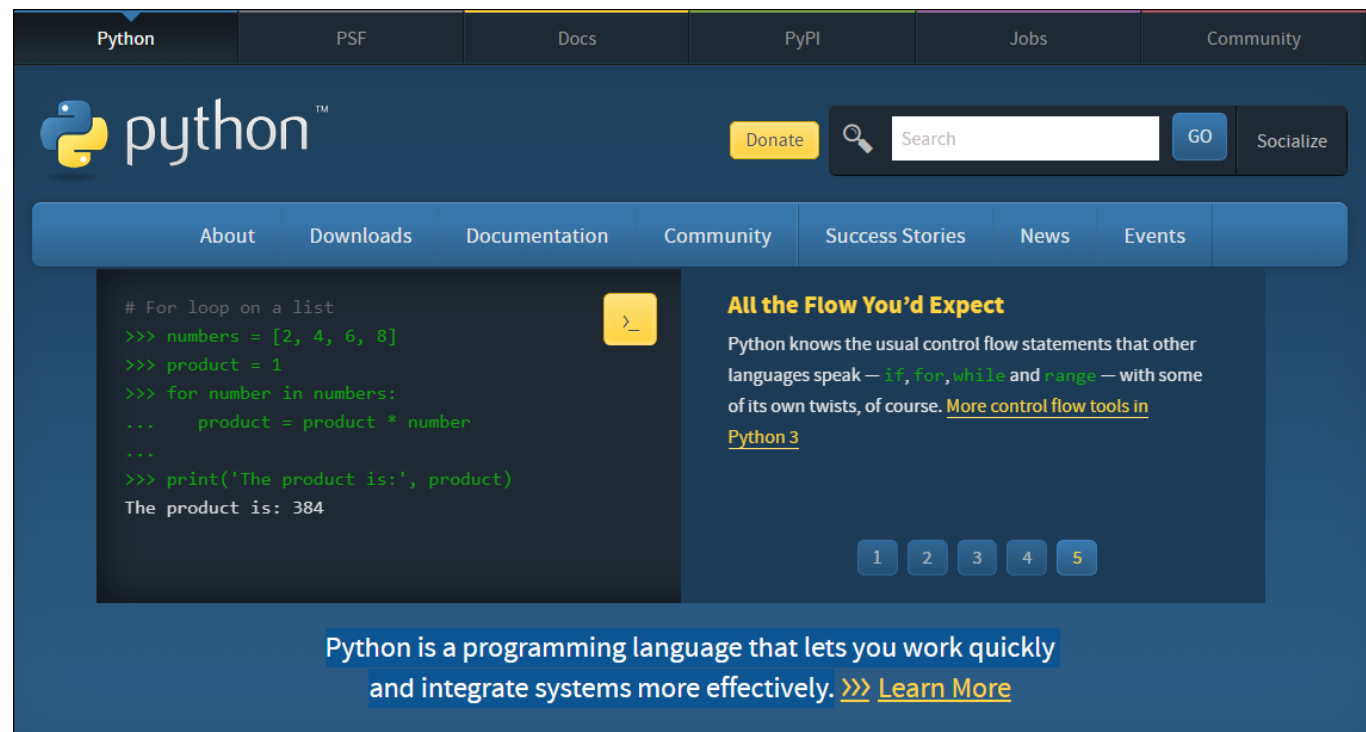
- 1990년 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발한 인터프리터 언어
- 인터프리터 언어 vs. 컴파일러 언어



파이썬이란



- <https://www.python.org>
 - Python is a programming language that lets you work quickly and integrate systems more effectively.



파이썬의 특징

- **문법이 쉬워 빠르게 배울 수 있다.**
 - 유명한 프로그래머인 에릭 레이먼드Eric Raymond는 파이썬을 공부한 지 단 하루 만에 자신이 원하는 프로그램을 작성할 수 있었다고 함
- **무료이면서 강력하다.**
 - 오픈 소스, 글루^{Glue} 언어
- **간결하다.**
 - 파이썬 코딩 스타일을 지키면 다른 사람이 작업한 소스 코드도 이해하기 쉽기 때문에 공동 작업과 유지 보수가 쉽고 편함
- **개발 속도가 빠르다.**
 - *"Life is too short, You need python."*

파이썬으로 할 수 있는 일

- 시스템 유틸리티
- GUI 프로그래밍
- C/C++와의 결합
- 웹 프로그래밍
- 데이터베이스 프로그래밍
- 수치 연산 프로그래밍
- 빅데이터 수집, 저장, 분석
- 머신러닝, 딥러닝 프로그래밍
- 사물 인터넷
- 그 외 다양한 분야

파이썬으로 하기 힘든 일

- 대단히 빠른 속도를 요하는 프로그램
- 운영체제를 건드리는 프로그램
- 하드웨어를 깊숙이 다루어야 하는 프로그램
- 모바일 프로그래밍

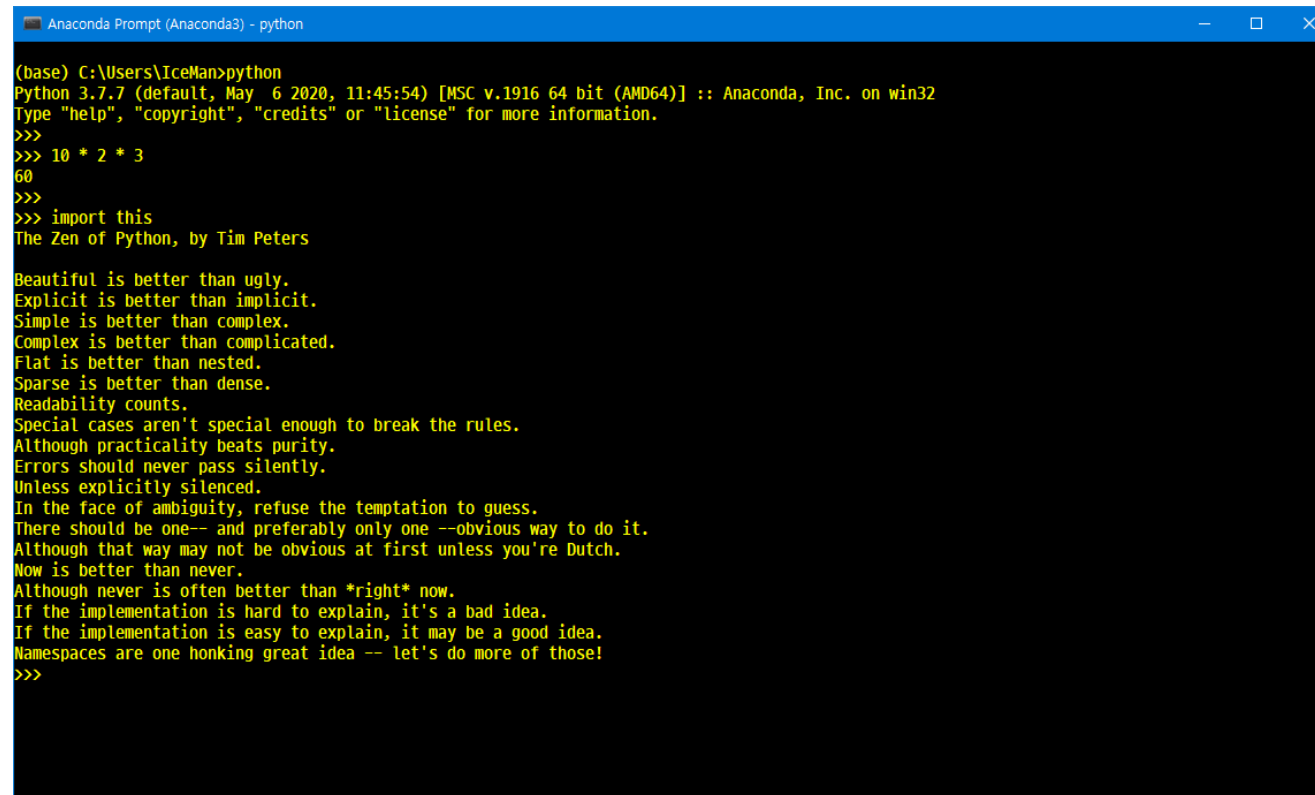
파이썬 설치

- <https://www.python.org/downloads/>



파이썬 개발 환경

■ Python Shell

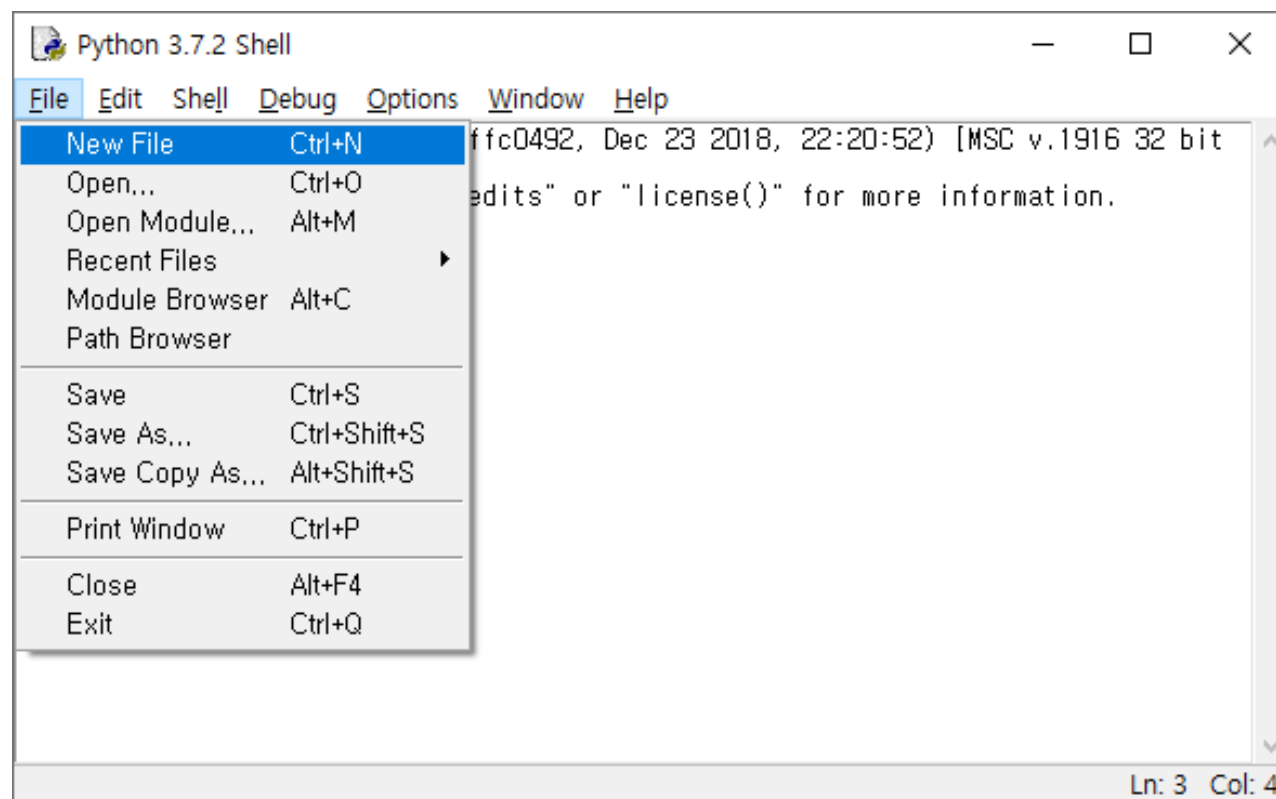


```
Anaconda Prompt (Anaconda3) - python
(base) C:\Users\IceMan>python
Python 3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> 10 * 2 * 3
60
>>>
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

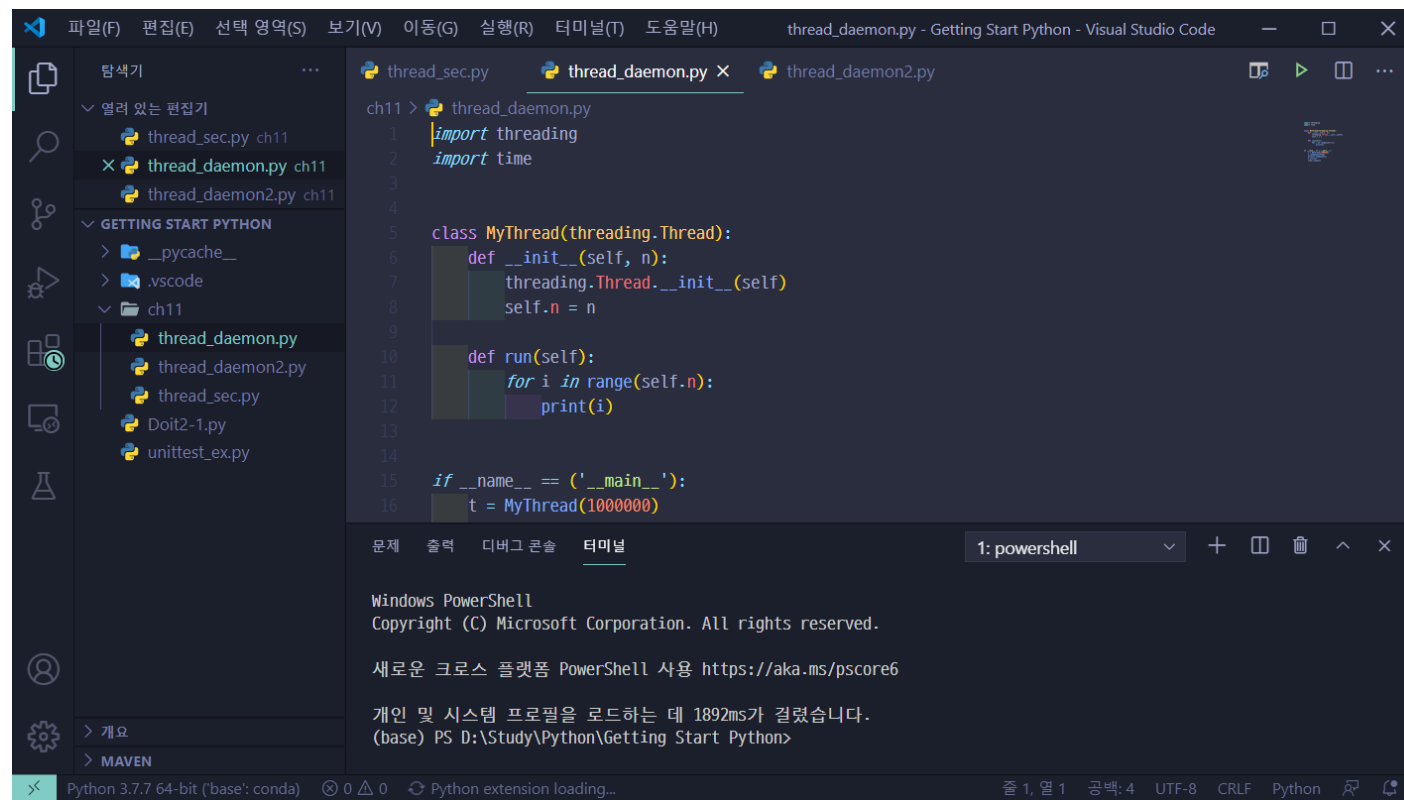
파이썬 개발 환경

■ Python Shell IDLE



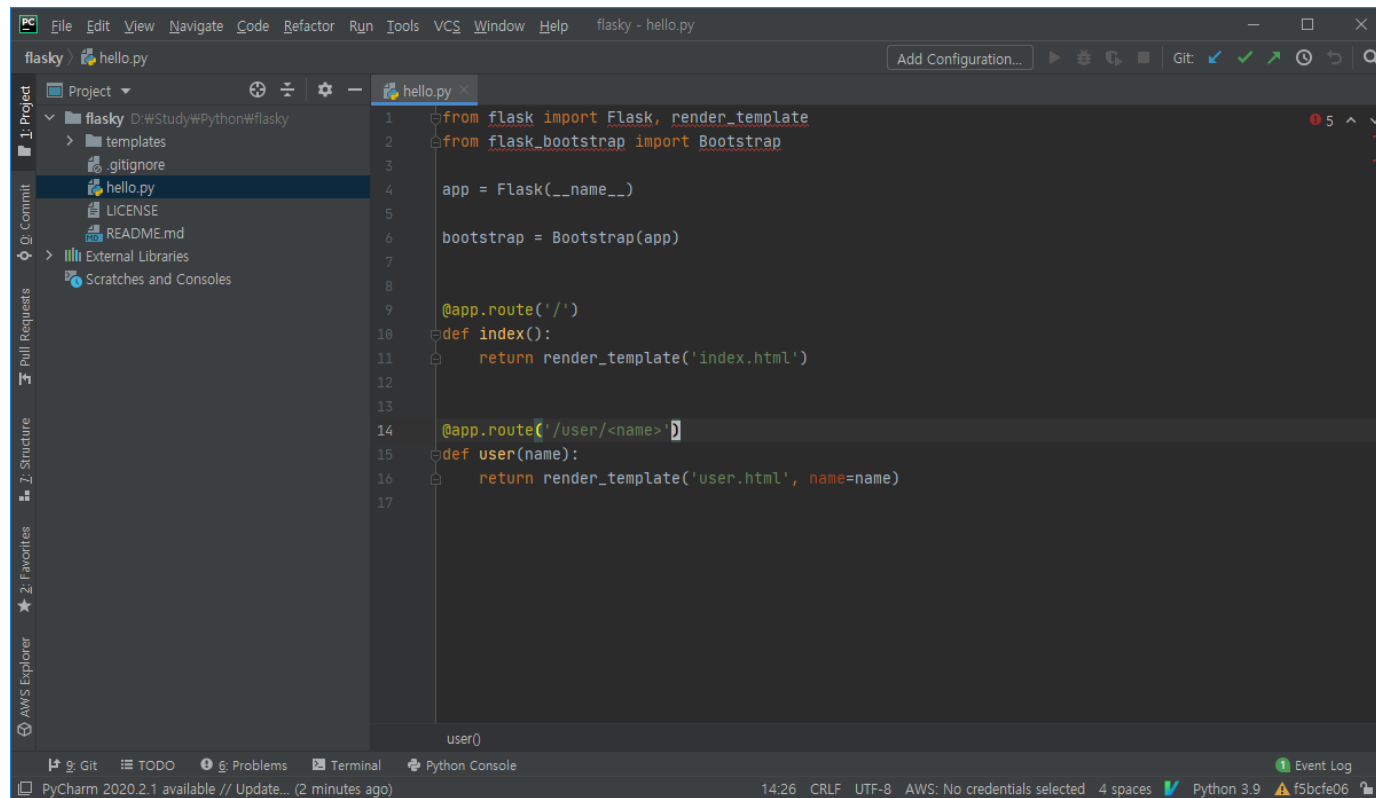
파이썬 개발 환경

■ Visual Studio Code



파이썬 개발 환경

■ PyCharm



파이썬 개발 환경

■ Jupyter Notebook, Jupyter Lab

The image displays three overlapping screenshots of Python development environments:

- Left Screenshot (Jupyter Notebook):** Shows a notebook titled "ch05 (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The main content area has a heading "Getting Started with pandas" and a code cell with the following Python code:

```
In [ ]: import pandas as pd

In [ ]: from pandas import Series, DataFrame

In [ ]: import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
PREVIOUS_MAX_ROWS = pd.options.display.max_rows
pd.options.display.max_rows = 20
np.set_printoptions(precision=4, suppress=True)
```

Below this, there is a section titled "Introduction to pandas Data Structures" and "Series" with another code cell:

```
In [ ]: obj = pd.Series([4, 7, -5, 3])
obj

In [ ]: obj.values
obj.index # like range(4)
```
- Middle Screenshot (Jupyter Lab File Explorer):** Shows a file explorer window in Jupyter Lab. The path is "/ deep-learning-with-python-notebooks-master /". It lists various files and folders with their last modified dates. The file "11. 손글씨 분류 파이썬 프로그램" is highlighted.
- Right Screenshot (Jupyter Notebook):** Shows a notebook titled "11. 손글씨 분류 파이썬 프로그램". The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar. The main content area has a heading "11. 손글씨 분류 파이썬 프로그램" and a code cell with the following Python code:

```
[ ]: %pwd

[ ]: %ls *.png

[ ]: from keras.preprocessing import image
```

Below this, there is a section titled "케라스로 이미지 불러들이기" and a code cell with the following Python code:

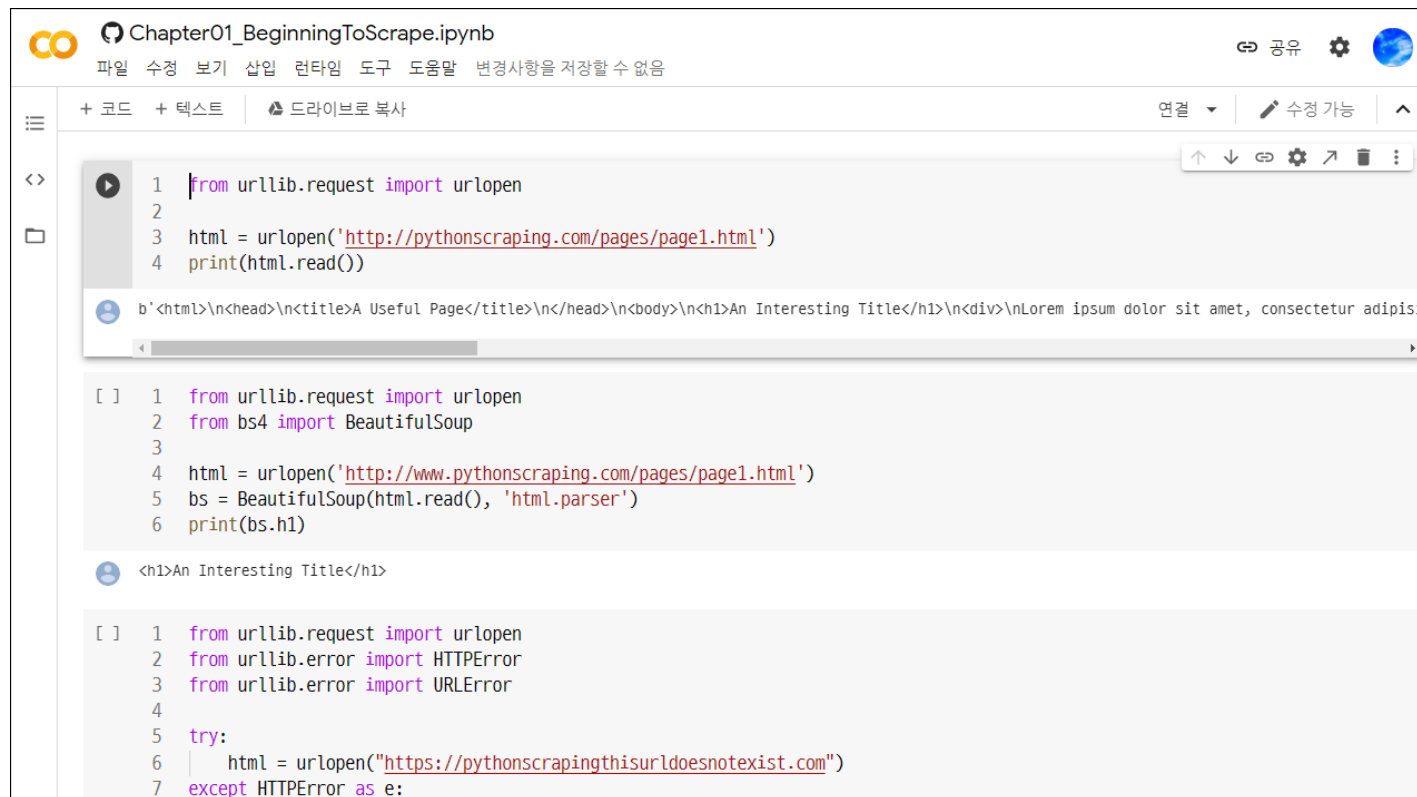
```
conda install -n deeplearning pillow

[ ]: hand_writing_number = image.load_img("./number5-02.png", color_mode="grayscale",
target_size=(28, 28), interpolation='bilinear')
```

파이썬 개발 환경

- Google Colaboratory

- <https://colab.research.google.com/>



```
Chapter01_BeginningToScrape.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 변경사항을 저장할 수 없음

+ 코드 + 텍스트 드라이브로 복사
연결 수정 가능

1 from urllib.request import urlopen
2
3 html = urlopen('http://pythonscraping.com/pages/page1.html')
4 print(html.read())

b'<html>\n<head>\n<title>A Useful Page</title>\n</head>\n<body>\n<h1>An Interesting Title</h1>\n<div>\nLorem ipsum dolor sit amet, consectetur adipisi

[ ] 1 from urllib.request import urlopen
2 from bs4 import BeautifulSoup
3
4 html = urlopen('http://www.pythonscraping.com/pages/page1.html')
5 bs = BeautifulSoup(html.read(), 'html.parser')
6 print(bs.h1)

<h1>An Interesting Title</h1>

[ ] 1 from urllib.request import urlopen
2 from urllib.error import HTTPError
3 from urllib.error import URLError
4
5 try:
6     html = urlopen("https://pythonscrapingthisurldoesnotexist.com")
7 except HTTPError as e:
```


Google Colaboratory

- **구글 클라우드로 제공되는 Jupyter Notebook 환경**
 - 브라우저에서 Python을 작성하고 실행
 - Python 설치 필요 없음
 - GPU 무료 접근
 - 간편한 공유
 - 구글 계정만 있으면 누구나 사용 가능
 - 구글 드라이브 및 GitHub과 연동 가능
- **Colaboratory에 오신 것을 환영합니다.**
 - <https://colab.sandbox.google.com/notebooks/welcome.ipynb?hl=ko>

Overview of Colaboratory Features

■ Cells

- 셀은 텍스트 및 코드와 코드의 실행 결과를 담는 노트북의 구조
- 코드 셀
 - Play icon, Cmd/Ctrl+Enter, Shift+Enter, Alt-Enter
- 텍스트 셀
 - 더블 클릭, Enter
 - 마크다운^{Markdown}, LaTeX
- 셀 선택, 추가 및 이동

Overview of Colaboratory Features

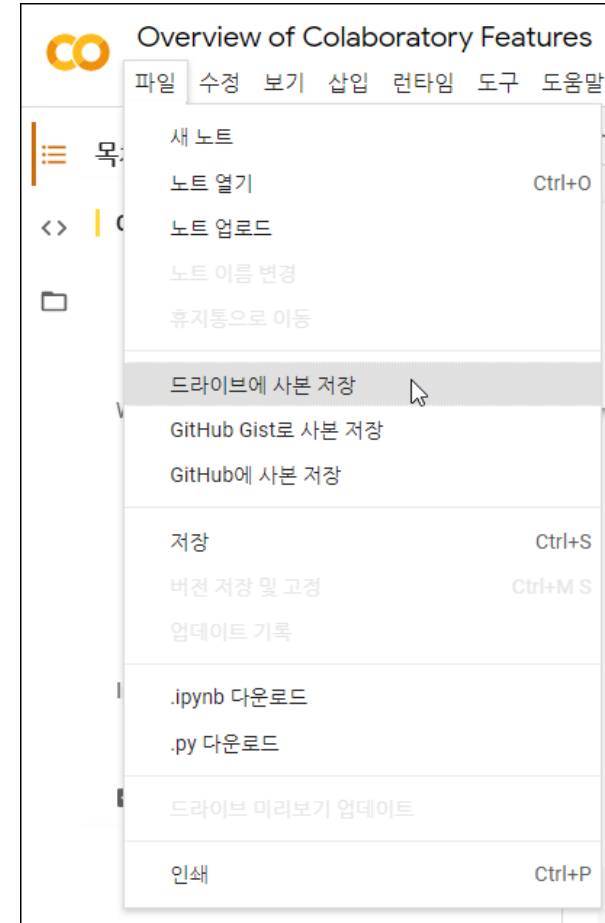
■ Working with Python

- 시스템 에일리어스 System aliases
- 매직 Magics
 - %lsmagic
- 탭 완성: Tab 또는 Ctrl+Space
 - 설정에서 자동 완성 설정이 켜져 있는 경우에는 Ctrl+Space만 가능
 - 자기관찰 Introspection
- 예외 포매팅
- 풍부하면서 대화식적인 출력

Overview of Colaboratory Features

■ Integration with Google Drive

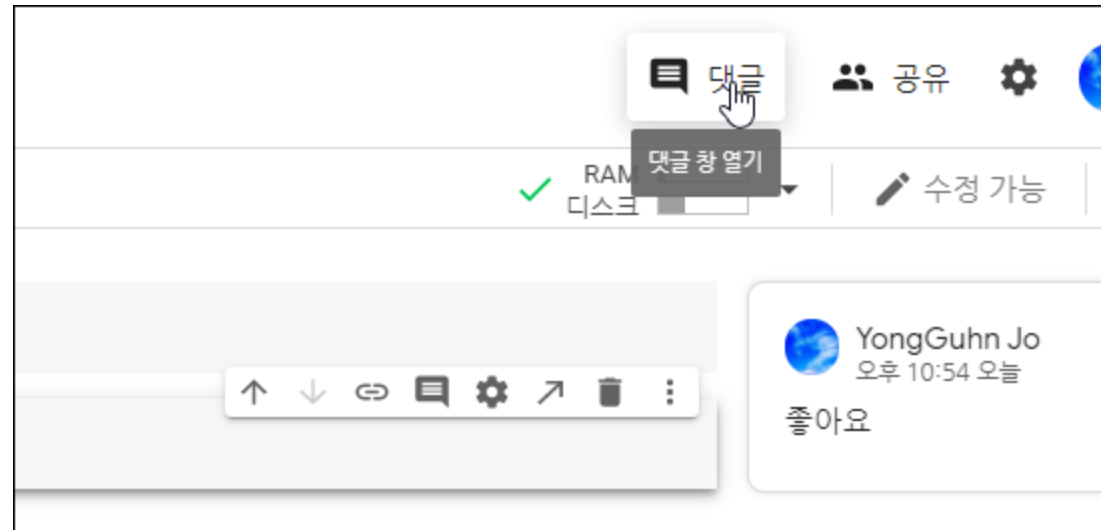
- 공유 버튼
- 파일 -> 드라이브에 사본 저장
- 파일 -> 저장
- 파일 -> 버전 저장 및 고정
- 파일 -> 업데이트 기록



Overview of Colaboratory Features

- **Commenting on a cell**

- 댓글 버튼
- 댓글 달 권한 필요
- 댓글 창 열기



Colaboratory Markdown Guide

- **마크다운이란**

- 웹 저작자들을 위한 text-to-HTML 변환 툴

- Quick reference

- Headers
 - Bold, Italic
 - Indentation of a block
 - Ordered/Unordered list
 - Links, Images
 - LaTeX
 - Equations, Tables

- Examples

Colaboratory

- 콜랩에 설치되어 있지 않은 라이브러리/패키지 가져오기

- 파이썬 패키지 설치

```
!pip install matplotlib-venn
```

- 리눅스 패키지 설치

```
!apt-get -qq install -y libfluidsynth1
```

Colaboratory

- PC에서 파일을 선택하여 콜랩으로 업로드

- 구글 콜랩 패키지 импорт

```
from google.colab import files  
files.upload()
```

- 확인

```
!ls
```


Colaboratory

- 파일을 만들고 PC로 다운로드

```
import pandas as pd  
df = pd.DataFrame([1, 2, 3])  
df
```

```
df.to_csv('new_file.txt')  
!cat new_file.txt
```

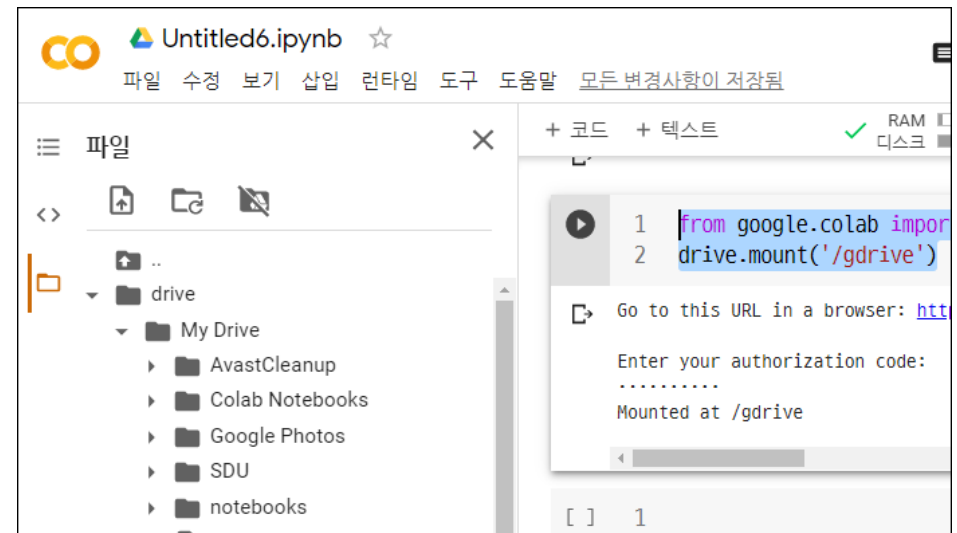
```
from google.colab import files  
files.download('new_file.txt')
```

Colaboratory

■ 구글 드라이브 마운트 하기

```
from google.colab import drive  
drive.mount('/gdrive')
```

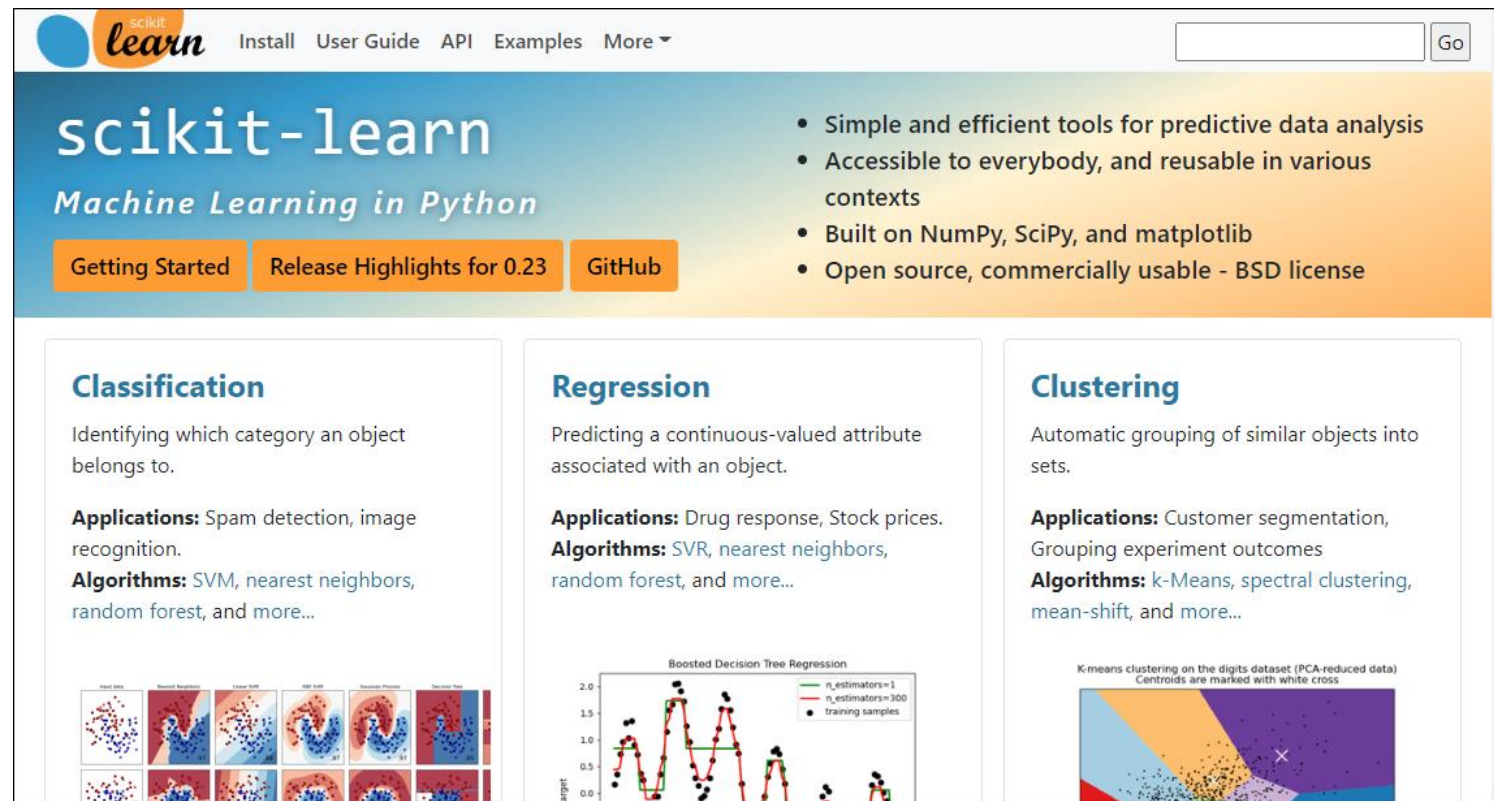
```
!ls /gdrive/My\ Drive/Colab\ Notebooks/data
```



scikit-learn

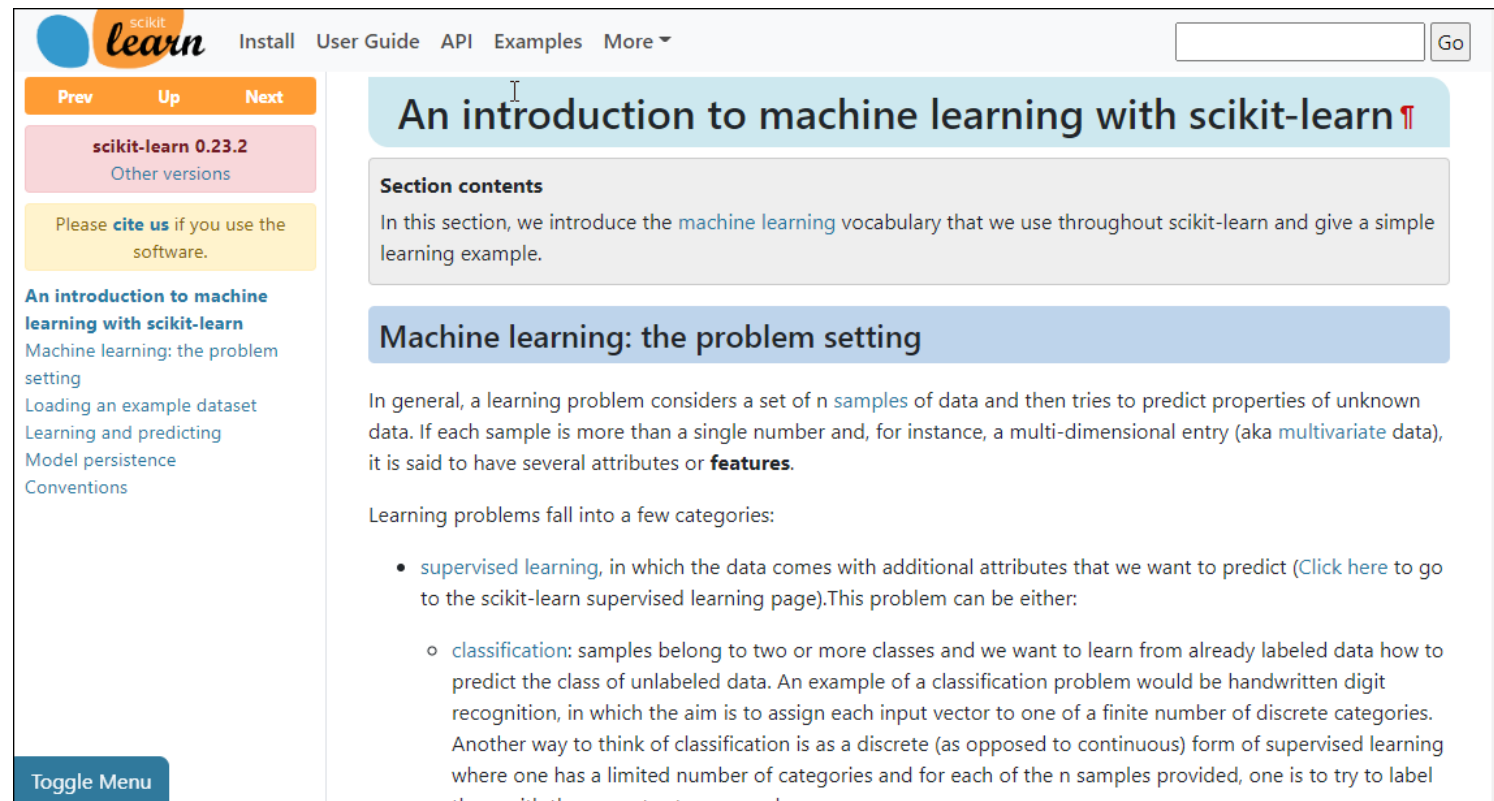
- (독보적인) 파이썬 머신러닝 라이브러리

- <https://scikit-learn.org>



scikit-learn

- An introduction to machine learning with scikit-learn
 - <https://scikit-learn.org/stable/tutorial/basic/tutorial.html>



The screenshot displays the scikit-learn website's tutorial page. The header includes the scikit-learn logo and navigation links: Install, User Guide, API, Examples, and More. A search bar with a 'Go' button is also present. The left sidebar contains a 'Toggle Menu' button and a list of tutorial topics: 'An introduction to machine learning with scikit-learn', 'Machine learning: the problem setting', 'Loading an example dataset', 'Learning and predicting', 'Model persistence', and 'Conventions'. The main content area features the title 'An introduction to machine learning with scikit-learn' with a red double exclamation mark icon. Below the title is a 'Section contents' box with the text: 'In this section, we introduce the machine learning vocabulary that we use throughout scikit-learn and give a simple learning example.' The main heading 'Machine learning: the problem setting' is followed by a paragraph explaining that a learning problem involves a set of n samples of data and aims to predict properties of unknown data. It defines multivariate data as having several attributes or features. A list of learning problem categories is provided, starting with supervised learning, which includes classification as a discrete form of supervised learning.

scikit-learn 필수 라이브러리

- NumPy: <https://numpy.org>

- 과학 계산용 필수 패키지
- NumPy 배열
 - scikit-learn의 기본 데이터 구조
 - ndarray: 다차원 배열
 - 배열의 모든 원소는 동일한 데이터 타입
- scikit-learn의 입력 데이터는 NumPy 배열 구조

- SciPy: <https://scipy.org>

- 수학, 과학 및 공학용 오픈소스 소프트웨어의 파이썬 기반 생태계
- 과학 계산용 함수를 모아놓은 파이썬 패키지
- 고성능 선형 대수, 함수 최적화, 신호 처리, 특수한 수학 함수, 확률 및 통계, 희소 행렬 등

scikit-learn 필수 라이브러리

- matplotlib: <https://matplotlib.org>

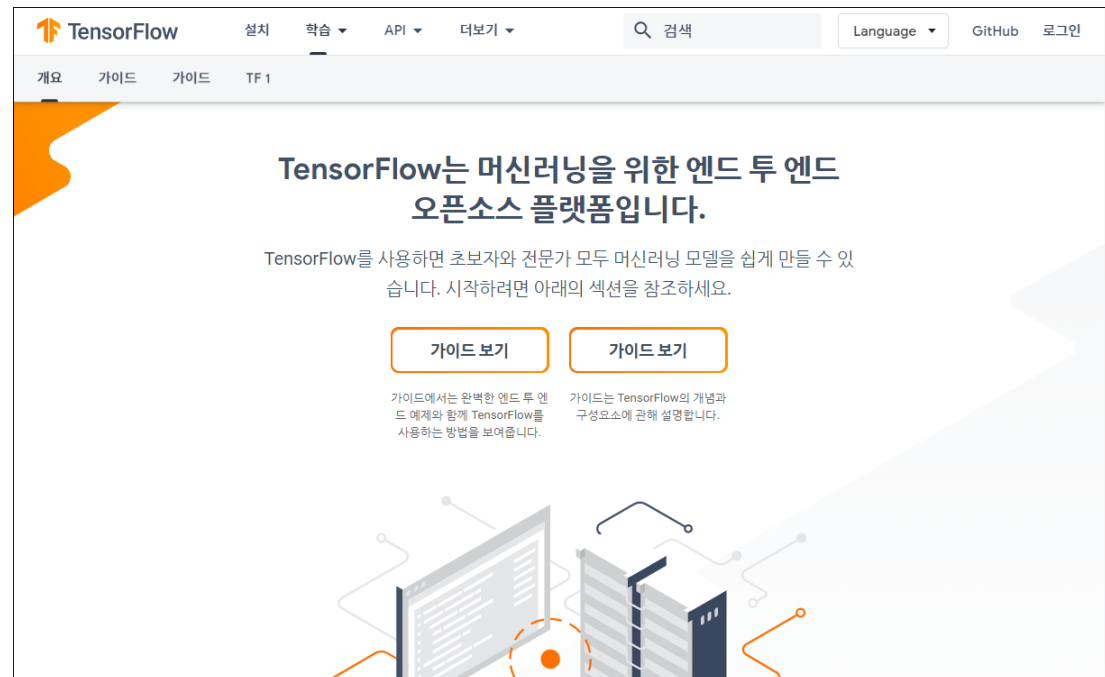
- 파이썬의 대표적인 과학 계산용 그래프 라이브러리
- 선 그래프, 막대그래프, 히스토그램, 산점도, 상자수염그래프
- 고품질 래스터 그래픽 지원
- 데이터와 분석 결과를 다양한 관점의 시각화를 통해 통찰력 얻는데 도움

- pandas: <https://pandas.pydata.org>

- 데이터 분석과 처리를 위한 빠르고, 강력하고, 유연하면서도 사용하기 쉬운 파이썬 기반 오픈 소스 라이브러리
- DataFrame 객체를 이용하여 엑셀의 스프레드 시트 및 SQL의 테이블 형태의 데이터를 쉽게 처리할 수 있음
- 다양한 데이터 소스(RDBMS, 엑셀, CSV 등)로부터 데이터를 읽어 들일 수 있음

TensorFlow

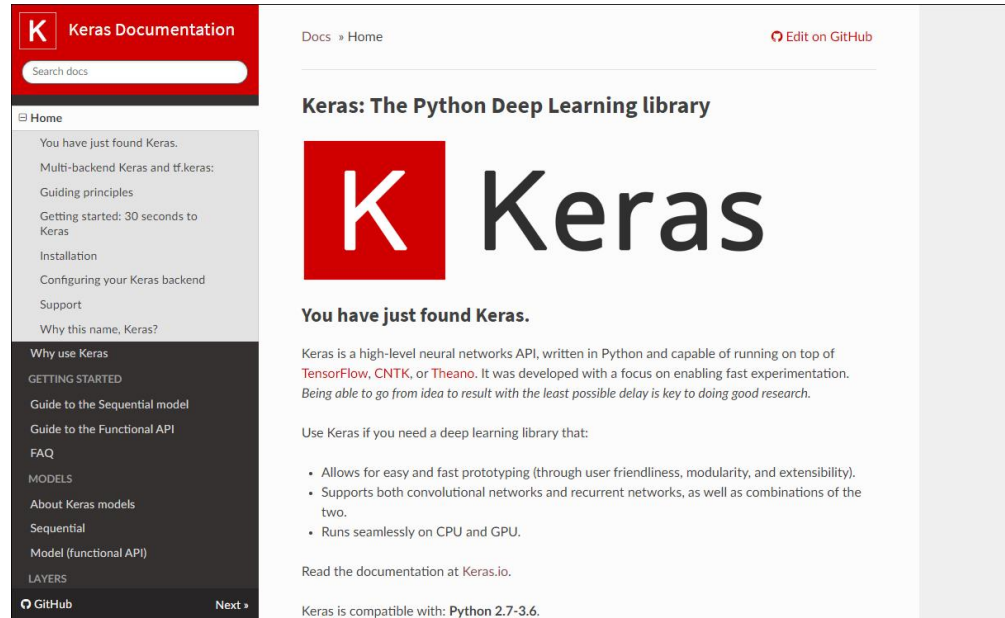
- TensorFlow: <https://www.tensorflow.org>
 - 구글이 개발하는 머신러닝을 위한 엔드 투 엔드 오픈소스 플랫폼
 - 데이터 흐름에 대한 그래프로써 사용자들이 마음대로 계산을 표현할 수 있게 하는 파이썬 라이브러리



Keras

- Keras: <https://keras.io/>

- 거의 모든 종류의 딥러닝 모델을 간편하게 만들고 훈련시킬 수 있는 파이썬을 위한 딥러닝 프레임워크
- 멀티백엔드(TensorFlow, CNTK, Theano) 기반 파이썬 고수준 신경망 API
- TensorFlow의 기본 라이브러리로 편입



TensorFlow vs. Keras

TensorFlow: XOR 문제 해결을 위한 퍼셉트론

```
import numpy as np
import tensorflow as tf

tf.set_random_seed(0)

# 데이터 준비
# XOR 게이트
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
Y = np.array([[0], [1], [1], [0]])

# 모델 생성
x = tf.placeholder(tf.float32, shape=[None, 2])
t = tf.placeholder(tf.float32, shape=[None, 1])

# 입력층-은닉층
W = tf.Variable(tf.truncated_normal([2, 2]))
b = tf.Variable(tf.zeros([2]))
h = tf.nn.sigmoid(tf.matmul(x, W) + b)

# 은닉층-출력층
V = tf.Variable(tf.truncated_normal([2, 1]))
c = tf.Variable(tf.zeros([1]))
y = tf.nn.sigmoid(tf.matmul(h, V) + c)

cross_entropy = - tf.reduce_sum(t * tf.log(y) + (1 -
t) * tf.log(1 - y))
```

```
train_step =
tf.train.GradientDescentOptimizer(0.1).minimize(cross
_entropy)
correct_prediction =
tf.equal(tf.to_float(tf.greater(y, 0.5)), t)

# 모델 학습
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(4000):
    sess.run(train_step, feed_dict={
        x: X,
        t: Y
    })
    if epoch % 1000 == 0:
        print('epoch:', epoch)

# 학습 결과 확인
classified = correct_prediction.eval(session=sess,
feed_dict={
    x: X,
    t: Y
})
prob = y.eval(session=sess, feed_dict={
    x: X
})
```

TensorFlow vs. Keras

■ Keras: XOR 문제 해결을 위한 퍼셉트론

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.optimizers import SGD

np.random.seed(123)

# 데이터를 생성한다
# XOR 게이트
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
Y = np.array([[0], [1], [1], [0]])

# 모델 생성
model = Sequential()

# 입력층-은닉층
model.add(Dense(input_dim=2, units=2))
model.add(Activation('sigmoid'))

# 은닉층-출력층
model.add(Dense(units=1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer=SGD(lr=0.1))

# 모델 학습
model.fit(X, Y, epochs=4000, batch_size=4)
```

```
# 학습 결과 확인
classes = model.predict_classes(X, batch_size=4)
prob = model.predict_proba(X, batch_size=4)

print('classified:')
print(Y == classes)
print()
print('output probability:')
print(prob)
```

THANK YOU