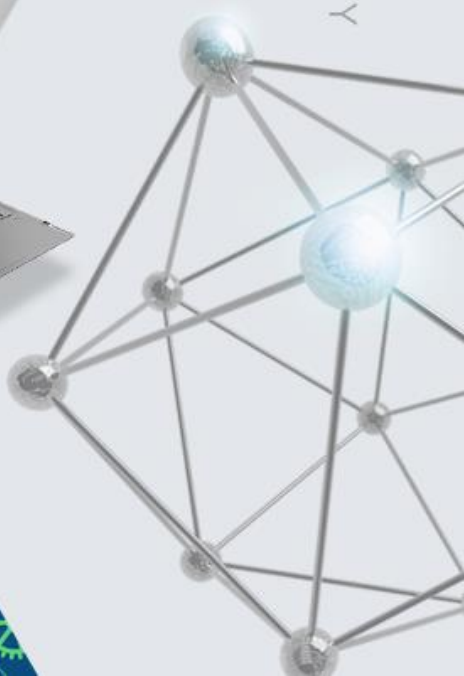
한국기술교육대학교
온라인평생교육원

The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

TECHNOLOGY

# 딥러닝 입문

## 딥러닝을 위한 파이썬



The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

# 딥러닝을 위한 파이썬

## 학/습/목/표

1. 파이썬의 기초 문법에 대해 알아보고, 직접 코딩할 수 있다.
2. 파이썬의 모듈에 대해 이해하고, 외부모듈을 설치 및 활용할 수 있다.
3. 딥러닝 프레임워크에 대해 이해하고, 텐서플로우를 설치할 수 있다.

## 학/습/내/용

1. 파이썬 기초 문법
2. 파이썬의 모듈
3. 딥러닝 프레임워크의 이해

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (1) 자료형이란?

- 프로그래밍
  - 자료(Data) 처리하는 일을 주로 함
- 파이썬에서는 자료를 손쉽게 다룰 수 있도록 내장 자료형 제공

### (2) 파이썬의 내장 자료형

**숫자 (수치) 자료형**

- 정수(int)
- 실수(float)
- 복소수(complex)

**불 자료형**

- True
- False

**군집 자료형**

- 문자열(str)
- 리스트(list)
- 튜플(tuple)
- 딕셔너리 (dict)
- 집합(set)

## 1. 파이썬 기초 문법

### 1) 파이썬 내장 자료형

(3) 내장 자료형의 특징

- 기억장소의 크기, 저장 데이터의 형태, 저장 방식, 값의 범위 등으로 구분
- 파이썬
  - 동적 자료형을 지원하기 때문에 직접 설정할 필요 없음
  - 자료형을 알아서 지정하므로 데이터를 그대로 사용
  - 데이터 타입을 알아낸 후 그에 맞는 객체 생성
- C언어
  - 데이터 입력 시 메모리/표현 방식 등에 따라 세분화 됨

  ex) 숫자의 경우 int, short, unsigned int, float, double, long 등

| 분류 기준 | 종류 |
|---|---|
| 데이터 저장 방법 | 직접 표현, 시퀀스, 매핑 |
| 변경 가능성 | 변경 가능, 불가능 |
| 저장 개수 | 리터럴(한 가지),<br>컨테이너(여러 가지 저장) |

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (4) 수치형 자료형(정수 – int)

- 소수점이 없는 숫자(양수, 0, 음수)
- 기본으로 10진수(접두어를 활용해 2, 8, 16진수 등으로 표현 가능)
- 내장 함수 int()를 활용해 정수 자료형으로 변경 가능
- 범위의 제한 없음(파이썬 버전 3부터 Long형 또한 정수형으로 통합)

```
a = 0
print(type(a))
print(a)
```

```
<class 'int'>
0
```

```
b = -11
print(type(a))
print(a)
```

```
<class 'int'>
-11
```

```
a = 12345 #10진수
print(type(a))
print(a)
```

```
<class 'int'>
12345
```

```
b = 0b11   # 2진수
print(type(b))
print(b)
```

```
<class 'int'>
3
```

```
a = 92233720368547758080045146362345
print(type(a))
print(a)
```

```
<class 'int'>
92233720368547758080045146362345
```

### (5) 수치형 자료형(실수 – float)

- 소수가 있는 숫자
- 지수 표현 가능(e)
- 내장 함수 float()를 활용해 실수자료형으로 변경 가능

```
a = float("0.12")
print(type(a))
print(a)
```

```
<class 'float'>
0.12
```

```
b = 2e-4
print(type(b))
print(b)
```

```
<class 'float'>
0.0002
```

```
c = 3e3
print(type(c))
print(c)
```

```
<class 'float'>
3000.0
```

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (6) 문자열 자료형

- 문자, 단어 등으로 구성된 문자들의 집합
- 내장 함수 str()을 활용해 문자열 자료형으로 변경 가능
- 큰 따옴표(")와 작은 따옴표(') 모두 사용 가능

```
a = '1'
print(type(a))
print(a)
```

```
<class 'str'>
1
```

```
b = "Hello, World !"
print(type(b))
print(b)
```

```
<class 'str'>
Hello, World !
```

```
c = 12345
c = str(c)
print(type(c))
print(c)
```

```
<class 'str'>
12345
```

### (7) 문자열 자료형(연산자)

- 연결 연산자(+): 연결과 반복으로 연산

```
'1'+'2'
```

```
'12'
```

⟺

```
1 + 2
```

```
3
```

```
a = 'Hello! '
b = "World"
print(a+b)
```

```
Hello! World
```

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (7) 문자열 자료형(연산자)

- 반복 연산자(*)

```
'2' * 3
```

'222'

```
2 * 3
```

6

```
'안녕하세요'*3
```

'안녕하세요안녕하세요안녕하세요'

### (8) 문자열 자료형(슬라이싱)

- 범위 선택 연산자

| 문자 | 안 | 녕 | 하 | 세 | 요 |
|---|---|---|---|---|---|
| 인덱스 | 0 | 1 | 2 | 3 | 4 |

```
a = '안녕하세요'
print(a[1:3])
```

녕하

```
a = '안녕하세요'
print(a[0:5:2])
```

안하요

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (8) 문자열 자료형(인덱싱)

- 선택 연산자
- 문자열은 시퀀스 자료형으로 인덱스가 있고, 인덱스로 값의 접근이 가능함

| 문자 | 안 | 녕 | 하 | 세 | 요 |
|---|---|---|---|---|---|
| 인덱스 | 0 | 1 | 2 | 3 | 4 |

```
a = '안녕하세요'
print(a[0])
```

```
안
```

```
a = '안녕하세요'
print(a[4])
```

```
요
```

## 1. 파이썬 기초 문법

### 1) 파이썬 내장 자료형

#### (10) 리스트 자료형

- 임의의 객체를 순차적으로 저장하는 집합적 자료형
- 문자열이 지닌 대부분의 연산들은 리스트도 지원함
- 대괄호로 정의함

```
l = [1, 2, 3]
```

- 다른 프로그래밍 언어(C, C++) 등과 달리 동적 배열,다차원 배열, 인덱싱 등을 쉽고 편리하게 사용 할 수 있음

```
l = list()
print(l, type(l))
```

```
[] <class 'list'>
```

```
l = [1,2,3]
print(type(l))
print(l)
```

```
<class 'list'>
[1, 2, 3]
```

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (10) 리스트 자료형

- 인덱싱과 슬라이싱

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

```
1
```

```
print(l[0:4])
```

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

```
print(l[5])
```

```
6
```

```
print(l[len(l)-1])
```

```
9
```

- 값의 변경

```
l = [1,2,3,4,5,6,7,8,9]
l[0] = 99
print(l)
```

```
[99, 2, 3, 4, 5, 6, 7, 8, 9]
```

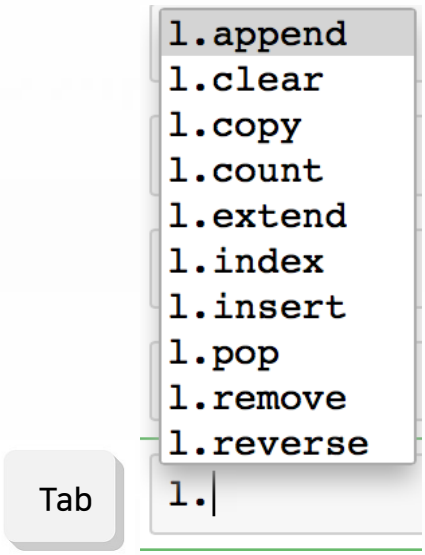
```
l[1] = [1,2,3]
l[2] = "문자"
print(l)
```

```
[99, [1, 2, 3], '문자', 4, 5, 6, 7, 8, 9]
```

## 1) 파이썬 내장 자료형

### (10) 리스트 자료형

- 함수 활용




```
l = [1,2,3,4,5]
print(l)
l.append(6)
print(l)
```

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

```
l = [1,2,3,4,5]
print(l)
l.remove(5)
print(l)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4]
```

### (11) 튜플의 특징

- 리스트와 비슷한 자료형(인덱싱, 슬라이싱 등)

```
l = [1,2,3]
t = (1,2,3)
print(l, type(l))
print(t, type(t))
```

```
[1, 2, 3] <class 'list'>
(1, 2, 3) <class 'tuple'>
```

```
print(l[0], l[0:2])
print(t[0],t[0:2])
```

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

```
print(l + l)
print(t + t)
```

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

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (11) 튜플의 특징

- 임의의 객체를 순차적으로 저장하는 집합적 자료
- 소괄호로 정의함

```
t = (1, 2, 3)
```

- 리스트와 유사하나 값 변경 불가능

```
t = tuple()
print(t, type(t))
```

```
() <class 'tuple'>
```

```
t = (1,2,3)
print(type(t))
print(t)
```

```
<class 'tuple'>
(1, 2, 3)
```

- 리스트와의 차이점(값의 변경 불가능)

  → 리스트보다 빠름, 여러 값을 저장할 때 튜플 사용

```
l = [1,2,3]
t = (1,2,3)
print(l, type(l))
print(t, type(t))
```

```
[1, 2, 3] <class 'list'>
(1, 2, 3) <class 'tuple'>
```

```
l[0] = 5
print(l)
```

```
[5, 2, 3]
```

```
t[0] = 5
print(t)
```

```
---------------------------------------------
TypeError                                 Tra
<ipython-input-27-0fff53691999> in <module>()
----> 1 t[0] = 5
      2 print(t)

TypeError: 'tuple' object does not support it
```

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (12) 사전 자료형

- 키를 이용하여 값을 저장하는 자료형
- 정수형 인덱스가 아닌 키로 값을 저장하므로
  저장된 자료 순서는 의미 없음
- 중괄호로 정의

$$d = \{'a' = 1, 'b'=2, 'c' = 3\}$$

```
d = dict()
print(d, type(d))
```

```
{} <class 'dict'>
```

```
d = {
    'a' : 1,
    'b' : 2,
    'c' : 3
}
print(type(d))
print(d)
```

```
<class 'dict'>
{'a': 1, 'b': 2, 'c': 3}
```

# 1. 파이썬 기초 문법

## 1) 파이썬 내장 자료형

### (12) 사전 자료형

- 정수형 인덱스가 아닌 키와 값으로 자료 저장

```
d = {
    1 : 1,
    'a' : [1,2,3],
    (1,2) : "aaa"

}
```

딕셔너리의 키 값은 변경 불가능한 객체(튜플 포함)가 올 수 있음

```
print(d)
```

```
{1: 1, 'a': [1, 2, 3], (1, 2): 'aaa'}
```

```
print(d['a'])
```

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

- 값의 추가, 수정

```
d = {'a' : 1, 'b' : 2}
print(d)
```

```
{'a': 1, 'b': 2}
```

```
d['a'] = 2
print(d)
```

```
{'a': 2, 'b': 2}
```

```
d['c'] = 3
print(d)
```

```
{'a': 2, 'b': 2, 'c': 3}
```

```
print(d['d'])
```

```
----------------------
KeyError
<ipython-input-38-c19e1
----> 1 print(d['d'])

KeyError: 'd'
```



## 2) 파이썬의 제어문

### (1) 제어문의 종류

- 파이썬의 조건문
  - 프로그램의 실행을 제어하는 제어문 중 하나
  - 조건에 따라 실행 결과를 달리함
  - 콜론과 들여쓰기 필수
  - 스페이스바의 개수에 따라서 들여쓰기를 다르게 인식
  - Tab 키를 활용해 들여쓰기 사용하는 것 추천

```
a = 2
if (a == 1):
    print(1)
elif(a == 2):
    print(2)
else:
    print(3)
```

```
2
```

```
a = 2
if (a == 1):
    print(1)
    else:
        print(2)
```

```
  File "<ipython-input-55-5b
    else:
    ^
SyntaxError: invalid syntax
```

- 예제와 같이 들여쓰기가 다르게 되면 문법적으로 오류 발생



## 2) 파이썬의 제어문

### (1) 파이썬의 조건문

- elif를 활용해 여러 개의 조건 추가 가능
- 조건문 내 또 다른 조건문을 넣어 중첩 조건 작성 가능
- 유의점 : 조건이 같지 않도록 함

  → 조건은 위에서부터 순서대로 확인하며 내려옴

```
a = 2
if (a == 2):
    print(1)
elif(a == 2):
    print(2)
else:
    print(4)
```

조건이 같기 때문에 두 번째 조건은 무시

1

```
a = 2
if (a == 1):
    print(1)
elif(a == 2):
    print(2)
elif(a == 3):
    print(3)
else:
    print(4)
```

2

```
a = 1
b = 2
if (a == 1):
    if(b ==3):
        print(1)
    else:
        print(2)
else:
    print(3)
```

2



## 2) 파이썬의 제어문

### (2) 파이썬의 for 반복문

- 파이썬의 제어문 중 하나로 프로그램의 실행을 반복함
- 조건문과 마찬가지로 들여쓰기와 콜론이 매우 중요
- 반복 가능한 객체를 순회하며 반복문 안의 코드를 한번씩 실행

```
for 아이템 in 반복 가능한 객체 :
    실행 코드
```

1. 반복 객체에서 순서대로 하나씩 값을 가져온다.
2. 아이템에 가져온 값을 담는다.
3. 실행 코드를 수행한다.
4. 반복 객체가 끝날때까지 순차적으로 반복한다.

- 리스트, 튜플 등 집합 자료형을 사용하여 요소를 하나씩 가져와 반복 가능
- 반복 가능 객체인 문자열, 사전, 집합 자료형으로도 반복문 사용 가능

```
for i in 10:
    print(i)
```

```
---------------------------------------
TypeError
<ipython-input-27-60bf35fedd47> in <modu
----> 1 for i in 10:
      2     print(i)

TypeError: 'int' object is not iterable
```

```
for i in {1,2,3}:
    print(i)
```

```
1
2
3
```

```
for i in [1,2,3]:
    print(i)
```

```
1
2
3
```

```
for i in "Hello":
    print(i)
```

```
H
e
l
l
o
```

## 2) 파이썬의 제어문

### (3) 파이썬의 while 반복문

- 해당 조건이 참(True)인 경우 반복해서 실행 코드를 반복하는 제어문

```
while (조건문) :
    실행 코드
```

콜론, 들여쓰기 주의!

- 유의점 : 무한루프에 빠지지 않도록 조건 설정에 주의

```
In [*]: while(True):
            print(1)
```

```
1
1
1
1
1
1
```

```
num = 5
while (num > 0):
    print(num)
    num -= 1
```

```
5
4
3
2
1
```

- 조건문과 break 보조 제어문을 활용하여 특별한 조건에 반복문을 완전히 빠져나감
- 유의점 : break를 만나는 순간 제어문을 빠져나가기 때문에 코드 작성에 유의

```
num = 10
while (num > 0):
    if(num == 6):
        print("--end--")
        break
    print(num)
    num -= 1
```

```
10
9
8
7
--end--
```



## 2) 파이썬의 제어문

### (3) 파이썬의 while 반복문

- 조건문과 continue 보조 제어문을 활용해 특별한 조건에 해당 반복을 건너뛸 수 있음
- 유의점 : continue를 만나는 순간, 반복을 건너뛰기 때문에 코드 작성에 유의

```
num = 10
while (num > 0):
    print(num, end =', ')
    num -= 1
    if(num == 6):
        continue
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
```

```
num = 10
while (num > 0):
    print(num, end =', ')
    if(num == 6):
        continue
    num -= 1
```

```
10, 9, 8, 7, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
```

## 2. 파이썬의 모듈

### 1) 모듈의 정의

(1) 모듈

- 파이썬 코드를 논리적으로 묶어서 별도로 관리하고 사용할 수 있도록 하는 것
- 하나의 파이썬 .py 파일이 하나의 모듈

(2) 모듈의 종류

| 표준 모듈 | 사용자 정의 모듈 | 외부 모듈 (3rd party) |
|---|---|---|

```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', '
, 'finally', 'for', 'from',
'return', 'try', 'while', '
```

(3) 모듈을 가져오는 법

- import 모듈명으로 모듈을 가져옴
- as(alias)를 활용해 긴 모듈명을 줄일 수 있음
- 모듈은 하나의 새로운 이름 공간을 확보하며 정의됨

## 2. 파이썬의 모듈

### 1) 모듈의 정의

#### (4) 함수와 모듈의 차이

**함수**

파일 내의 일부 코드를 묶어 사용하는 것

**모듈**

파일 단위로 코드를 묶어 사용하는 것

비슷하거나 관련된 함수 등의 코드를 한 파일에 저장하고 추후 사용하는 단위

```
import keyword as k

print(k.kwlist)
```

```
['False', 'None', 'True', '
, 'finally', 'for', 'from',
'return', 'try', 'while', '
```

```
print(kwlist)
```

```
---------------------------------------
NameError
<ipython-input-7-43df03ada53d> in <module>
----> 1 print(kwlist)

NameError: name 'kwlist' is not defined
```

#### (5) 모듈의 장점

- 코드의 재사용을 줄임
- 전체 코드를 관련된 모듈들로 분리하여 설계함으로써 구조적 프로그래밍 가능
- 별도의 이름공간을 제공함으로써 동일한 이름의 여러 함수나 변수들을 모듈마다 정의 가능

## 2. 파이썬의 모듈

### 1) 모듈의 정의

#### (6) 모듈의 검색 경로

- 파이썬이 모듈을 검색하는 순서

이미 메모리에 로딩된 모듈

현재 디렉토리에 있는 .py 파일

환경변수(PYTHONPATH)에 등록된
경로에 있는 파일들

표준 모듈 목록

## 2. 파이썬의 모듈

### 2) 외부 모듈

(1) 외부 모듈이란?

- 파이썬은 개방형 소스 개발 프로젝트로서, 다양한 사용자층과 커뮤니티가 활성화되어 있음
- 공개 소스 라이센스 계약에 따라 개발한 소프트웨어를 다른 사용자들이 사용 가능
- 파이썬 패키지 인덱스(PyPI)로 불리는 라이브러리 저장소 관리 기구가 대표적


- 파이썬은 다양한 외부 모듈을 제공하기 때문에 4차 산업혁명과 관련해 파이썬의 중요성이 높아지고 있음

| | |
|---|---|
| 데이터 분석, 통계 | numpy, pandas, matplotlib … |
| 인공지능 | Tensorflow, PyTorch, Keras … |
| 웹 크롤링 | BeautifulSoup, selenium … |

## 2. 파이썬의 모듈

### 2) 외부 모듈

#### (1) 외부 모듈 설치하기

- 패키지 관리자 사용하기(PIP)

  - 파이썬 3.4 이후 버전은 pip를 포함하고 있으며, 손쉽게 외부 모듈 설치 가능

  - 콘솔 창에서 pip install 패키지명 한 줄로 외부 모듈 설치 가능

```
[zoostar@~$pip install pandas
Requirement already satisfied: pandas in ./anaconda/li
(0.23.0)
Requirement already satisfied: python-dateutil>=2.5.0
6/site-packages (from pandas) (2.6.0)
Requirement already satisfied: pytz>=2011k in ./anacon
```

- Jupyter Notebook 환경에서 명령어 앞 !를 붙여 설치 가능

  - 느낌표(!)를 붙이면 콘솔 창에서 입력하는 것과 같은 역할

  - 삭제는 pip uninstall 모듈명

```
In [2]: !pip install pandas

        Requirement already satisfied: panda
        Requirement already satisfied: pytho
        ndas) (2.6.0)
        Requirement already satisfied: pytz>
        .10)
        Requirement already satisfied: numpy
```

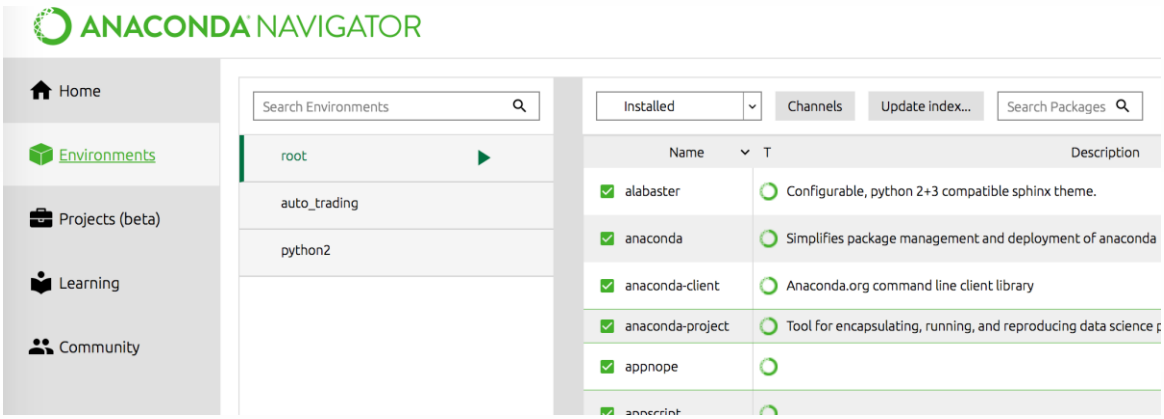
## 2. 파이썬의 모듈

### 2) 외부 모듈

#### (1) 외부 모듈 설치하기

- Anaconda를 설치했다면, Anaconda Navigator에서 설치 가능

  - 환경을 달리 정의하고, 환경 마다 패키지(외부 모듈) 설치 가능



# 3. 딥러닝 프레임워크의 이해

## 1) 딥러닝 프레임워크의 정의

(1) 딥러닝(인공지능)에 사용되고 있는 알고리즘

- CNN, RNN… 등의 알고리즘이 대표적
- 하나의 문제를 해결하기 위해 두 개 이상의 알고리즘을 함께 사용하는 경우도 있음
- 매번 새롭게 직접 구현하는 것은 딥러닝을 사용하는데 매우 비효율적임

(2) 프레임워크

- 어떤 프로그램을 개발하기 위해 필요한 여러가지 기능들을 효율적으로 사용할 수 있도록 묶어둔 기본 틀

(3) 딥러닝 프레임워크

- 딥러닝 학습을 위해 검증되어 온 라이브러리와 알고리즘들을 직접 구현하지 않고 사용할 수 있도록 제공하는 일종의 패키지
- 반복적이고 소모적인 작업으로부터 벗어남
- 복잡한 문제 해결을 위한 핵심 알고리즘 개발 및 구현에 집중
- 대표적인 딥러닝 프레임워크
  - 텐서플로우(TensorFlow, Torch, Keras, Theano 등)

## 3. 딥러닝 프레임워크의 이해

### 2) 텐서플로우 정의

#### (1) 텐서플로우 정의

- 구글(google)에서 만든 기계학습을 위한 오픈소스 플랫폼
- 손쉽게 인공지능 개발을 할 수 있도록 여러 기능을 제공함
- 기계학습의 복잡한 모델을 쉽게 구현하기 위해 데이터의 획득, 모델 학습, 평가, 예측 등의 과정을 쉽게 구현한 플랫폼
- 자바스크립트, Swift 등 다른 언어에서도 지원하지만 파이썬에서 가장 많이 사용하고 있음
- 추상화를 통해 알고리즘 구현을 위한 세부사항 신경쓰지 않아도 됨
- 다차원 데이터 배열(Tensor)과 데이터의 흐름(Data Flow)으로 구성
- 일종의 그래프로 표현함

| 기존 버전(1.X) | 2020년 버전(2.X) |
| --- | --- |
| 인공지능 모델을 구성하기 위해 텐서플로우의 구조와 데이터를 입력하기 위한 각종 문법들을 알아야 함 | 단순화된 인터페이스를 제공하는 또 하나의 딥러닝 프레임워크인 케라스를 지원 |

# 3. 딥러닝 프레임워크의 이해

## 3) 텐서플로우 설치

### (1) 텐서플로우 설치

| 무료로 쉽게 다운 및 설치 가능 | 일반 CPU 버전과 병렬처리를 위한 GPU 버전 2가지를 제공 |
|---|---|

| 구글 Colab | 아나콘다를 설치했다면? |
|---|---|
| - 텐서플로우 2.X 버전이 설치된 환경 제공<br>- 파이썬의 외부 모듈 설치 명령어 pip를 활용하여 손쉽게 설치 가능 | - 가상환경을 만드는 명령어를 활용하여, 딥러닝 학습을 위한 특정 환경 새롭게 구축<br>- 깨끗한 상태로 직접 설치 |

Summary

## 1. 파이썬 기초 문법

- 파이썬에서는 데이터를 손쉽게 다룰 수 있도록 각종 내장 자료형을 제공
- 내장 자료형의 종류 : 수치, 문자열, 불, 리스트, 튜플, 사전, 집합 등
- 그 외에도 프로그램의 실행을 제어할 수 있는 제어문, 함수 등 인공지능에 보다 집중할 수 있도록 편리한 기능들을 제공

## 2. 파이썬의 모듈

- 모듈은 파이썬 코드를 논리적으로 묶어서 관리하고 사용할 수 있도록 하는 것으로 하나의 파일은 하나의 모듈
- 모듈의 종류는 표준 모듈, 사용자 정의 모듈, 외부 모듈이 있으며 import 키워드로 현재 파일에 가져와 사용할 수 있음
- 파이썬은 비영리 재단이 관리하는 오픈 소스로써, 다양한 단체, 커뮤니티에서 활발하게 관련 모듈을 만들고 배포중
- 그 중 대표적으로 PyPI 공식 기구에서 관리하는 모듈은 PIP 이라는 패키지 관리자 명령어로 손쉽게 설치 가능
- 설치 명령어는 pip install 패키지명, 삭제 명령어는 pip uninstall 패키지명으로 간단하게 설치 및 삭제

## 3. 딥러닝 프레임워크의 이해

- 딥러닝 프레임워크란 딥러닝을 쉽게 개발할 수 있도록 각종 알고리즘이나 기능들을 제공하는 일종의 패키지
- Torch, Tensorflow, Keras 등 여러가지 종류가 있지만 본 학습에서는 가장 잘 알려진 텐서플로우를 사용