

PART I

- 딥러닝의 출발

- 사람이 할 수 있는 것과 유사한 판단을 컴퓨터가 해 낼 수 있을까?
→ 인공지능 연구의 시작
- 인공지능 연구 도중, 기존의 데이터를 이용하여 예측하는 '머신러닝(machine learning)' 기법이 효과적임을 발견
- 머신러닝 안의 여러 알고리즘들 중 가장 좋은 효과를 내는 것이 딥러닝

- 딥러닝의 현재

- 딥러닝이 암을 대신 진단하고 생명 현상의 신비를 풀어내며, 각종 산업 전반에 커다란 변화를 가져오고 있음

- 인공지능 > 머신러닝 > 딥러닝



인공지능, 머신러닝, 딥러닝의 관계

- 인공지능의 큰 범주 안에 머신러닝이 속하고, 머신러닝의 일부분이 딥러닝
 - 딥러닝을 배우려면 반드시 머신러닝의 기초 개념을 알아야 함

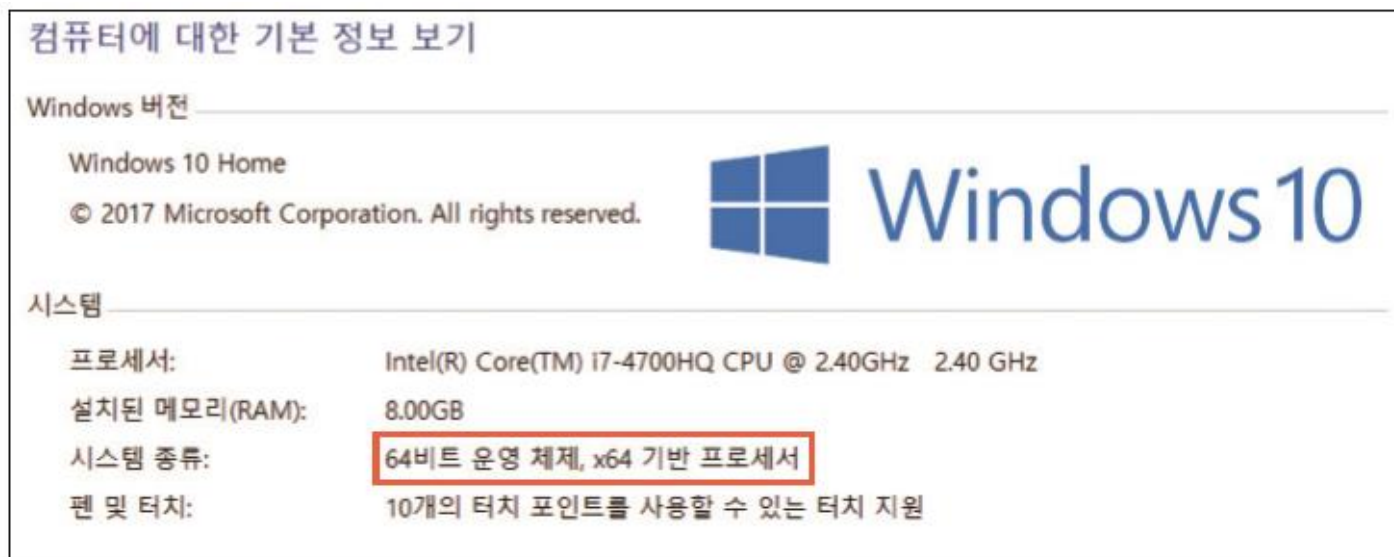
- 딥러닝 학습의 어려움

- 딥러닝은 머신 러닝을 기반으로 만들어진 최신 알고리즘
 - 딥러닝을 배우기 위해서는 머신러닝의 기초를 먼저 배워야 함
- 머신러닝은 여러 가지 수학 공식이 쏟아져 나오는 쉽지 않은 분야
- 그 '진입 장벽'을 뛰어 넘고 나면 다른 사람이 쉽게 넘보지 못하는 경쟁력을 얻게 된다는 의미

1 | 딥러닝 실행을 위한 준비 사항

내 컴퓨터의 시스템 정보 확인하기

- 텐서플로는 64비트 윈도우만을 지원하므로 사용하는 PC가 64비트인지 확인
- 시작 > 검색 > 제어판 > 시스템 순서로 시스템 패널을 열어 확인



내 컴퓨터의 시스템 정보 확인

1 | 딥러닝 실행을 위한 준비 사항

CPU? GPU?

- 딥러닝을 일반 CPU에서 동작시킬지, 고속 그래픽 처리에 특화된 전용 프로세서인 GPU에서 동작시킬지 선택할 수 있음
- 책의 모든 예제는 CPU와 GPU 어떤 환경에서도 잘 동작
- 딥러닝을 처음 접하는 사람은 CPU환경에서 먼저 학습해 보는 것도 좋다.
- 학습을 마친 후 대용량 데이터를 사용할 경우, GPU 작업 환경을 갖추길 추천

2 | 딥러닝 작업 환경 만들기

아나콘다 설치하기

- ① 아나콘다 사이트에서 아나콘다3 64비트 인스톨러를 내려받은 후 설치
(파이썬 3.5 이상 버전을 선택)

- <https://www.continuum.io/downloads>



2 | 딥러닝 작업 환경 만들기



아나콘다 설치

2 | 딥러닝 작업 환경 만들기

- ② 윈도우 버튼을 누른 후 방금 전에 설치한 아나콘다 프로그램 폴더 중 Anaconda Prompt 선택
프로그램 폴더가 안 보이면 검색(🔍) 버튼을 누르고 Anaconda Prompt 입력

Anaconda Prompt를 선택

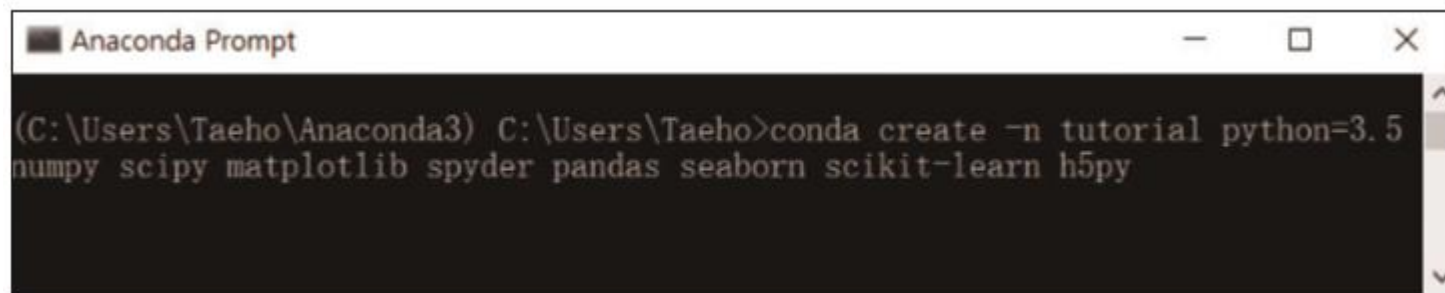


2 | 딥러닝 작업 환경 만들기

- ③ 명령 프롬프트가 나타나면 다음과 같이 아나콘다 환경과 실습용 작업 환경을 tutorial이란 이름으로 생성하여 저장

conda create -n tutorial python=3.5 numpy scipy matplotlib
spyder pandas seaborn scikit-learn h5py → 라이브러리의 이름

작업 환경 이름 파이썬 버전



```
Anaconda Prompt
(C:\Users\Taeho\Anaconda3) C:\Users\Taeho>conda create -n tutorial python=3.5
numpy scipy matplotlib spyder pandas seaborn scikit-learn h5py
```

아나콘다 환경 설정

2 | 딥러닝 작업 환경 만들기

텐서플로 설치하기

- ① 아나콘다 명령 프롬프트에서 activate tutorial이라고 입력하고 Enter를 눌러 명령을 실행(앞서 생성한 tutorial 환경을 활성화하는 명령)
명령을 실행하면 (tutorial) 표시가 프롬프트 맨 앞에 나타남

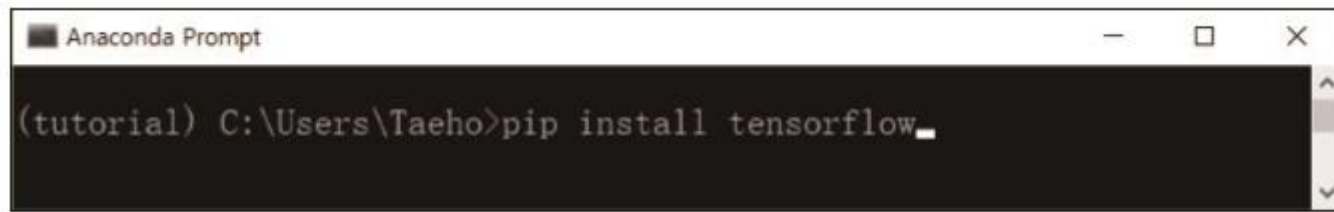


```
Anaconda Prompt
(C:\Users\Taeho\Anaconda3) C:\Users\Taeho>activate tutorial
(tutorial) C:\Users\Taeho>_
```

tutorial 환경을 활성화

2 | 딥러닝 작업 환경 만들기

- ② pip install tensorflow를 입력하고 Enter를 눌러 텐서플로를 설치

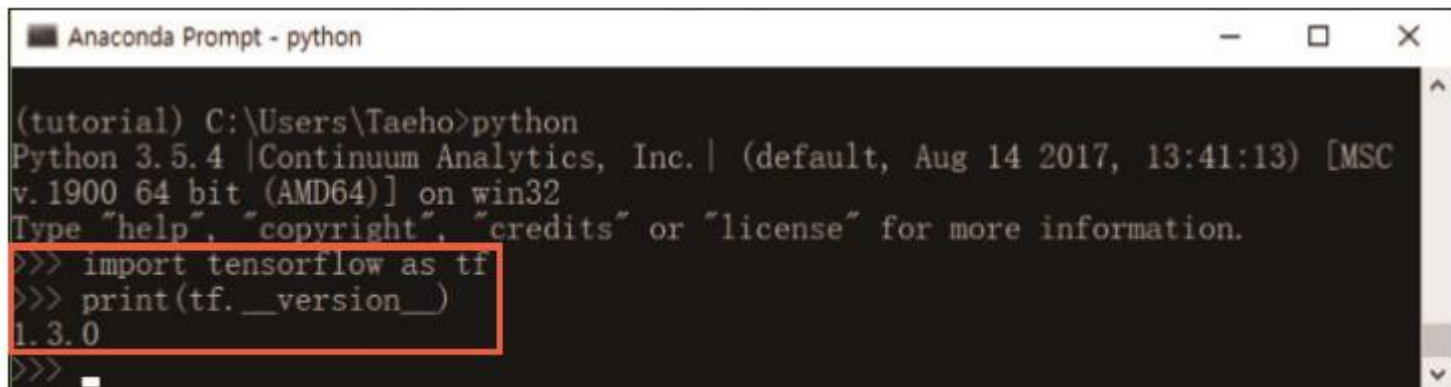
A screenshot of an Anaconda Prompt window. The title bar says "Anaconda Prompt". The command prompt shows the path "C:\Users\Taeho" and the command "pip install tensorflow" being entered. The prompt is "(tutorial) C:\Users\Taeho>".

```
(tutorial) C:\Users\Taeho>pip install tensorflow_
```

텐서플로 설치

2 | 딥러닝 작업 환경 만들기

- ③ 텐서플로가 제대로 설치됐는지 확인하려면 python을 실행한 다음 import tensorflow as tf를 입력
그리고 print(tf.__version__)을 입력했을 때 텐서플로의 버전이 출력되면 설치가 완료된 것



```
Anaconda Prompt - python

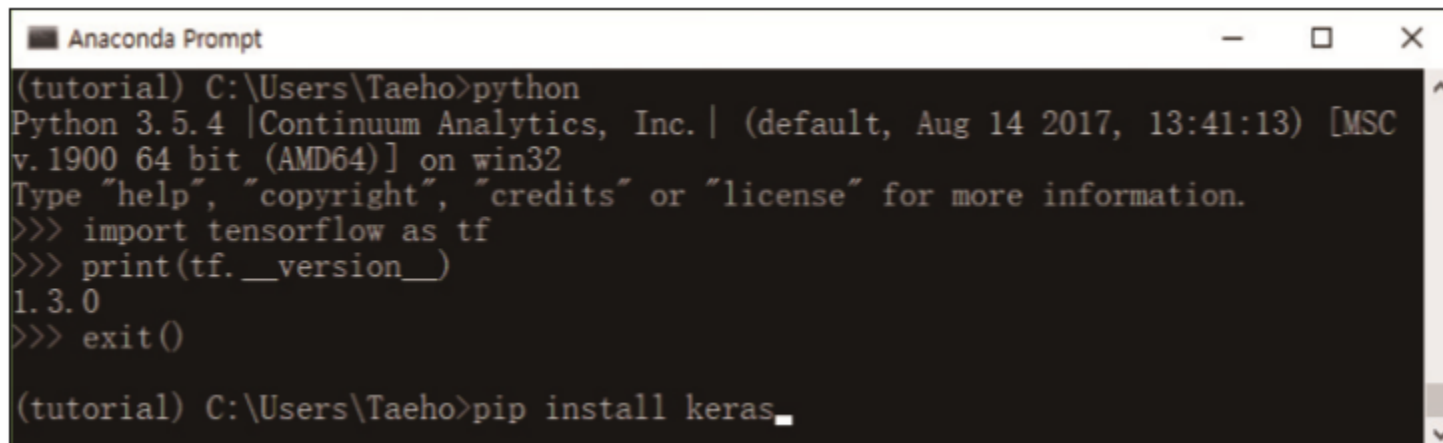
(tutorial) C:\Users\Taeho>python
Python 3.5.4 |Continuum Analytics, Inc. | (default, Aug 14 2017, 13:41:13) [MSC
v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> print(tf.__version__)
1.3.0
>>>
```

텐서플로 설치 여부 확인

2 | 딥러닝 작업 환경 만들기

케라스 설치하기

- ① `exit()` 명령으로 대화형 셸을 빠져나간 다음 `pip install keras`라고 입력하여 케라스를 설치



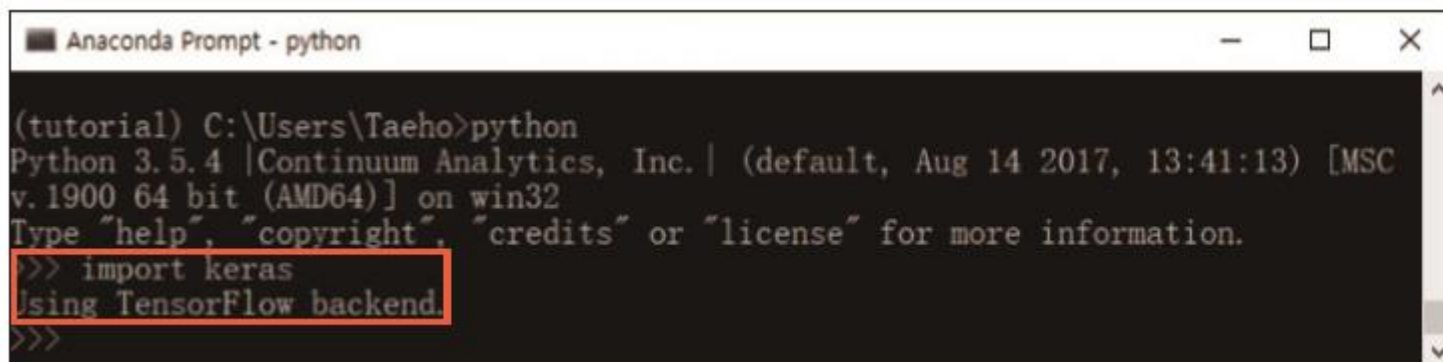
```
Anaconda Prompt
(tutorial) C:\Users\Taeho>python
Python 3.5.4 |Continuum Analytics, Inc. | (default, Aug 14 2017, 13:41:13) [MSC
v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> print(tf.__version__)
1.3.0
>>> exit()

(tutorial) C:\Users\Taeho>pip install keras_
```

케라스 설치

2 | 딥러닝 작업 환경 만들기

- ② 케라스가 설치됐는지 확인하려면 마찬가지로 python을 실행한 다음 import keras를 입력했을 때 다음과 같이 출력되면 설치가 완료된 것

A screenshot of an Anaconda Prompt window titled "Anaconda Prompt - python". The window shows a terminal session where the user has entered 'python' at the prompt. The output shows the Python version (3.5.4) and environment details. The user then enters '>>> import keras', and the output shows 'Using TensorFlow backend.' which is highlighted with a red rectangular box. The prompt '>>>' is visible at the bottom of the terminal window.

```
Anaconda Prompt - python

(tutorial) C:\Users\Taeho>python
Python 3.5.4 |Continuum Analytics, Inc. | (default, Aug 14 2017, 13:41:13) [MSC
v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>>
```

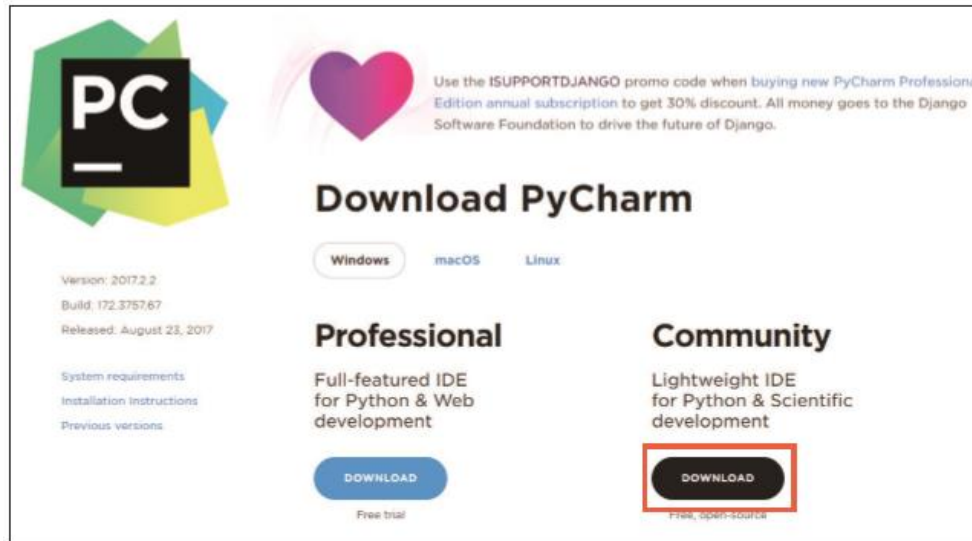
케라스 설치 여부 확인

- ③ 설치를 마친 다음 테스트를 위해 다음 절 "파이참 설치하기"로 넘어감

3 | 파이참 설치하기

① 파이참 내려받기 페이지에 접속하여 Community 버전을 선택한 후 내려받음

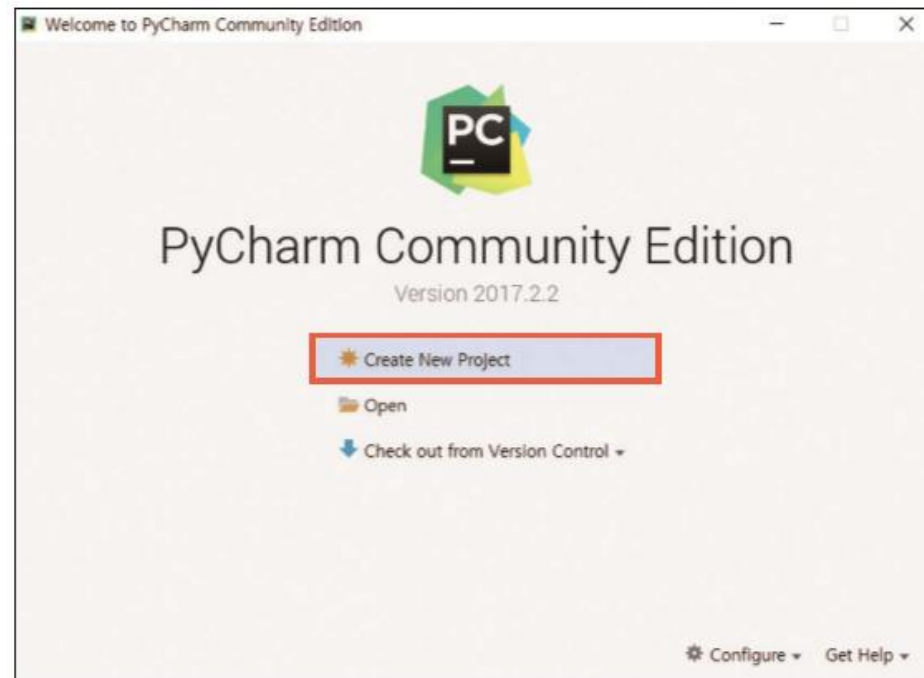
- <https://www.jetbrains.com/pycharm/download/>



파이참 내려받기 페이지에서 Community 버전 내려받기

3 | 파이참 설치하기

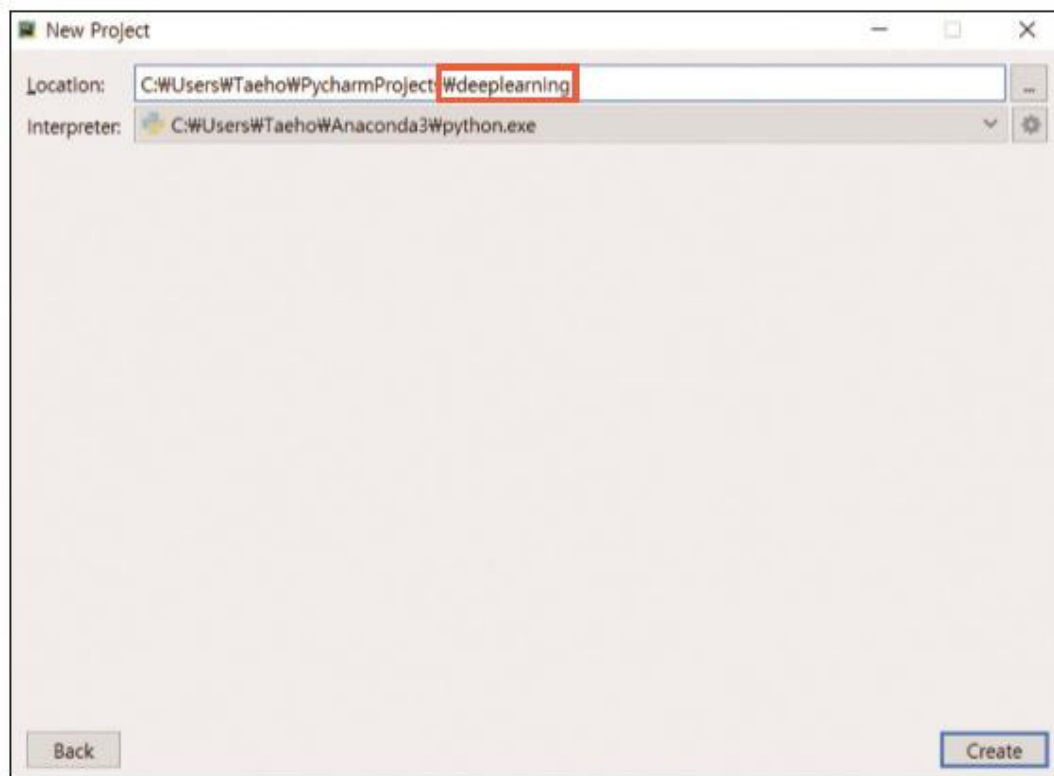
- ② 설치를 마쳤으면 파이참을 실행한 다음 Create New Project 버튼을 눌러 새 프로젝트를 만들



파이참에서 새 프로젝트 생성

3 | 파이참 설치하기

- ③ Location 항목에 PycharmProject 폴더가 나오면 뒤에 **Wdeeplearning** 입력



작업 폴더 만들기

3 | 파이참 설치하기

- ④ 아나콘다 환경을 불러오기 위해 Interpreter 항목 오른쪽 끝에 있는 톱니바퀴 모양 버튼을 눌러 Add Local을 선택

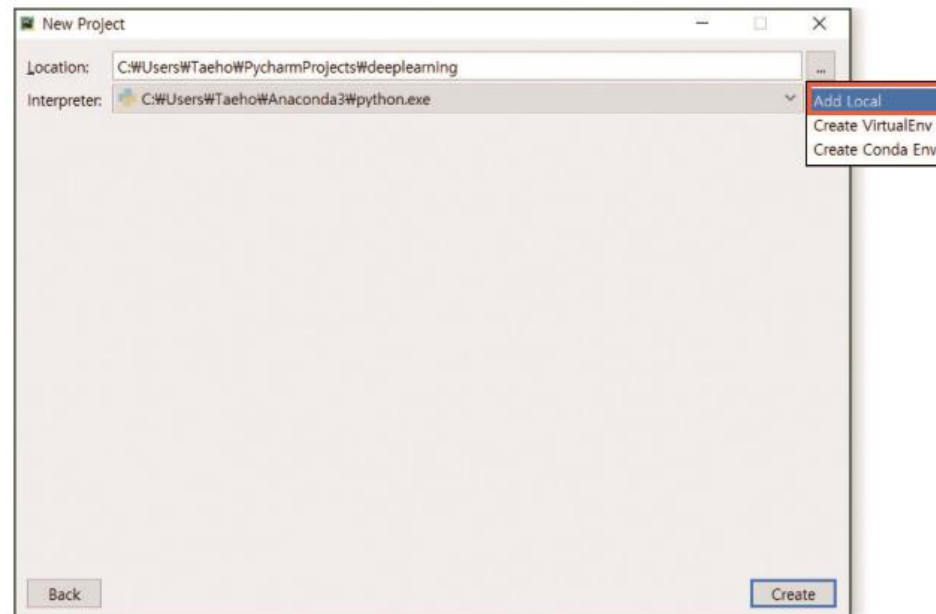
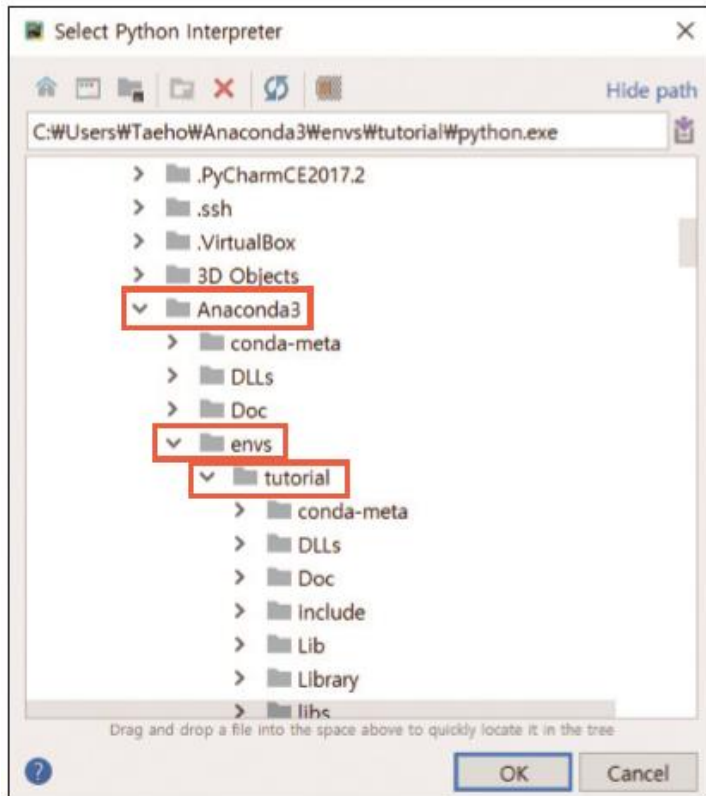


그림 1-14 아나콘다 환경 불러오기

참고: 파이참 버전이 2017.2.2 보다 높을 경우 Add Local의 위치가 다른 수 있음

3 | 파이참 설치하기

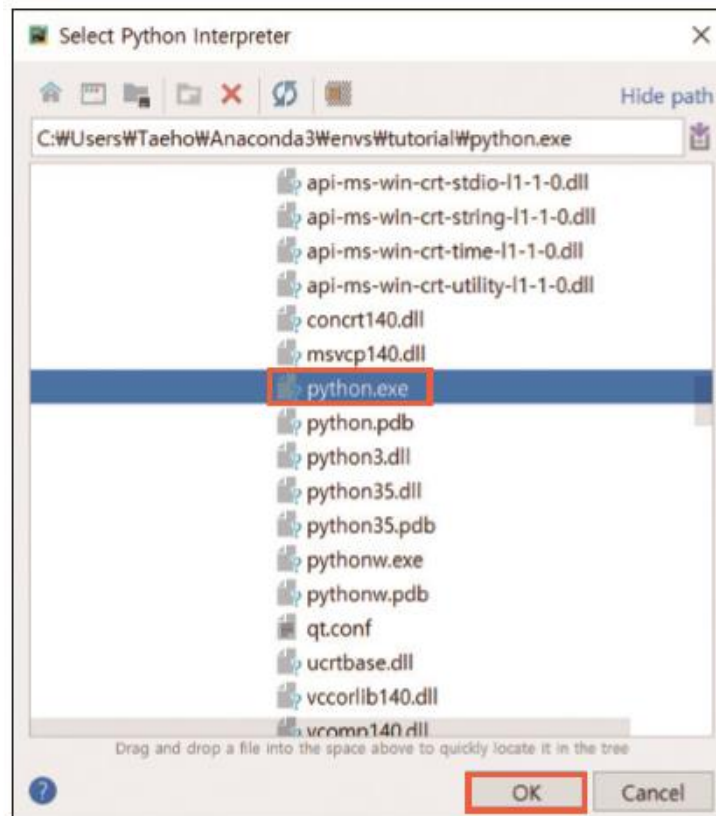
- ⑤ 앞절에서 설치한 tutorial 작업 환경이 Anaconda3 > envs > tutorial에 설치되어 있음. 그림 1-15와 같이 tutorial 폴더를 선택한 후 OK 버튼을 누름



Anaconda3 폴더에서 tutorial 폴더를 찾아 선택

3 | 파이참 설치하기

- ⑥ tutorial 폴더 안에 있는 python.exe를 선택하고 OK 버튼을 누릅니다

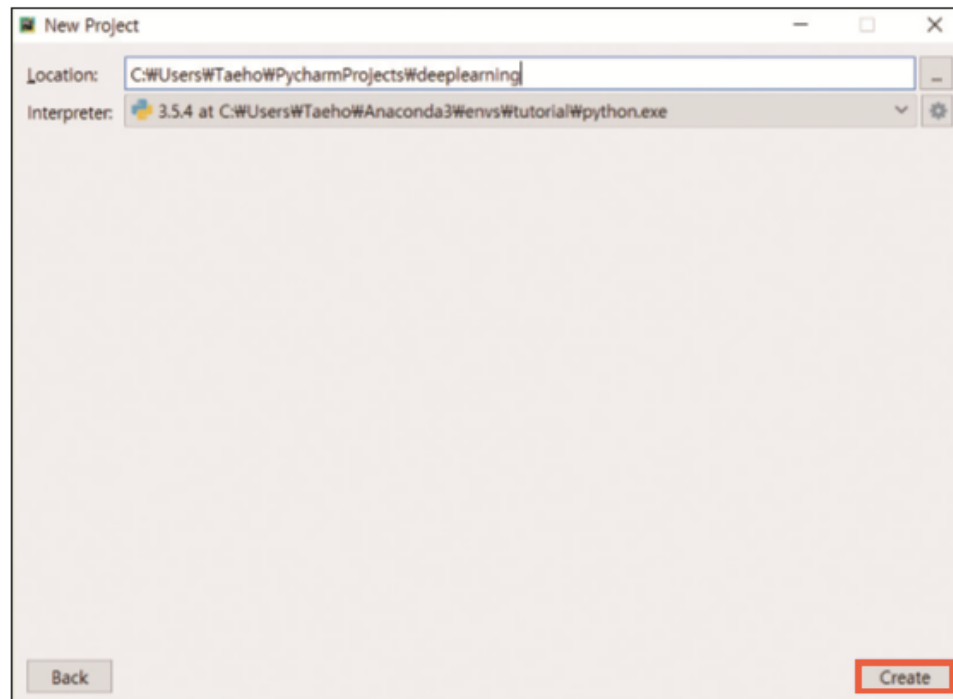


tutorial 폴더 내의 python.exe 선택

3 | 파이참 설치하기

⑦ Interpreter가 바뀌는 것을 확인(다소 시간이 걸릴 수 있음)

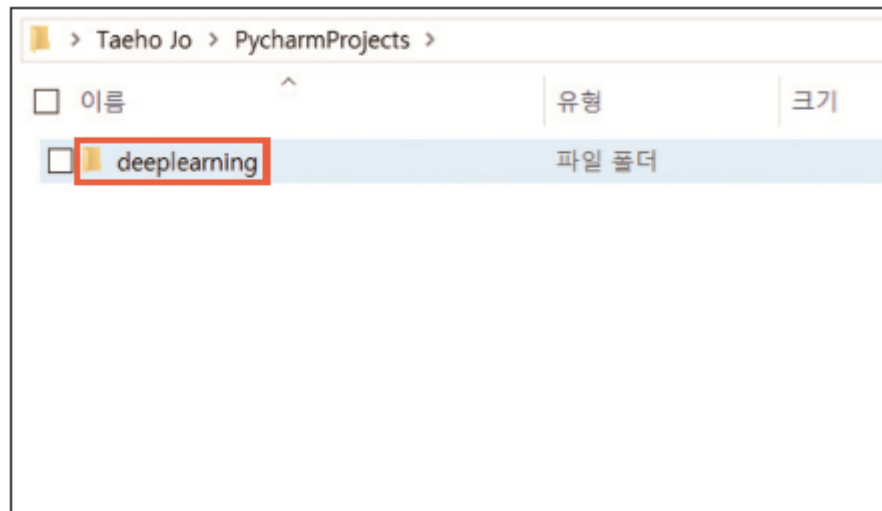
Create 버튼을 눌러 프로젝트 환경을 만듦



Create 버튼 클릭

3 | 파이참 설치하기

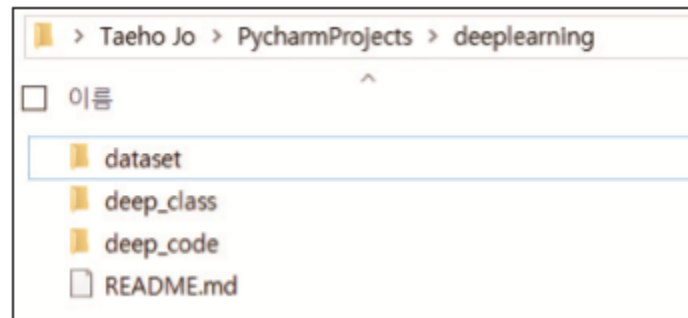
- ⑧ 윈도우 탐색기를 열어 PycharmProjects 폴더에 들어가면 deeplearning 폴더가 생성된 것을 확인할 수 있음



deeplearning 폴더 생성 확인

3 | 파이참 설치하기

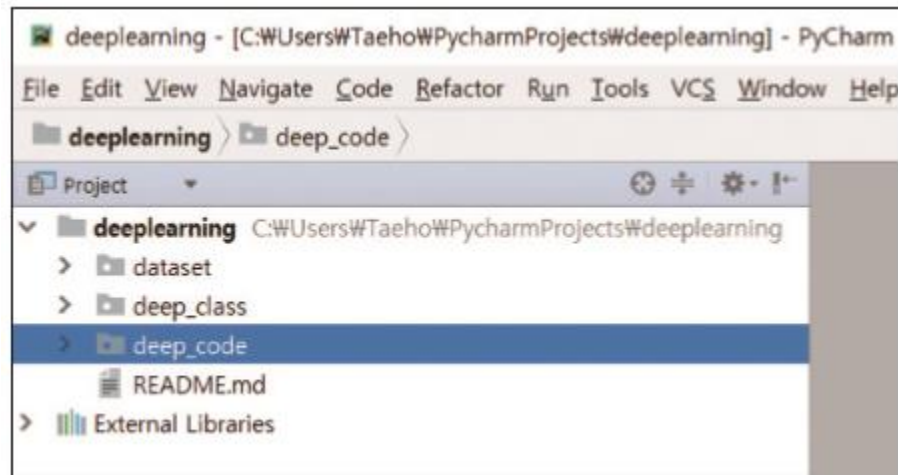
- ⑨ deeplearning 폴더 안에 기 작성된 소스 파일이 있으면 복사해 넣음



deeplearning 폴더 안에 소스 파일 복사

3 | 파이참 설치하기

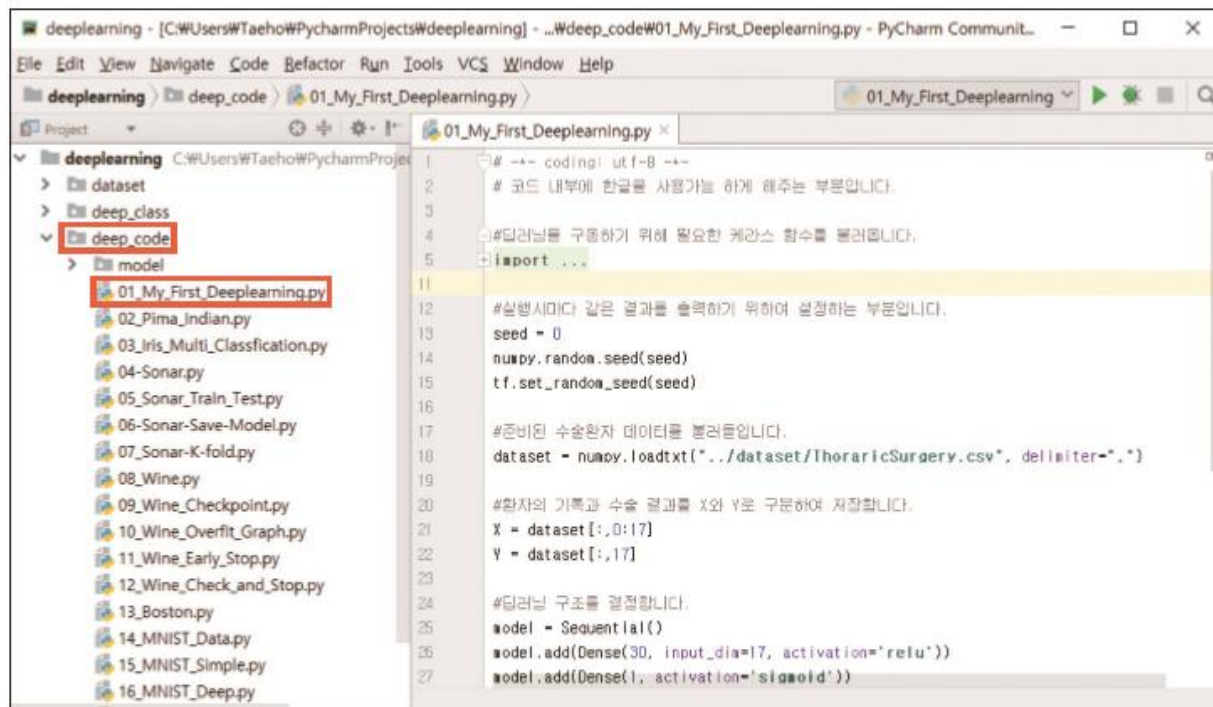
- ⑩ 다시 파이참을 실행. 소스 코드를 실행할 수 있게 설정된 것을 볼 수 있음



파이참에서 소스 코드 폴더 목록 확인하기

4 | 딥러닝 실행하기

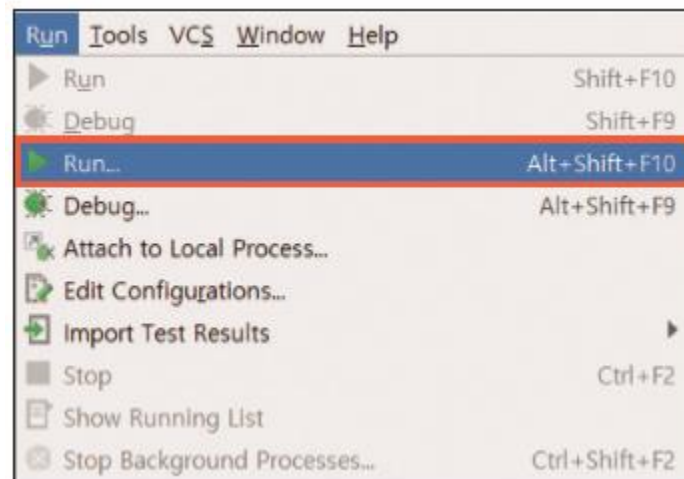
- ① 첫 번째 실습 코드를 불러오기 위해 파이참에서 보이는 deep_code 폴더를 선택
01_My_First_Deeplearning.py를 선택하면 소스 파일이 열림



01_My_First_Deeplearning.py 소스 파일 불러오기

4 | 딥러닝 실행하기

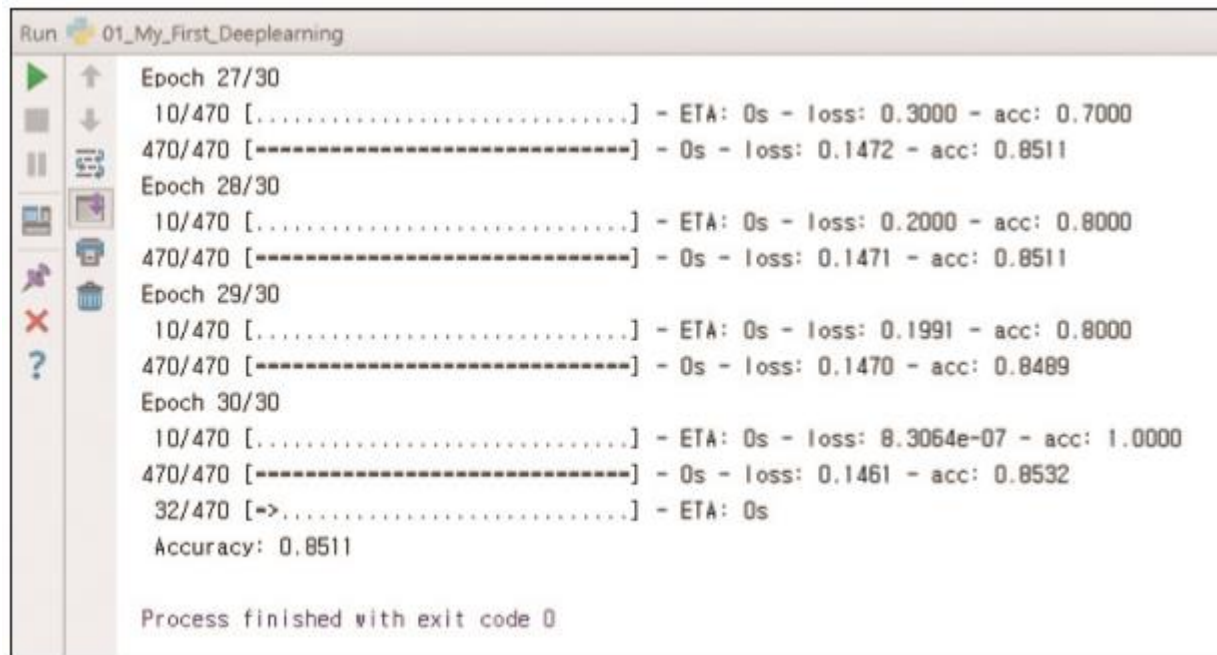
② 메뉴의 Run > Run을 선택해 01_My_First_Deeplearning.py를 실행



01_My_First_Deeplearning.py 실행

4 | 딥러닝 실행하기

- ③ 정상적으로 모두 설치되었다면 다음과 같이 코드가 성공적으로 실행됨



```
Run 01_My_First_DeepLearning
Epoch 27/30
 10/470 [.....] - ETA: 0s - loss: 0.3000 - acc: 0.7000
470/470 [-----] - 0s - loss: 0.1472 - acc: 0.8511
Epoch 28/30
 10/470 [.....] - ETA: 0s - loss: 0.2000 - acc: 0.8000
470/470 [-----] - 0s - loss: 0.1471 - acc: 0.8511
Epoch 29/30
 10/470 [.....] - ETA: 0s - loss: 0.1991 - acc: 0.8000
470/470 [-----] - 0s - loss: 0.1470 - acc: 0.8489
Epoch 30/30
 10/470 [.....] - ETA: 0s - loss: 8.3064e-07 - acc: 1.0000
470/470 [-----] - 0s - loss: 0.1461 - acc: 0.8532
32/470 [=>.....] - ETA: 0s
Accuracy: 0.8511

Process finished with exit code 0
```

실행 결과 확인

직접 해 보는 딥러닝

- 1 | 미지의 일을 예측하는 힘
- 2 | 폐암 수술 환자의 생존율 예측하기
- 3 | 딥러닝 코드 분석
- 4 | '블랙박스'를 극복하려면?

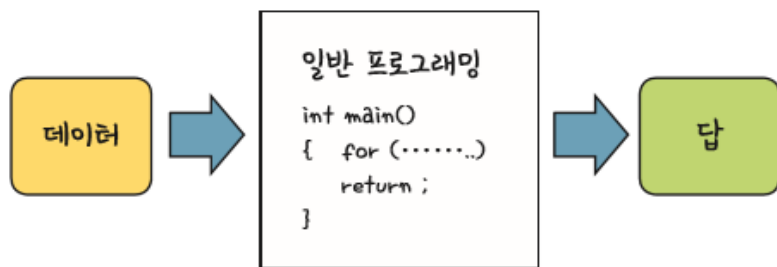
1 | 미지의 일을 예측하는 힘

“기존 환자의 데이터를 이용해 새로운 환자의 생사를 예측하는 프로그램을 짜봐!”

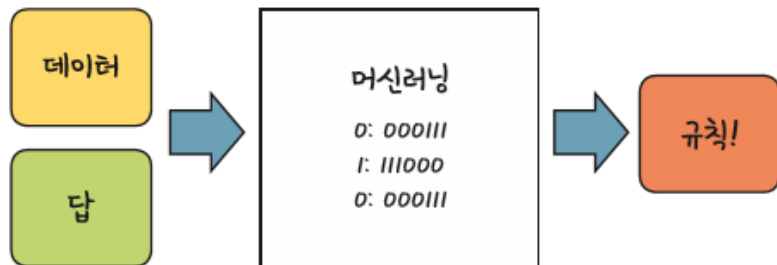
1 | 미지의 일을 예측하는 힘

“기존 환자의 데이터를 이용해 새로운 환자의 생사를 예측하는 프로그램을 짜봐!”

→ 머신 러닝을 이용해야 함!



- 기존의 프로그래밍은 데이터를 입력해서 답을 구한다.
- 머신러닝은 데이터와 답을 입력해서 규칙을 찾는다.
이 규칙을 다른 데이터에도 적용! (예측)



→ 미지의 일을 예측하기 위해 서는 기존 프로그래밍 방식이 아니라 머신 러닝 기법을 써야함

1 | 미지의 일을 예측하는 힘

- 중환자를 전문으로 수술하는 어느 병원 외과 의사의 질문

“혹시 수술하기 전에 수술 후의 생존율을 수치로 예측할 방법이 있을까?”

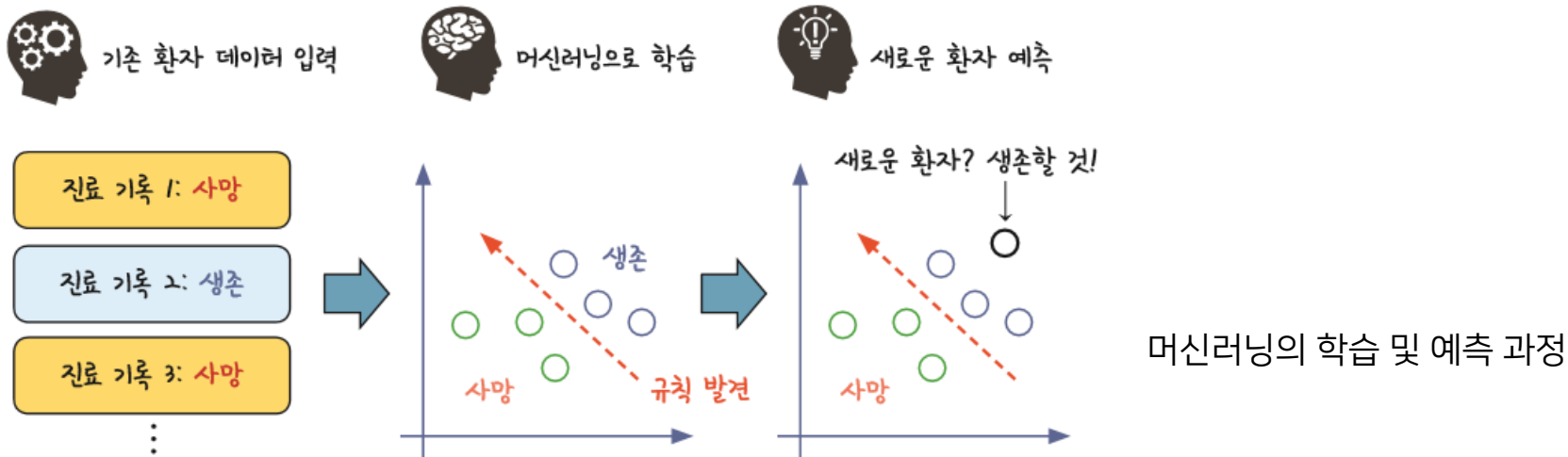
→ 방법은 자신이 그동안 집도한 수술 환자의 수술 전 상태와 수술 후의 생존율을 정리해 놓은 데이터를 머신러닝 알고리즘에 넣는 것

- 기존 환자의 데이터는 머신러닝에 입력되는 순간, 그 패턴과 규칙이 분석됨

→ 이를 학습(training) 이라고 함

- 분석 결과를 새로운 환자의 데이터와 비교해 생존 가능성이 몇 퍼센트인지 알려 줌
이것이 바로 머신러닝이 하는 일

1 | 미지의 일을 예측하는 힘



- **학습** : 깨끗한 좌표 평면에 기존 환자들을 하나씩 배치하는 과정
 - 환자들의 분포를 그래프 위에 펼쳐 놓고
 - 이 분포도 위에 생존과 사망 여부를 구분짓는 경계를 그려넣음.
 - 이를 잘 저장해 놓았다가 새로운 환자가 오면 분포도의 어디쯤 위치하는지를 파악
 - 머신러닝의 예측 성공률은 결국 얼마나 정확한 경계선을 긋느냐에 달려 있다!

2 | 폐암 수술 환자의 생존율 예측하기

폐암 수술 환자의 생존율 예측하기 실습

- 01_My_First_DeepLearning.py

```
# 딥러닝을 구동하는 데 필요한 케라스 함수를 불러옵니다.  
from keras.models import Sequential  
from keras.layers import Dense  
  
# 필요한 라이브러리를 불러옵니다.  
import numpy  
import tensorflow as tf  
  
# 실행할 때마다 같은 결과를 출력하기 위해 설정하는 부분입니다.  
seed = 0  
numpy.random.seed(seed)  
tf.set_random_seed(seed)
```



2 | 폐암 수술 환자의 생존율 예측하기



```
# 준비된 수술 환자 데이터를 불러들입니다.
Data_set = numpy.loadtxt("../dataset/ThoracicSurgery.csv", delimiter=",")

# 환자의 기록과 수술 결과를 X와 Y로 구분하여 저장합니다.
X = Data_set[:,0:17]
Y = Data_set[:,17]

# 딥러닝 구조를 결정합니다(모델을 설정하고 실행하는 부분입니다).
model = Sequential()
model.add(Dense(30, input_dim=17, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# 딥러닝을 실행합니다.
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
model.fit(X, Y, epochs=30, batch_size=10)

# 결과를 출력합니다.
print("\n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

2 | 폐암 수술 환자의 생존율 예측하기

- 실행 결과

(중략)

Epoch 27/30

470/470 [=====] - 0s - loss: 0.1472 - acc: 0.8511

Epoch 28/30

470/470 [=====] - 0s - loss: 0.1470 - acc: 0.8511

Epoch 29/30

470/470 [=====] - 0s - loss: 0.1469 - acc: 0.8489

Epoch 30/30

470/470 [=====] - 0s - loss: 0.1459 - acc: 0.8532

32/470 [=>.....] - ETA: 0s

Accuracy: 0.8511

- 여기서 눈여겨 봐야 할 값은 맨 아래 줄의 정확도(Accuracy)
- 정확도가 1.0 이면 예측 정확도가 100%라는 뜻
- 정확도가 0.8511이라면 이는 이 스크립트를 통해 구현된 딥러닝의 예측 정확도가 85.11%라는 뜻

3 | 딥러닝 코드 분석

첫 번째 부분: 데이터 분석과 입력

- 데이터를 불러와서 사용할 수 있게 만들어 주는 부분

```
# 필요한 라이브러리를 불러옵니다.
```

```
import numpy
```

```
import tensorflow as tf
```

(중략)

```
# 준비된 수술 환자 데이터를 불러들입니다.
```

```
Data_set = numpy.loadtxt("../dataset/ThoracicSurgery.csv", delimiter=",")
```

```
# 환자의 기록과 수술 결과를 X와 Y로 구분하여 저장합니다.
```

```
X = Data_set[:,0:17]
```

```
Y = Data_set[:,17]
```

3 | 딥러닝 코드 분석

- 책에서 설명하는 모든 코드는 파이썬으로 만들어짐
- 파이썬은 초보자부터 전문가까지 사용자 폭이 넓은 프로그래밍 언어로, 다양한 플랫폼에서 사용할 수 있음
- 특히 라이브러리가 풍부하여 연구 기관 및 산업계에서 두루 사용되고 있다
 - 라이브러리란 특정 기능을 담은 작은 프로그램(모듈, module)을 말함
- 함수나 클래스를 따로 담아 라이브러리 형태로 공개한 것이 많은데, 이렇게 미리 만들어진 라이브러리를 불러와서 언제든지 그 기능을 사용하면 됨

3 | 딥러닝 코드 분석

- 라이브러리를 불러올 때 사용하는 명령어가 import

```
import numpy
```

- 넘파이(numpy)라는 라이브러리를 불러오라는 뜻
- 넘파이는 수치 계산을 위해 만들어진 라이브러리로 데이터 분석에 많이 사용됨

3 | 딥러닝 코드 분석

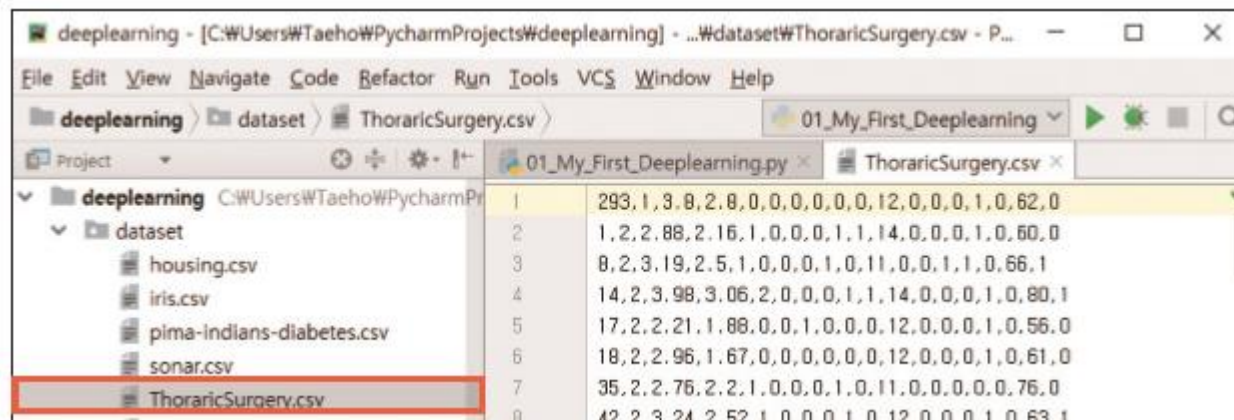
```
Data_set = numpy.loadtxt("../dataset/ThoraricSurgery.csv", delimiter=",")
```

- 넘파이 라이브러리 안에는 여러 함수가 내장되어 있음
- 위 코드에서는 Data_set이라는 임시 저장소를 만들고, 넘파이 라이브러리 안에 있는 loadtxt()라는 함수를 사용했음
- 이를 이용해 'ThoraricSurgery.csv'라는 외부 데이터셋을 불러옴

폴란드의 브로츠와프 의과대학에서 2013년 공개한 폐암 수술 환자의 수술 전 진단 데이터와 수술 후 생존 결과를 기록한 실제 의료 기록 데이터

3 | 딥러닝 코드 분석

- 'ThoraricSurgery.csv' 파일 들여다 보기
- 첫 딥러닝에 사용할 데이터인 'ThoraricSurgery.csv' 파일을 파이참에서 열어보면 모두 470개의 라인으로 이루어져 있고 각 라인은 18개 항목으로 구분되어 있음



ThoraricSurgery.csv 파일 확인

3 | 딥러닝 코드 분석

- 알아보기 쉽게 정리해 보면 다음과 같다.

줄 항목	속성																	클래스
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	293	1	3.8	2.8	0	0	0	0	0	0	12	0	0	0	1	0	62	0
2	1	2	2.88	2.16	1	0	0	0	1	1	14	0	0	0	1	0	60	0
3	8	2	3.19	2.5	1	0	0	0	1	0	11	0	0	1	1	0	66	1
...
470	447	8	5.2	4.1	0	0	0	0	0	0	12	0	0	0	0	0	49	0

폐암 수술 환자의 의료 기록 데이터

3 | 딥러닝 코드 분석

- 한 줄 한 줄이 서로 다른 환자의 상태를 기록한 정보
→ 총 470행이므로 470명의 환자에 대한 정보
- 심표(,)로 이어진 항목이 각 줄마다 18개가 있음
→ 환자마다 18개의 정보를 순서에 맞춰 정리했다는 뜻
- 앞의 17개 정보는 17가지의 환자 상태를 조사해서 기록해 놓은 것. 마지막 18번째 정보는 수술 후 생존 결과. (1은 true, 해당 사항이 있다는 것을 뜻하고 0은 false, 해당 사항이 없다는 것을 뜻함)
- 18번째 항목에서 1은 수술 후 생존했음을, 0은 수술 후 사망했음을 의미

3 | 딥러닝 코드 분석

- 1번째 항목부터 17번째 항목까지를 '속성(attribute)'이라고 하고, 정답에 해당하는 18번째 항목을 '클래스(class)'라고 함
- 이번 프로젝트의 목적
 - 1번째부터 17번째까지의 항목(속성)을 분석해서 18번째 항목(클래스), 즉 수술 후 생존 또는 사망을 맞히는 것
- 프로젝트를 위해 가장 먼저 할 일
 - '속성'만을 뽑아 데이터셋을 만들고,
 - '클래스'를 담은 데이터셋을 따로 만들어 줌

3 | 딥러닝 코드 분석

- 속성 데이터셋 X를 다음과 같이 생성함

```
X = Data_set[:,0:17]
```

- 클래스 데이터셋 Y는 18번째 항목을 이용해 다음과 같이 만들어 줌

```
Y = Data_set[:,17]
```

3 | 딥러닝 코드 분석

두 번째 부분: 딥러닝 실행

딥러닝을 구동하는 데 필요한 케라스 함수를 불러옵니다.

```
from keras.models import Sequential
```

```
from keras.layers import Dense
```

(중략)

딥러닝 구조를 결정합니다(모델을 설정하고 실행하는 부분입니다).

```
model = Sequential()
```

```
model.add(Dense(30, input_dim=17, activation='relu'))
```

```
model.add(Dense(1, activation='sigmoid'))
```

딥러닝을 실행합니다.

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```

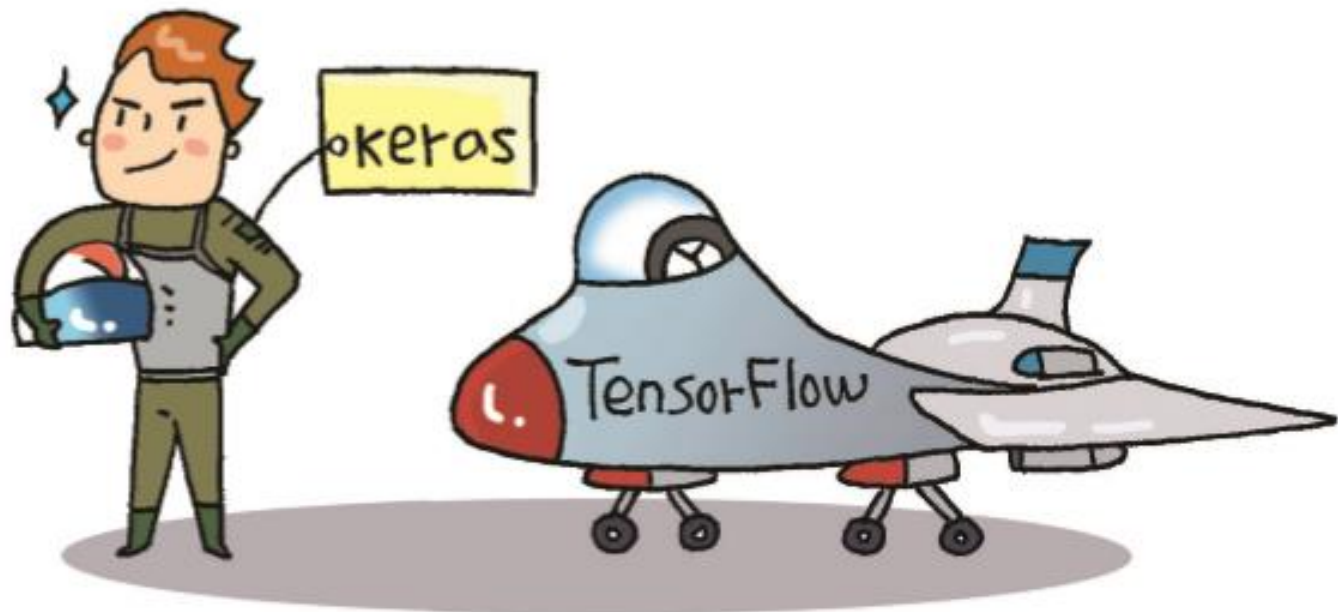
```
model.fit(X, Y, epochs=30, batch_size=10)
```

3 | 딥러닝 코드 분석

- 딥러닝을 실제로 실행하는 부분
- 케라스(keras)를 사용해 딥러닝을 실행
- 케라스가 구동되려면 텐서플로(TensorFlow) 또는 씨아노(theano)라는 두 가지 라이브러리 중 하나가 미리 설치되어 있어야 함
- 우리는 앞서 텐서플로를 선택해서 설치하였음

3 | 딥러닝 코드 분석

- 케라스와 텐서플로의 관계
 - 딥러닝 프로젝트를 '여행'으로 비유해 본다면 텐서플로는 목적지까지 빠르게 이동시켜주는 '비행기'에 해당
 - 케라스는 비행기의 이륙 및 정확한 지점까지의 도착을 책임지는 '파일럿'에 비유할 수 있음



3 | 딥러닝 코드 분석

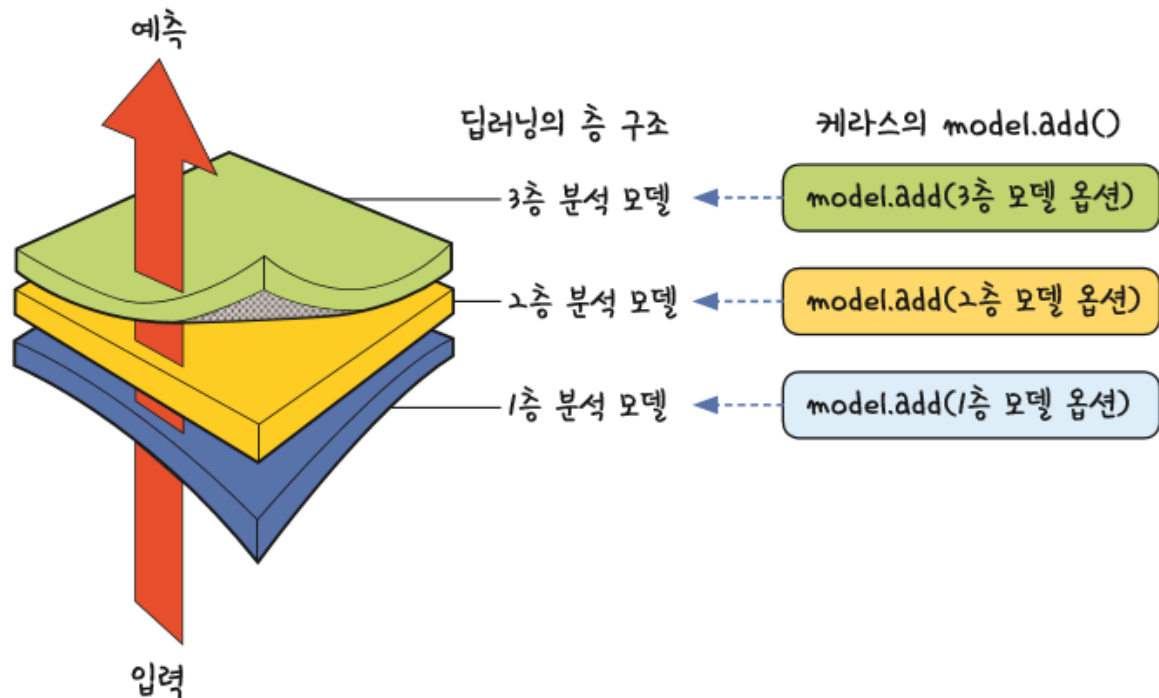
- 설치가 모두 올바르게 되었다면 다음과 같은 방법으로 케라스 라이브러리를 불러올 수 있음

```
from keras.models import Sequential  
from keras.layers import Dense
```

- 케라스 라이브러리 중에서 Sequential 함수와 Dense 함수를 불러왔음

3 | 딥러닝 코드 분석

- 딥러닝은 아래 그림과 같이 여러 층이 쌓여 결과를 만들어 냄
- Sequential 함수는 딥러닝의 구조를 한 층 한 층 쉽게 쌓아올릴 수 있게 해 줌
- Sequential 함수를 선언하고 나서 model.add() 함수를 사용해 필요한 층을 차례로 추가하면 됨



딥러닝의 층 구조와 케라스

3 | 딥러닝 코드 분석

- 01_My_First_Deeplearning.py에서는 model.add() 함수를 이용해 두 개의 층을 쌓아 올렸음

```
model = Sequential()  
model.add(Dense(30, input_dim=17, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

- 층을 몇 개 쌓을지는 데이터에 따라 그때 그때 결정
- 케라스의 가장 큰 장점 중 하나는 model.add() 함수를 이용해 필요한 만큼의 층을 빠르고 쉽게 쌓아 올릴 수 있다는 것

3 | 딥러닝 코드 분석

- `model.add()` 안에는 `Dense()` 함수가 포함되어 있음
→ 영어 단어 `dense`는 '조밀하게 모여있는 집합'이란 뜻, 여기서는 각 층이 제각각 어떤 특성을 가질지 옵션을 설정하는 역할을 함
- 딥러닝의 구조와 층별 옵션을 정하고 나면 `compile()` 함수를 이용해 이를 실행시킴

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])  
model.fit(X, Y, epochs=30, batch_size=10)
```

- 여기에 쓰인 `loss`, `optimizer`, `activation` 등의 키워드를 이해하는 것이 바로 딥러닝 학습의 핵심

3 | 딥러닝 코드 분석

마지막 부분: 결과 출력

```
# 결과를 출력합니다.  
print("\n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

- 출력 부분에서는 model.evaluate() 함수를 이용해 앞서 만든 딥러닝의 모델이 어느 정도 정확하게 예측하는지를 점검할 수 있음
- 이 코드를 통해 출력되는 정확도(Accuracy)는 학습 대상이 되는 기존 환자들의 데이터 중에 일부를 랜덤하게 추출해, 새 환자인 것으로 가정하고 테스트한 결과
- 좀 더 신뢰할 수 있는 정확도를 측정하려면, 학습 단계에서 미리 일부를 떼어내어 테스트셋으로 저장하고 테스트할 때는 오직 이 테스트셋만을 사용

4 | '블랙박스'를 극복하려면?



- 딥러닝 프로젝트를 실행하는 것은 어렵지 않다.
- 그러나 이 결과가 어떻게 나왔는지를 설명하지 못하면 더 나은 결과를 만들기가 어려울 것
- 이것이 바로 딥러닝이라는 '블랙박스'를 열어 그 안에서 구동되는 여러 가지 원리를 공부해야 하는 이유