



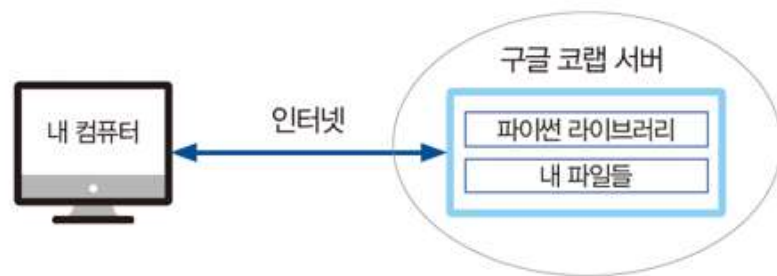
딥러닝 개발 환경

■ 클라우드 방식

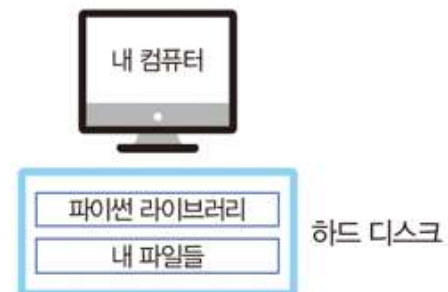
- 프로그램과 데이터가 서버에 저장되고 관리
- 서버에 환경이 대부분 갖추어져 있어 로그인하면 바로 프로그래밍 가능
- 인터넷 연결만 있으면 어느 곳에서도 개발 가능. 협업 가능
- 구글의 Colab, 아마존의 SageMaker, 마이크로소프트의 Azure
- 내 프로젝트에 최적의 환경을 갖추 수 없는 한계

■ 스탠드얼론 방식

- 자신에 최적의 환경 구축 가능. 프로그램과 데이터가 자신의 컴퓨터에 저장
- 소프트웨어를 설치하고 환경을 스스로 구축해야 함



(a) 클라우드 방식



(b) 스탠드얼론 방식

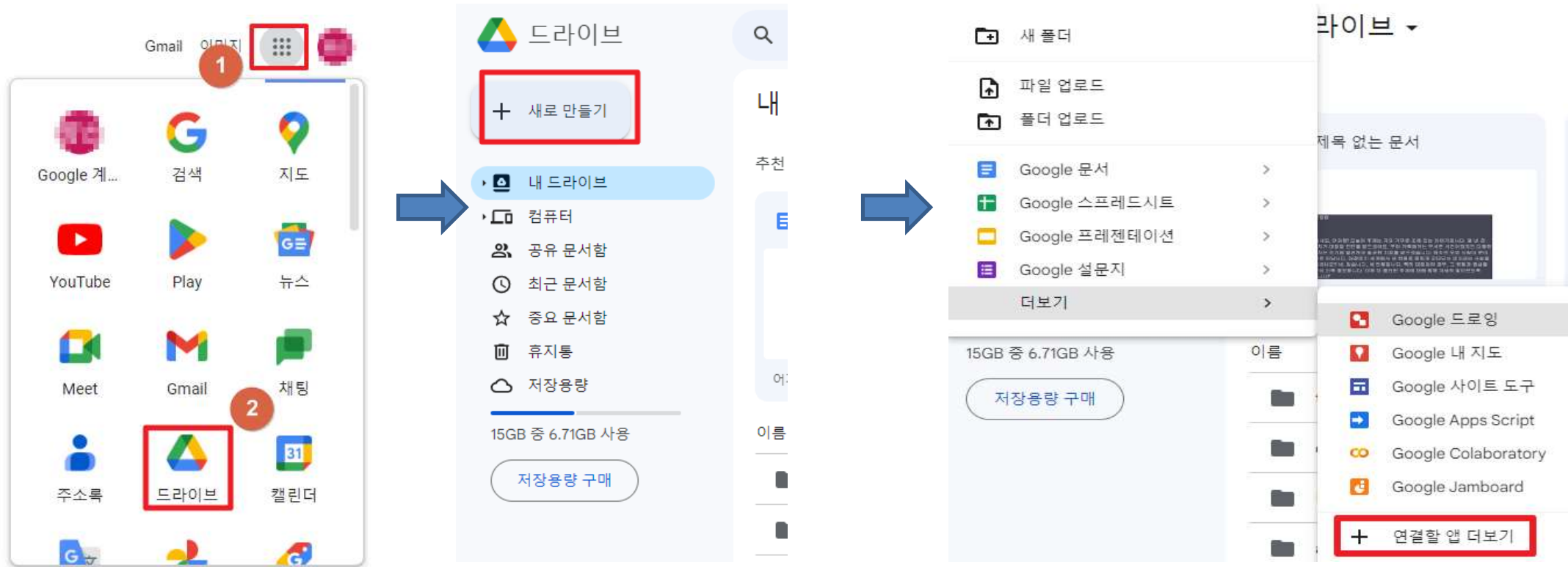
클라우드 방식 딥러닝 개발 환경

■ Colab

- 클라우드 기반의 무료 Jupyter 노트북 개발 환경
 - 코랩 + 구글드라이브 + 도커 + 리눅스 + 구글클라우드의 기술스택
 - Git과의 연동이 용이하여 타인과 지식을 공유하기 좋은 환경이다.
 - 최대 세션 유지시간은 12시간
-
- 구글 계정에 가입한 후 , <https://drive.google.com>에 접속 후, 우클릭하여 다음과 같이 test 폴더를 만든다
 - 좌측 상단의 [+새로만들기] 버튼 > 더보기 > 연결할 앱 더보기를 선택한다

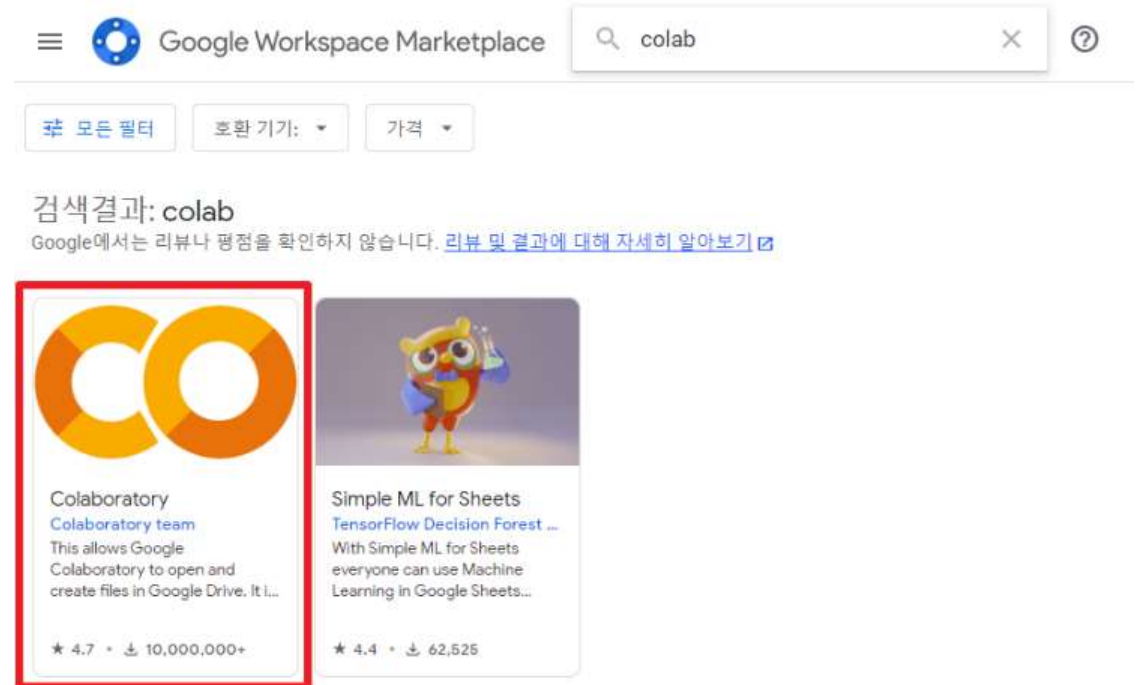
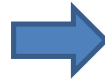
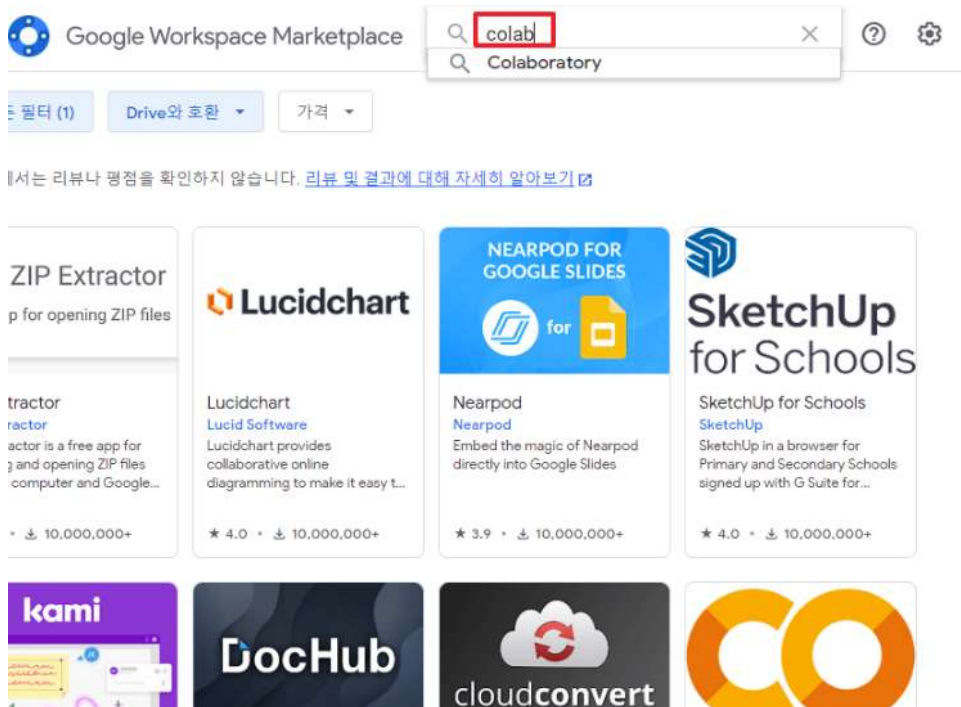
클라우드 방식 딥러닝 개발 환경

■ 구글 드라이버 접속



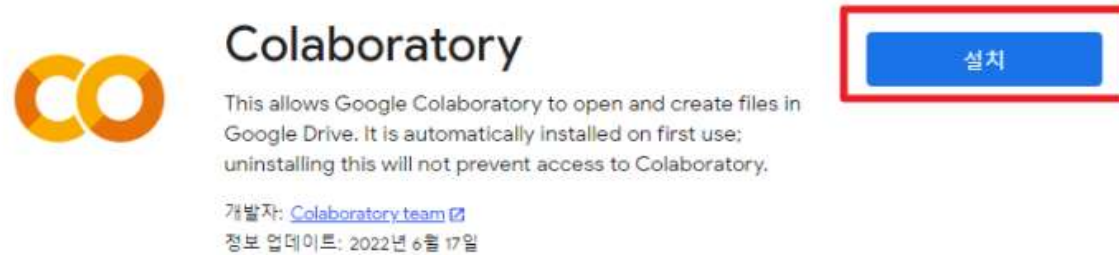
클라우드 방식 딥러닝 개발 환경

■ colab 검색 => Colaboratory



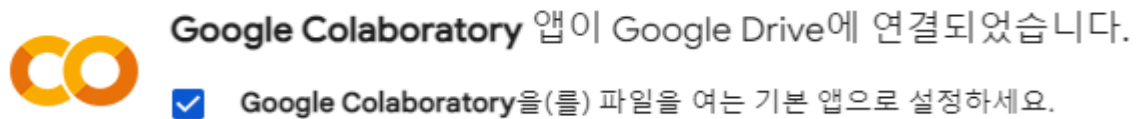
클라우드 방식 딥러닝 개발 환경

■ Colab 설치



호환 기기: 

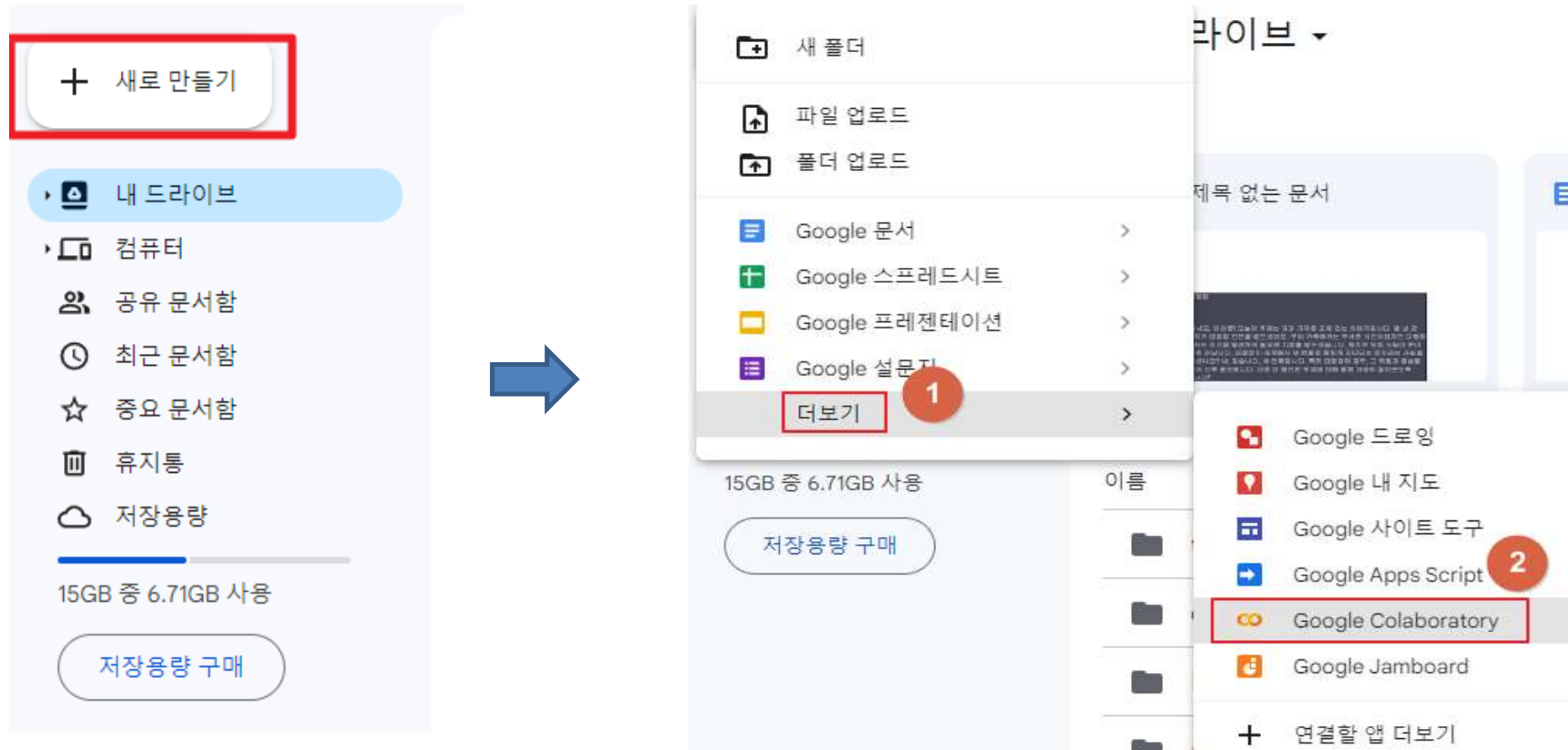
★★★★★ 3,489 ⓘ ⬇ 10,000,000+



확인

클라우드 방식 딥러닝 개발 환경

■ Colab 접속 및 사용



새로 만들기

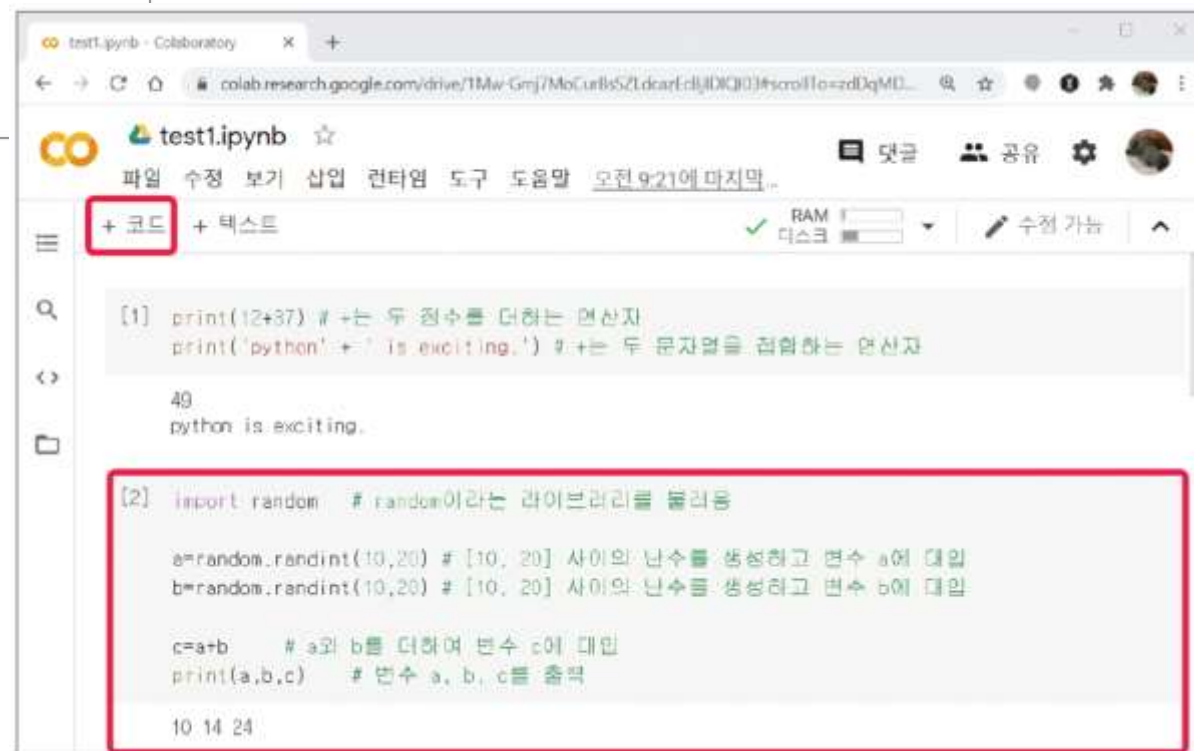
클라우드 방식 딥러닝 개발 환경

■ Colab 접속 및 사용



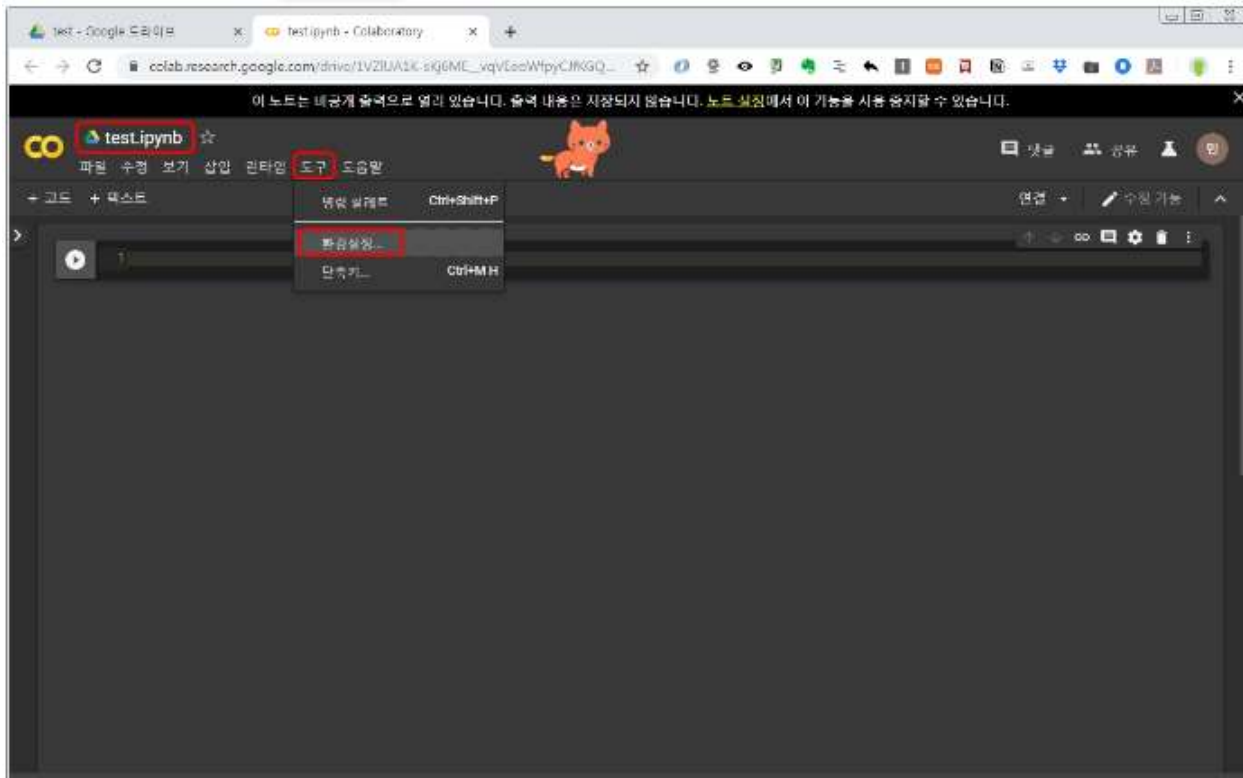
주피터 노트북

파이썬을 대화식으로 편리하게 실행할 수 있는 환경을 만드는 IPython 프로젝트를 수행하던 페르난도 레레츠는 2014년에 파이썬뿐 아니라 R과 Julia 언어도 지원하는 새로운 프로젝트인 주피터 노트북을 분리하여 진행한다. 주피터 노트북은 파이썬 프로그래밍을 위한 대화형 인터페이스의 표준으로 자리 잡는다.



클라우드 방식 딥러닝 개발 환경

Colab 접속 및 사용



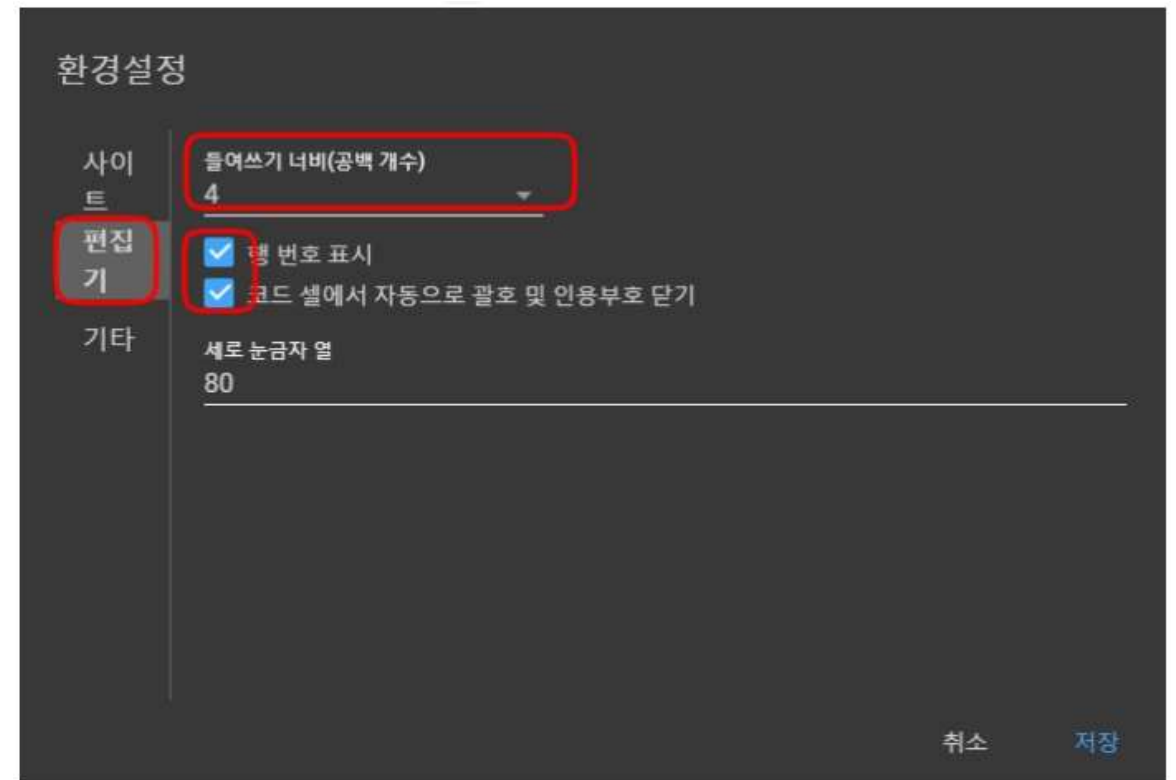
드라이브 메인화면으로 이동 > 톱니바퀴 모양 버튼 클릭
> 설정을 클릭한다

모듈과 라이브러리

모듈은 독립적으로 설치하고 import 명령어로 불러올 수 있는 최소 단위의 코드 모음으로, 실제로는 .py파일로 구성되어 있다
예) random 모듈은 파이썬이 설치되어 있는 폴더에 random.py파일로 저장되어 있다
다른 프로그래밍 언어에서는 모듈 대신 라이브러리라는 용어를 쓴다
파이썬에서는 여러 모듈의 집합을 라이브러리라고 하는데 때때로 모듈 자체를 라이브러리라고 부르기도 한다.

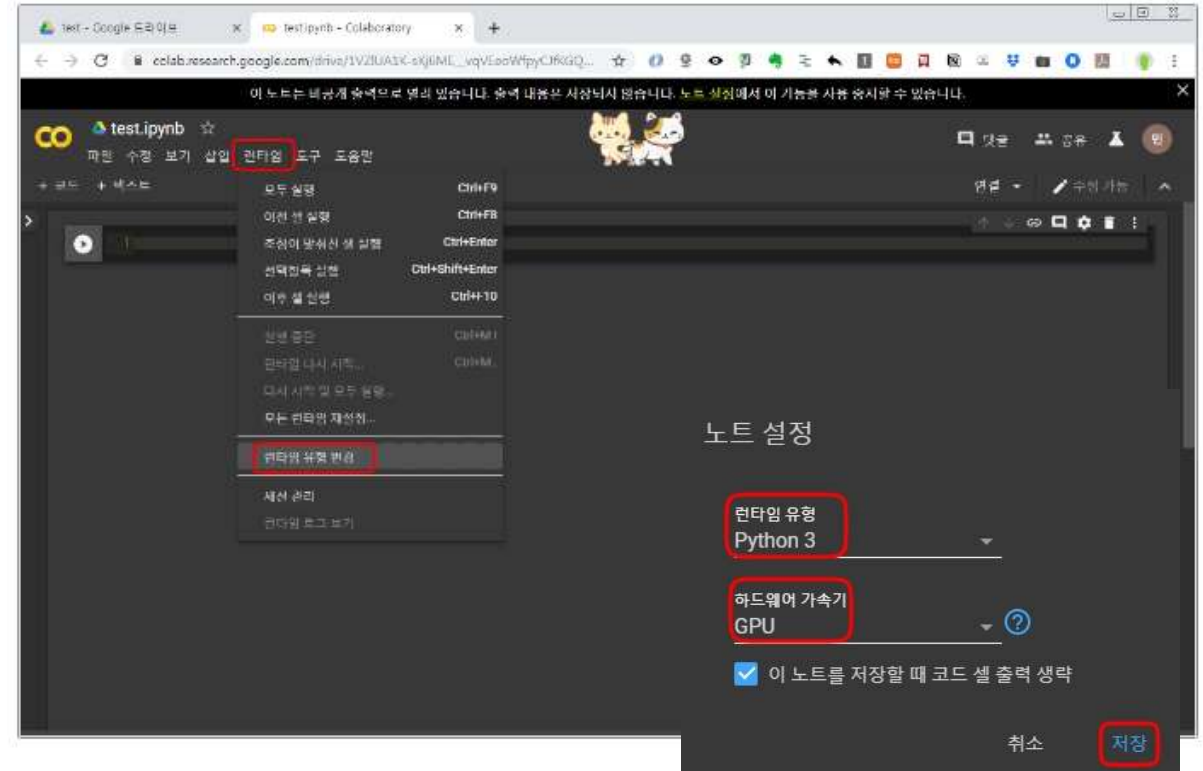
클라우드 방식 딥러닝 개발 환경

■ Colab 접속 및 사용



클라우드 방식 딥러닝 개발 환경

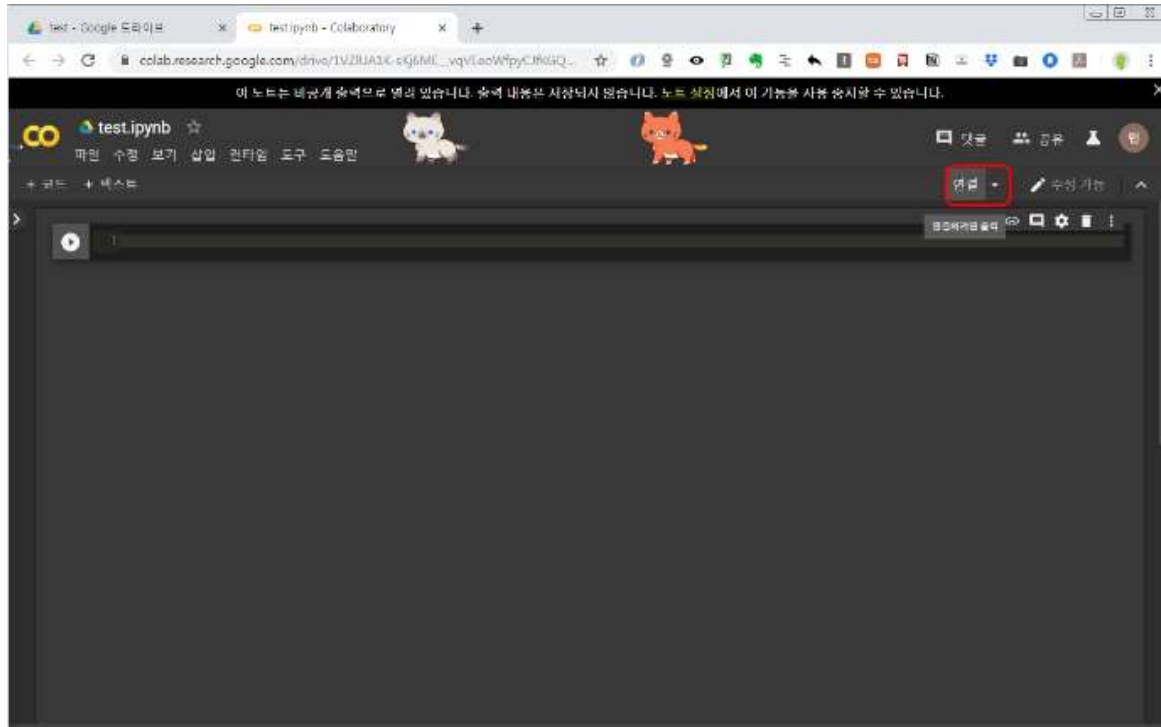
■ Colab 접속 및 사용



상단 메뉴 런타임 > 런타임 유형 변경
런타임 유형은 Python3를, 가속기는 GPU를 선택 > 저장을 클릭한다

클라우드 방식 딥러닝 개발 환경

■ Colab 접속 및 사용



연결 > 할당중.. > 연결중... > 초기화중.. 으로 텍스트가 변경되며, 최종 RAM, 디스크 사용량 막대그래프가 나올 것이다.
[>] 모양의 버튼을 클릭하면, 목차, 코드스니펫, 파일등의 기능을 활용할 수 있다.

클라우드 방식 딥러닝 개발 환경

■ Colab 서버 스펙 확인

```
from tensorflow.python.client import device_lib  
device_lib.list_local_devices()
```

```
import platform  
platform.platform()
```

클라우드 방식 딥러닝 개발 환경

■ 파일 처리

```
%%writefile test.py  
print('hello world!')
```

```
# test.py 실행시키기  
%run test.py
```

```
from google.colab import files  
# 브라우저에 다운로드 됨을 확인할 수 있다.  
files.download('test.py')
```

```
# [Cancel upload] 버튼을 클릭하여 잠시 멈춘 후 파일선택 버튼을 클릭하면 PC 내 파일을 선택할 수 있는  
다이얼로그 창이 뜬다.
```

```
# 리턴값을 받는 변수인 myupload라는 이름의 디렉토리가 생성된다.  
myupload = files.upload()
```

클라우드 방식 딥러닝 개발 환경

■ 구글 드라이브 연동

```
import os
print(os.getcwd())
!ls
# 실행시 등장하는 URL을 클릭하여 허용해주면 인증KEY가 나타난다. 복사하여 URL아래 빈칸에 붙여넣으면
# 마운트에 성공하게된다.
from google.colab import drive
drive.mount('/MyDrive')
# 마운트된 내 드라이브를 확인해보자
!ls
# 해당 드라이브로 이동
# 내 드라이브는 원격서버가 아니라 로컬서버로 간주하므로 명령어 실행시 앞단에 !를 붙이지 않는다.
cd MyDrive/My Drive

# 내드라이브의 전체 목록이 나타난다.
ls
# 특정파일을 가져오고 싶은 경우 다음과 같이 접근한다.
import pandas as pd
df = pd.read_csv("./MyDrive/test/test.csv")
```

클라우드 방식 딥러닝 개발 환경

■ Colab & Markdown

- 선택된 셀을 실행 : Ctrl + Enter
- 선택된 셀을 실행 후 다음 셀로 포커스 이동 : Shift + Enter
- 실행 후 다음줄로 이동 : Alt + Enter

- 엔터키 : 편집모드(Vi 편집기와 유사)
- ESC : 선택모드(Vi 편집기와 유사)
- 마크다운으로 전환 : Ctrl + M M
- 코드로 전환 : Ctrl + M Y
- 저장 : Ctrl + S

- 코드셀에 줄번호 부여 : Ctrl + M L
- 바로 윗줄에 셀 생성 : Ctrl + M A
- 바로 아랫줄에 셀 생성 : Ctrl + M B
- 셀 삭제 : Ctrl + M D

클라우드 방식 딥러닝 개발 환경

■ Colab & Markdown

- 셀 병합 : (shift를 누른 상태에서 병합을 원하는 셀들을 한번에 다중 선택 후), Shift + M
- 셀 분할 : (분기를 원하는 부분에 커서를 지정 후), Ctrl + Shift + -
- 코드가 오래 실행되어 멈추고 싶은경우 : Ctrl+ M + I
- 위 코드로도 멈추지 않고 작업을 완전종료하고 싶은 경우 : Ctrl+M+.

스탠드얼론 방식 딥러닝 개발 환경

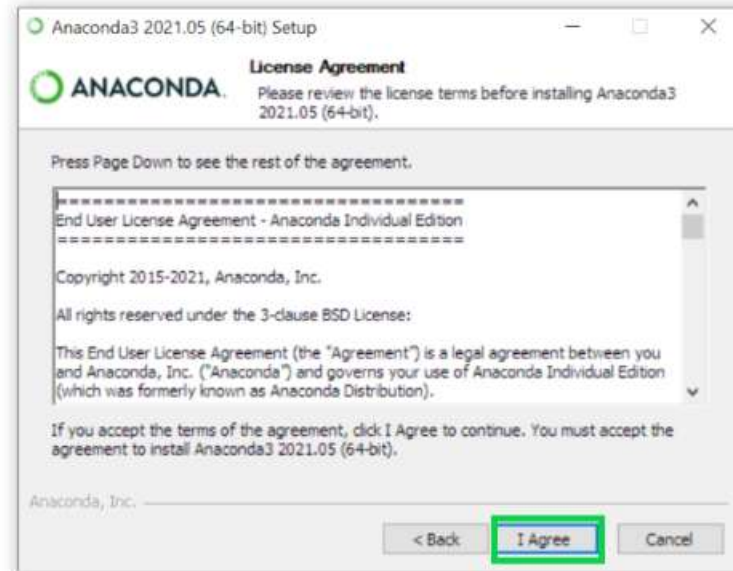
■ Anaconda Jupyter notebook 활용

- 아나콘다는 파이썬 인터프리터를 기반으로 하며, 데이터 분석 및 과학 작업에 자주 사용되는 다양한 라이브러리를 기본으로 제공합니다.
- NumPy, Pandas, Matplotlib, SciPy, Scikit-learn 등의 라이브러리를 포함하고 있어 데이터 분석 및 머신러닝 작업에 필요한 도구를 갖추고 있습니다.
- 아나콘다는 conda라는 패키지 관리자를 사용하여 쉽게 파이썬 패키지를 설치, 업데이트, 관리할 수 있습니다.
- 가상 환경을 생성하여 프로젝트별로 패키지를 격리된 환경에서 관리할 수 있습니다.
- 아나콘다는 다양한 운영체제(Windows, macOS, Linux)에서 동일하게 작동합니다.
- 아나콘다에는 대화형 개발 환경으로, 코드, 그래프, 문서 등을 하나의 문서에 통합하여 데이터 분석 작업을 수행할 수 있는 Jupyter Notebook이 기본으로 포함되어 있습니다.
- TensorFlow, Keras, PyTorch 등의 딥러닝 프레임워크, PySpark 등의 빅데이터 처리 도구, Plotly 등의 데이터 시각화 라이브러리 등을 아나콘다 환경에서 쉽게 설치하고 사용할 수 있습니다.

스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

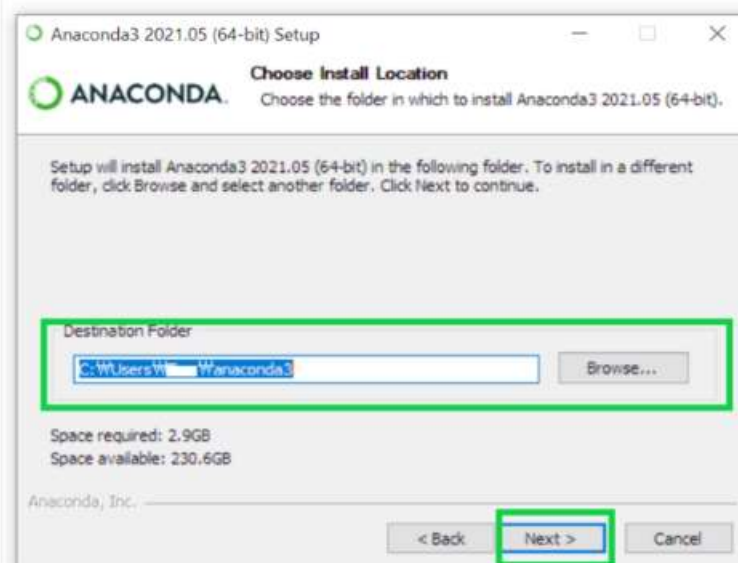
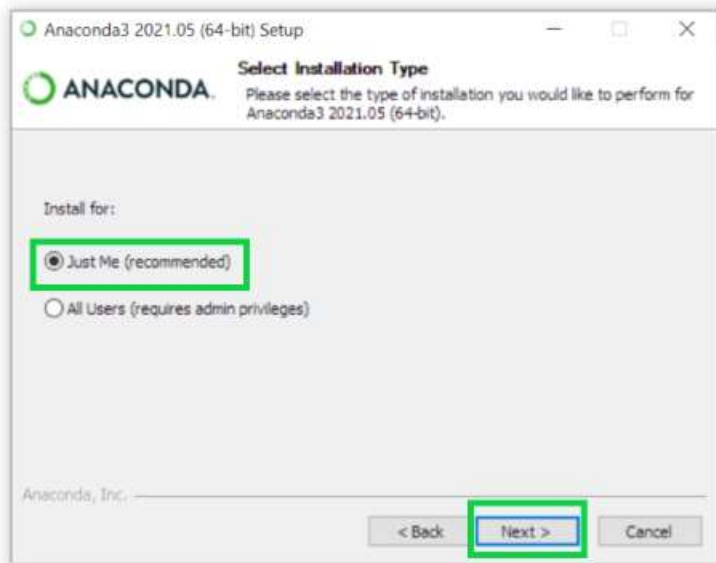
- 최신 아나콘다 버전을 다운로드합니다.
- <https://www.anaconda.com/products/individual>



스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

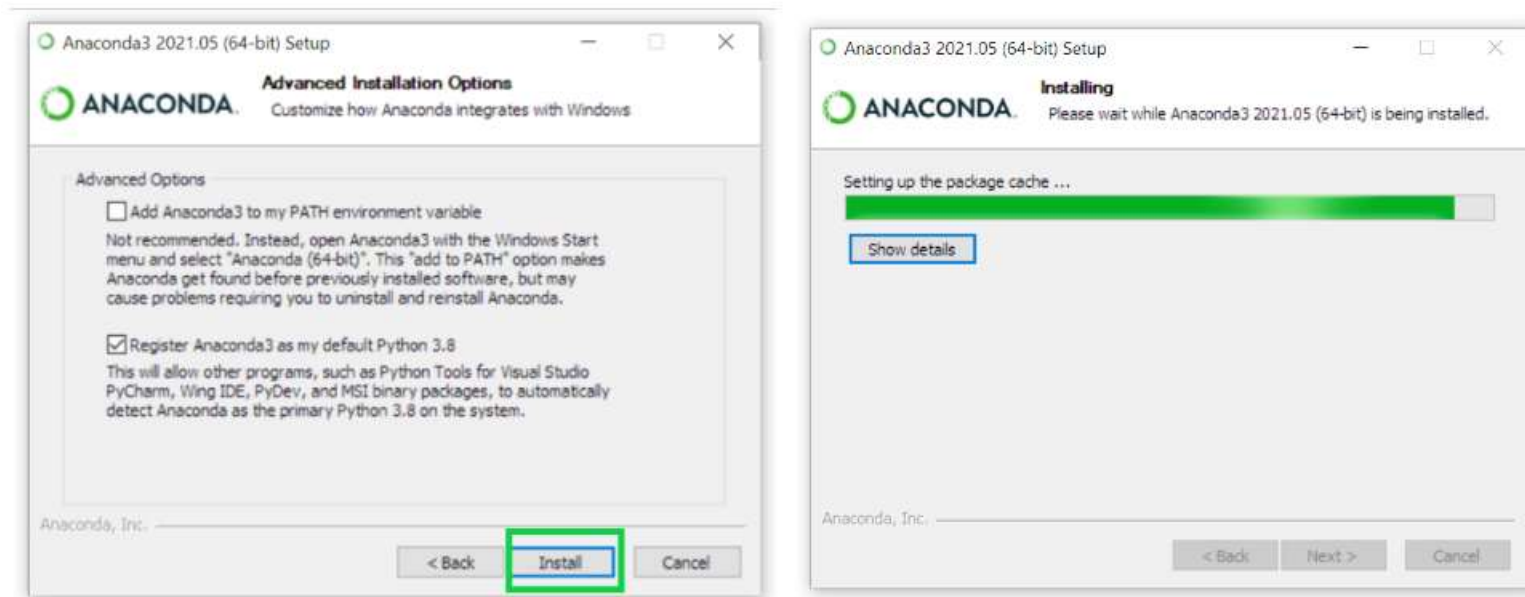
- "Just Me"설치를 선택하고 'Next'를 클릭합니다.
- 다운 받을 위치를 선택하고 'Next' 클릭



스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

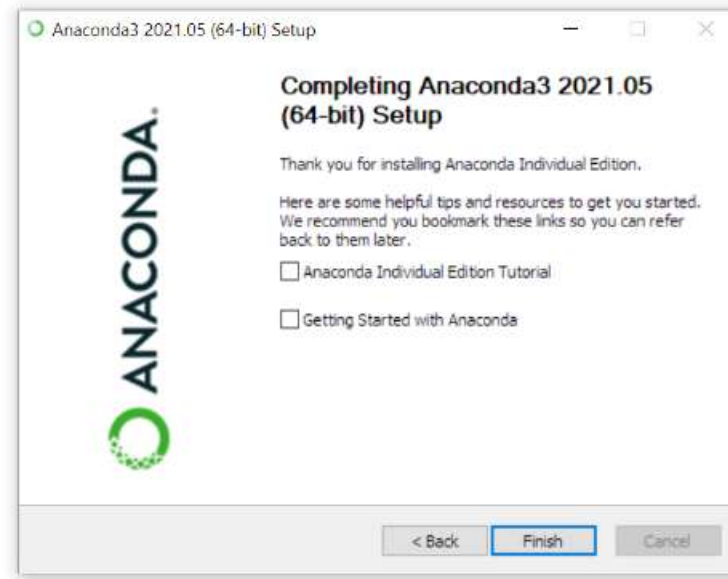
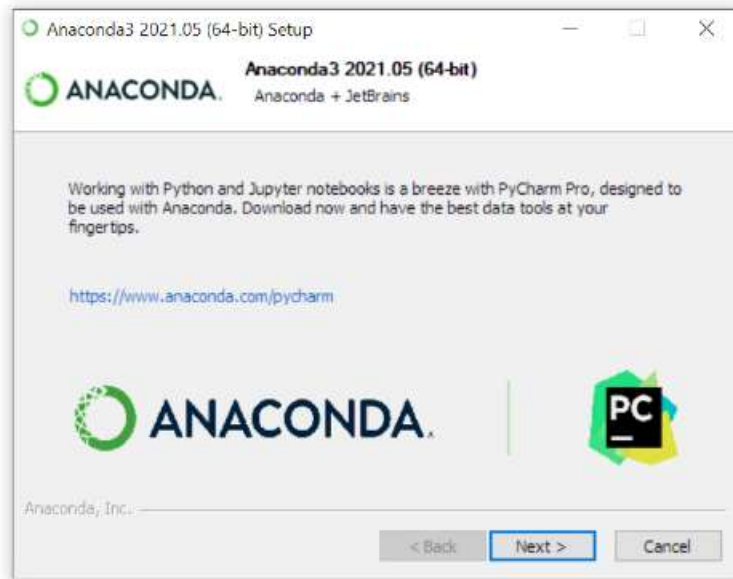
- Anaconda를 PATH 환경에 추가할지 여부를 선택하고, Anaconda를 기본 Python으로 등록할지 여부를 선택합니다.



스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

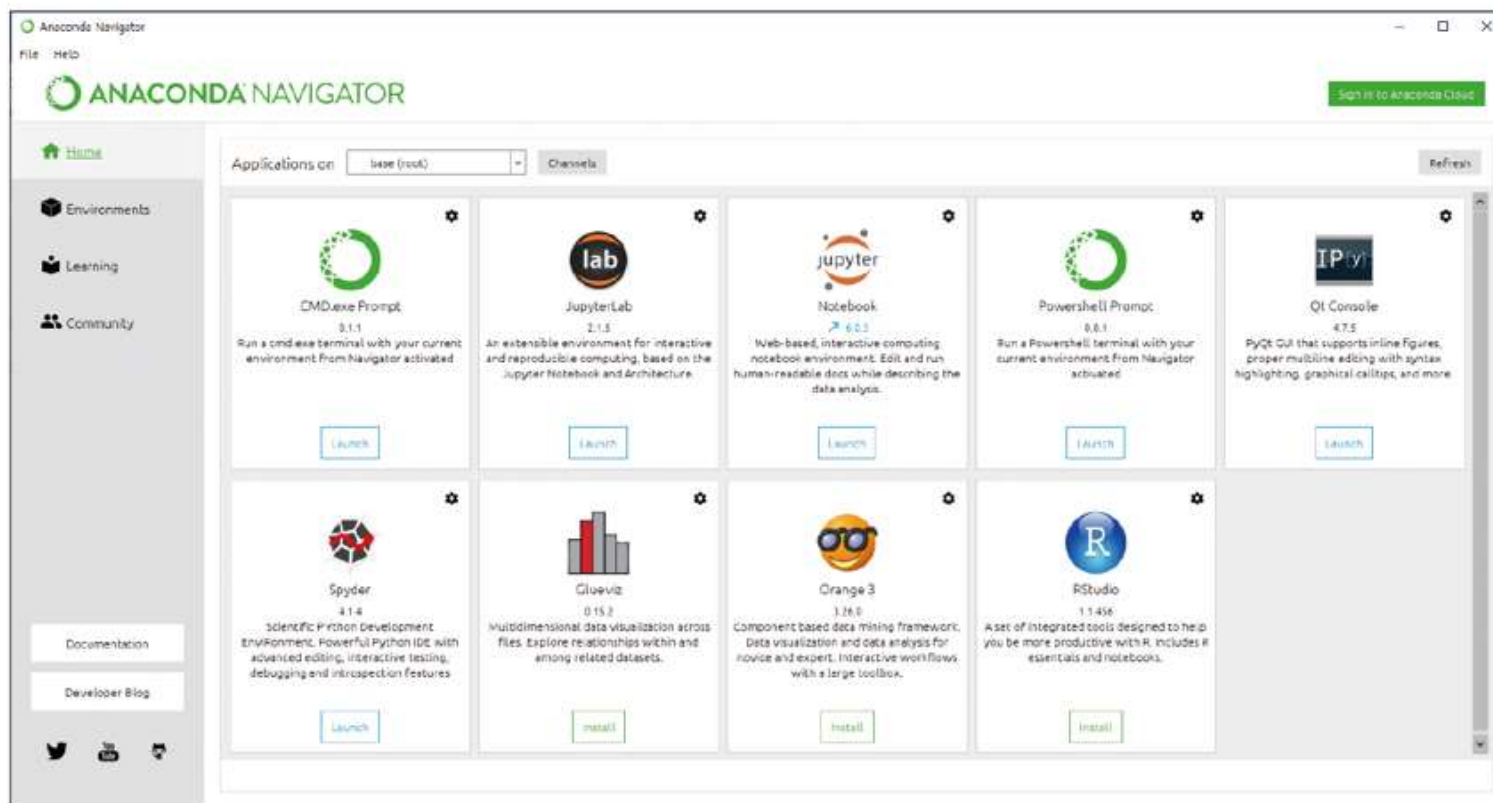
- (옵션) 파이참을 받고 싶으면 다운로드할 수 있습니다.
- 다운이 완료되면 'Finish'를 누르면 됩니다..



스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

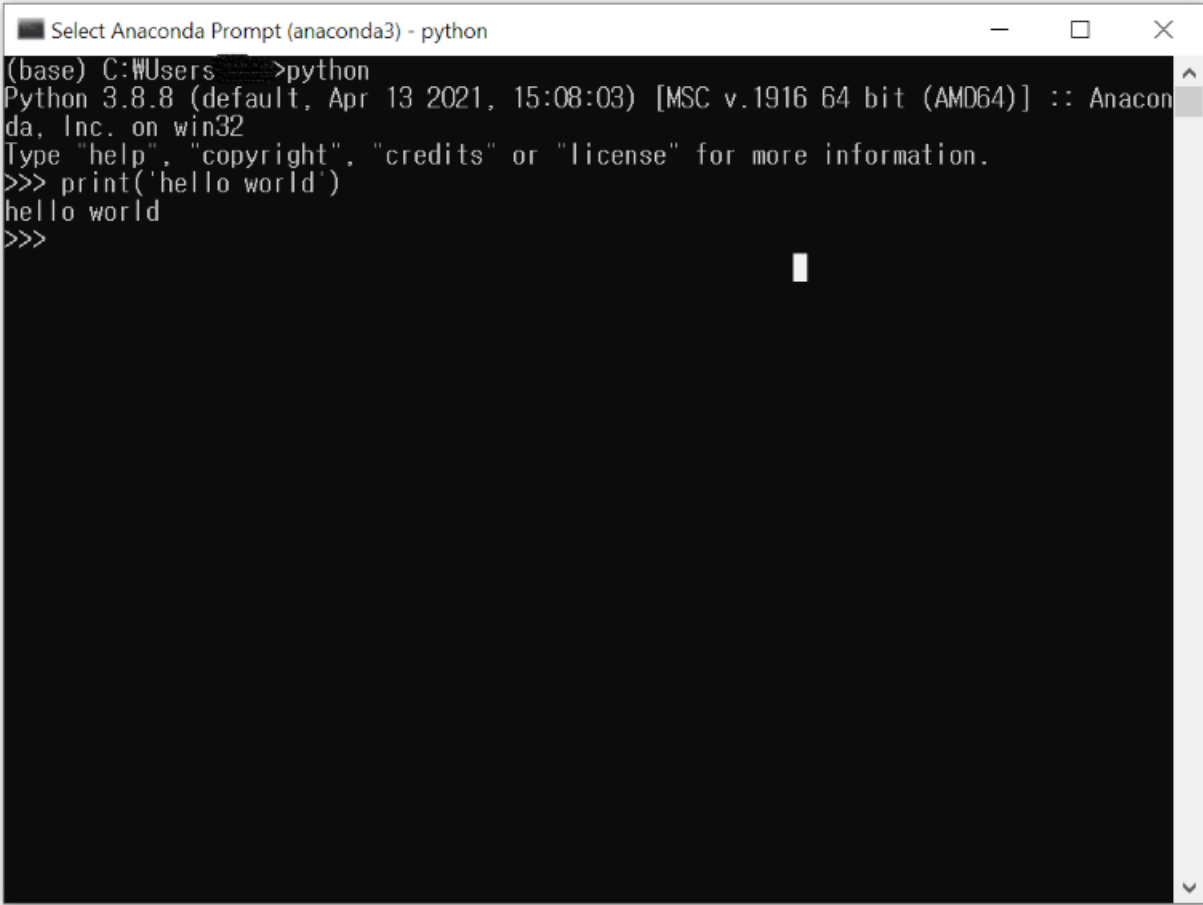
- 아나콘다 내비게이터 화면



스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

- 아나콘타 프롬프트 Anaconda Prompt에서 python을 입력하면 파이썬 버전과 아나콘다 버전이 뜨고 파이썬 셸이 실행됩니다.
- 아나콘다 프롬프트에서 콘다 버전 확인 : `conda -V` 나 `conda --version`
- 아나콘다 프롬프트에서 파이썬 버전을 확인 : `python -V` 나 `python --version`



```
Select Anaconda Prompt (anaconda3) - python
(base) C:\Users\...>python
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('hello world')
hello world
>>>
```


스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

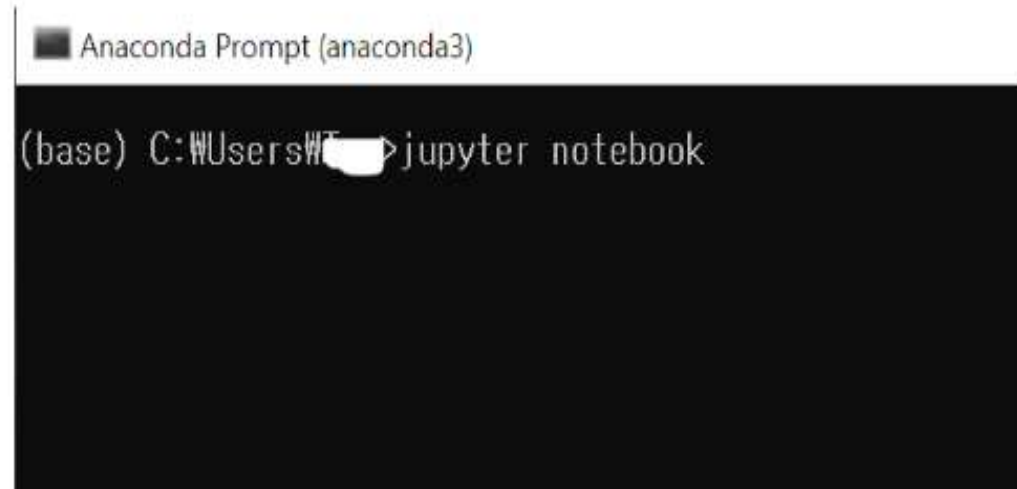
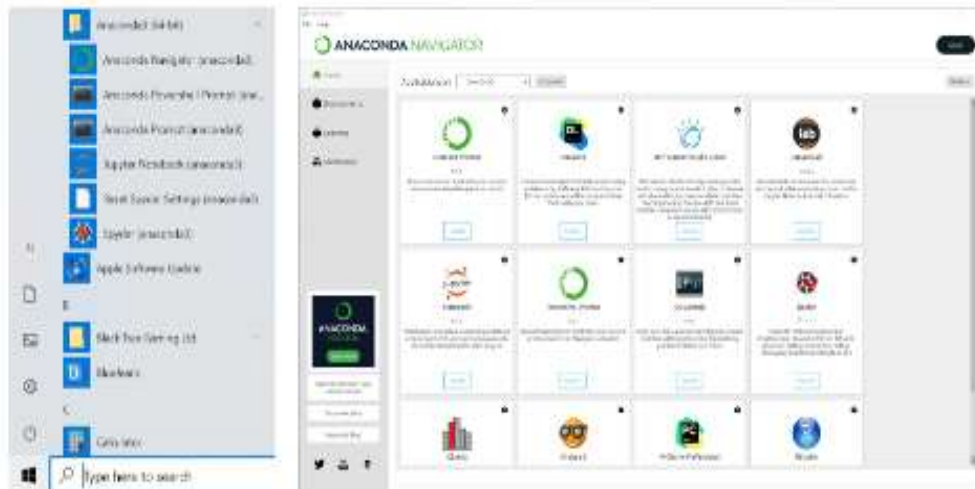
- 설치된 라이브러리를 확인 : `conda list`
- 패키지 설치 : `conda install <패키지이름>`
- 패키지 삭제 : `conda remove <패키지이름>`

```
(base) C:\Users\W...>conda list
# packages in environment at C:\Users\W...anaconda3:
#
# Name                                Version                                Build                                Channel
_ipyw_jlab_nb_ext_conf               0.1.0                                py38_0
alabaster                             0.7.12                               pyhd3eb1b0_0
anaconda                              2021.05                              py38_0
anaconda-client                       1.7.2                                py38_0
anaconda-navigator                    2.0.3                                py38_0
anaconda-project                      0.9.1                                pyhd3eb1b0_1
anyio                                 2.2.0                                py38haa95532_2
appdirs                              1.4.4                                py_0
argh                                  0.26.2                               py38_0
argon2-cffi                           20.1.0                               py38h2bbff1b_1
asn1crypto                           1.4.0                                py_0
astroid                               2.5                                  py38haa95532_1
astropy                               4.2.1                                py38h2bbff1b_1
async_generator                       1.10                                 pyhd3eb1b0_0
atomicwrites                          1.4.0                                py_0
attrs                                 20.3.0                               pyhd3eb1b0_0
autopep8                             1.5.6                                pyhd3eb1b0_0
babel                                 2.9.0                                pyhd3eb1b0_0
backcall                              0.2.0                                pyhd3eb1b0_0
backports                             1.0                                  pyhd3eb1b0_2
backports.funcutils_lru_cache         1.6.4                                pyhd3eb1b0_0
backports.shutil_get_terminal_size    1.0.0                                pyhd3eb1b0_3
backports.tempfile                    1.0                                  pyhd3eb1b0_1
backports.weakref                     1.0.post1                            py_1
bcrypt                                3.2.0                                py38he774522_0
beautifulsoup4                        4.9.3                                pyhaB47dfd_0
bitarray                              1.9.2                                py38h2bbff1b_1
bkcharts                              0.2                                  py38_0
black                                 19.10b0                              py_0
blas                                  1.0                                  mkl
bleach                                3.3.0                                pyhd3eb1b0_0
blosc                                 1.21.0                               h19a0ad4_0
bokeh                                 2.3.2                                py38haa95532_0
boto                                   2.49.0                               py38_0
bottleneck                            1.3.2                                py38h2a96729_1
brotli                                1.0.9                                ha925a31_2
brotlipy                              0.7.0                                py38h2bbff1b_1003
bzip2                                 1.0.8                                he774522_0
ca-certificates                       2021.4.13                            haa95532_1
certifi                               2020.12.5                            py38haa95532_0
cffi                                  1.14.5                               py38hcd4344a_0
chardet                               4.0.0                                py38haa95532_1003
```

스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

- 주피터 노트북은 무료, 오픈 소스 도구로 웹 브라우저에서 파이썬 코드를 작성하고 실행할 수 있으면 데이터 분석에 많이 쓰이는 도구입니다.
- 시작 > Anaconda3 > Jupyter Notebook
- 시작 > Anaconda3 > Anaconda Navigator > Jupyter Notebook
- 시작 > Anaconda3 > Anaconda Prompt > jupyter notebook 입력



스탠드얼론 방식 딥러닝 개발 환경

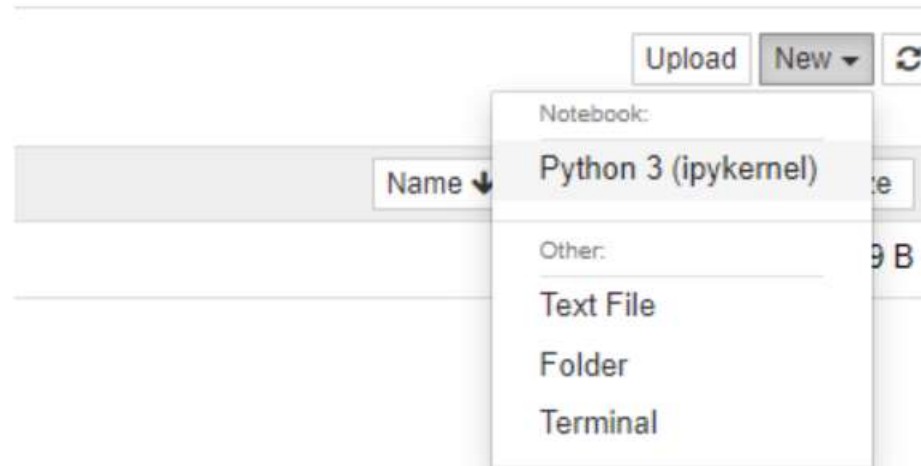
■ Anaconda Jupyter notebook 활용

- 가상환경 생성 명령 : `conda create -n` 가상환경 이름 `python=파이썬 버전`
예) `$ conda create -n velog python=3.8`
- 가상환경 활성화 & 비활성화 : `conda activate` 가상환경 이름
예) `$ conda activate velog`
`$ conda deactivate`
- 가상환경 목록 보기 명령 : `conda info --envs` 또는 `conda env list`
- 가상환경 삭제 명령 : `conda env remove -n` 가상환경 이름

스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

- Cell 생성 및 실행
- 코드를 입력하는 창을 Cell이라고 하며, 실행하는 단축키는 Ctrl+Enter와 Shift+Enter 입니다.
- 명령 모드 (ESC) : 셀에 대한 수정을 중단하고 해당 셀에 다른 jupyter notebook의 단축키나 기능을 이용할 때 사용한다.
- 편집 모드 (ENTER) : 셀의 내용을 수정하고 싶을 때 사용.
- MARKDOWN (명령모드 상태 + M) : 셀을 마크다운 형식으로 쓸수 있도록 해준다.
- CODE(명령모드 상태 + Y) : 셀을 설정한 언어로 사용할 수 있는 상태로 변환해 준다.



스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

- 셀 실행 및 명령모드(SHIFT + ENTER) : 해당 셀을 실행하고 밑에 실행할 셀이 없을 경우 생성 및 명령모드로 전환
- 셀 실행 및 편집모드 (CTRL + ENTER) : 해당 셀을 실행하고 명령모드로 전환
- 셀 실행 및 셀 생성 및 새로운 셀 편집모드 (ALT + ENTER) : 해당 셀 실행, 무조건 셀 생성, 생성 셀 편집모드 전환
- 셀 삭제 (명령모드 + X , 명령모드 + D, D) : 해당 셀 삭제
- 셀 되돌리기 (명령모드 + Z) : 전 상태로 되돌린다.
- 셀 위로 생성 (명령모드 + a) : 명령모드 상태 셀에서 위로 셀을 생성 (Above)
- 셀 아래로 생성 (명령모드 + b) : 명령모드 상태 셀에서 아래로 셀을 생성 (Below)

스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

- Magic Command : 주피터 노트북에서 제공해주는 특별 동작 명령어.
- 셀 스타일이 코드 스타일일 경우 사용이 가능하다.
- % : 한 라인의 매직 커맨드를 동작시킨다.
- %% : 셀 단위의 매직 커맨드를 동작시킨다.
- Magic Command List
 - pwd : 현 주피터 노트북 파일의 경로
 - ls : 현 폴더(Directory)의 파일 리스트
 - whos : 현 파일에서 메모리에 할당된 변수 리스트
 - reset : 현 파일에서 메모리에 할당된 변수 리스트를 모두 삭제
- Shell Command : 주피터 노트북에서 설정된 쉘 환경의 명령을 사용할 수 있다.
명령어 앞에 !를 붙이면 된다.

스탠드얼론 방식 딥러닝 개발 환경

■ Anaconda Jupyter notebook 활용

- 스파이더(Spyder) 통합 개발 환경

