

2021 인공지능 소수전공

2차시: 파이썬 기초

2021.07.19 19:30~20:15

Seokhwan Yang

1. 파이썬 소개
2. 개발환경 설정
3. Jupyter Notebook
4. Colab



1. 파이썬 소개

- 파이썬의 기원

- 1990년 네덜란드 암스테르담, 귀도 반 로섬에 의해 개발, 1991년 발표된 인터프리터형 언어
- 왜 개발했는가?



“1989년 12월, 저는 크리스마스 주중에 저의 ‘취미’가 될 만한 프로그램을 찾고 있었습니다.”

1999년, DARPA에게 Computer Programming for Everybody라는 자금 제안을 제출하여 Python에 대한 나의 목표를 정의하였습니다. 당연히 무료이며 오픈 소스이므로 누구나 개발할 수 있습니다.

평이한 영어로 이해할 수 있는 코드, 일상적인 업무에 대한 적합성과 짧은 개발 시간 등 장점을 기반으로 파이썬은 대중적인 프로그래밍 언어가 되었습니다.

(2008, 구글 개발자 컨퍼런스에서)

1. 파이썬 소개

• 파이썬의 특징

- 쉽게 익힐 수 있는 프로그래밍 언어이다(문법이 쉽다)
- 간결하다
- 강력하다
- 무료이다
- 개발 속도가 빠르다



정말 쉬울까?

1. 파이썬 소개

• 파이썬이 쉽다는 말은 누가 했을까?

귀도 반 로섬: 취미로 프로그래밍 언어를 만들 정도의 엄청난 능력자

쉽다는 기준이
일반인과 다름

기존 개발자: 개발 경험이 풍부하므로 C/C++/C#, Java 등과
비교하면 당연히 쉬움

쉽다는 기준이
일반인과 다름

영어권 일반인: "평이한 영어로 이해할 수 있는 코드"가
개발 컨셉 → 아무래도 접근하기가 용이함

쉽다는 기준이
한국인과 다름

비 영어권 일반인: "**뭔 소린지 하나도 모르겠다!!!**"
라는 반응이 생각보다 많음

진짜 쉽나???

1. 파이썬 소개

- 입문자에게 파이썬은...

- 파이썬의 특징이자 장점인 동적 언어 → 입문자에게는 의미를 알 수 없는 특징
- 자료형을 신경 쓰지 않아도 된다 → 나중에 꼬이기 시작하면 답이 없음
- 객체지향, 절차지향, 함수형 언어의 특징을 모두 지원한다
→ C/C++/C#/Java 등 다른 언어의 특징을 모두 신경 써야 할 지도 모른다
- 엄격한 들여쓰기, 탭, 스페이스...
→ 알려진 것과 다르게 코드의 형태를 매우 엄격하게 관리한다
- 등등...

1. 파이썬 소개

그럼에도 불구하고 파이썬의 접근성은 타 언어보다는 용이하다

시작부터 겁을 주는 이유는?

이해가 잘 가지 않더라도 나 혼자만 뒤처지는 것이 아니다!!

겁먹을 필요 없다!! 해 보면 다른 것보다는 쉽다!!

1. 파이썬 소개

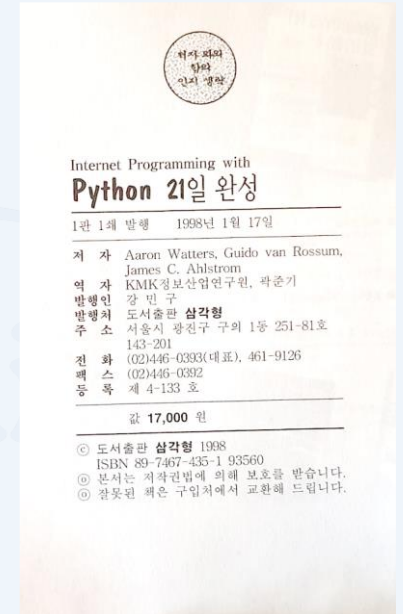
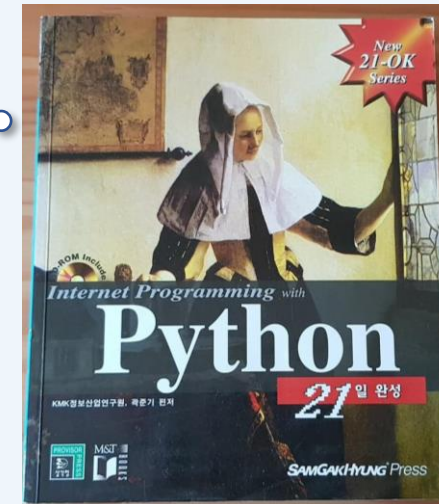
• 파이썬의 언어적 특징

- 플랫폼 독립적인 언어 : 어떤 운영체제든 상관없이 사용할 수 있는 언어 → **글쎄..**
- 인터프리터 언어 : 컴파일러 언어와 달리 소스코드 자체가 바로 실행되는 특징이 있는 언어. 이로 인해 속도는 느리지만, 굉장히 간편하게 사용할 수 있다.
- 객체 지향 언어 : 해당 프로그램이 해결해야 할 문제의 구성요소를 요소 별로 정의한 뒤 각 요소의 기능(메서드)과 정보(속성)를 정의하여 요소들을 결합하고, 프로그램을 작성하는 방식 → **클래스 지원 언어**
- 동적 타이핑 언어 : 프로그램의 실행 시점에서 각 프로그램 변수의 타입을 결정하는 언어 → **코딩할 때 신경 쓰지 않아도 된다**

1. 파이썬 소개

- AI, 데이터 과학분야에서는 왜 파이썬을 많이 사용할까?
- 1991년에 발표된 언어지만 국내에선 그다지 주목받지 못해..

책장에서 발견한 옛 파이썬 도서
대학생때 사 놓고 거의 안봤음...
(1998.1.17 발행)



- 알파고 이후, AI에 대한 관심이 급증하면서 일단 **외국의 트렌드를 따라 감**
- 그럼 외국에서는 왜?

- 데이터 과학, AI 분야에 대한 파이썬의 강점

- 개발 속도, 개발의 용이성 등 다양한 특징
- 언어 자체적으로 64Bit 이상의 매우 큰 정수 연산 지원 → 이·공학 분야에서 많이 활용
- 매우 다양한 기능의 라이브러리 제공(특히 이·공학 분야를 위한 강력한 기능 제공)
- Numpy, Pandas, SciPy, Scikit-Learn, Matplotlib 등 복잡한 수치와 시각화, 큰 데이터에 특화된 라이브러리를 포함한 매우 다양한 기능의 라이브러리 제공
- 이·공학 분야의 경우
 - 수많은 데이터를 기반으로 특정한 모델의 연구 개발 및 실험 지속, 성능 증명이 필수
 - 인터프리터형 언어의 특징 + 다양한 라이브러리 → 연구 과정에서 요구되는 노력 감소 지원
- 이러한 이유들로 인해 채택됨

1. 파이썬 소개

• 파이썬의 단점

- **느리다: 인터프리터형 언어**이므로 코드를 한 줄씩 읽고 해석하여 실행
 - Cpython 확장 모듈: 개발된 파이썬 모듈을 C/C++ 루틴 호출 연동 등을 통해 성능향상
 - Cython: Cpython 확장 모듈을 쉽게 생성하도록 지원하는 컴파일 언어
- 디자인, 환경 등에 대한 제약: 개선을 위한 다양한 라이브러리 개발 중
- GUI 지원 취약: QtPy, Tkinter 등 라이브러리 및 툴킷 지원으로 보완 중

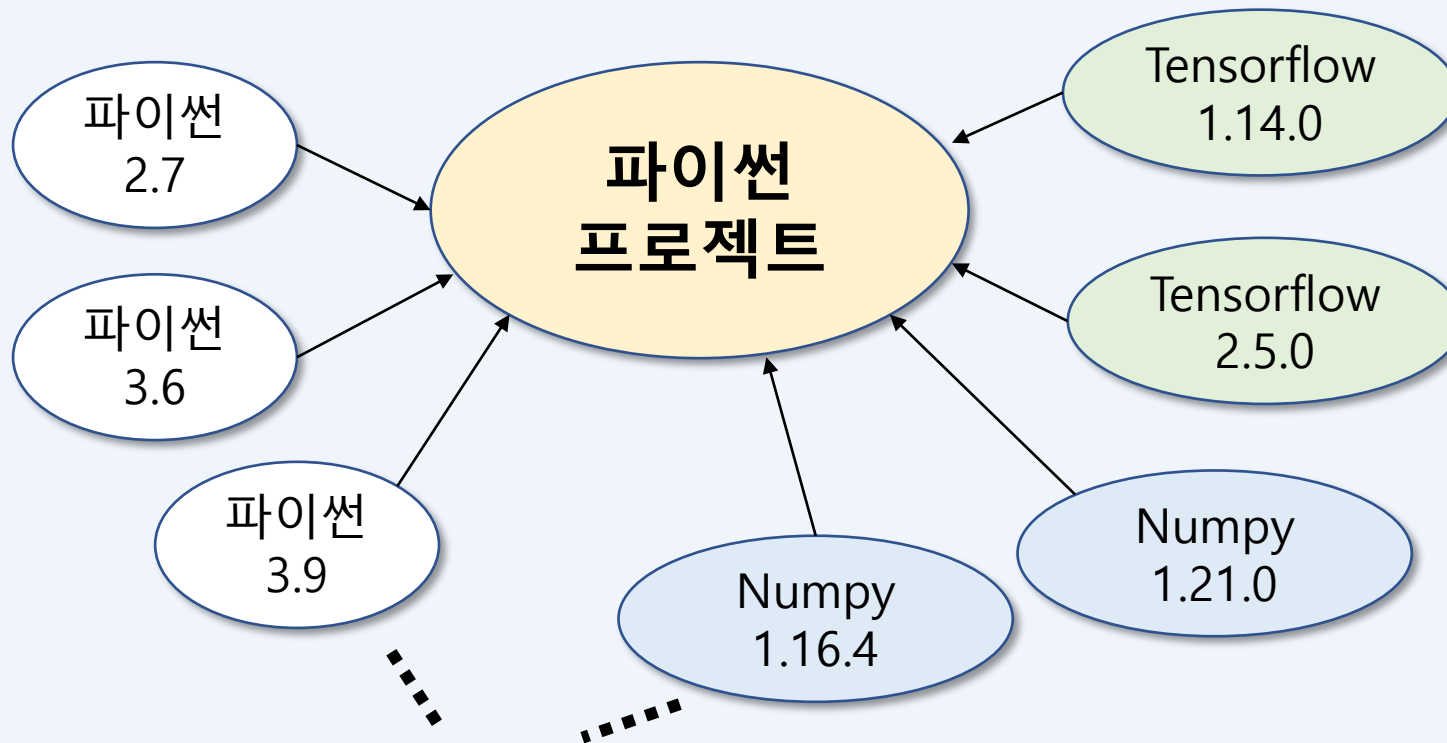
인터프리터형 언어: 장점인 동시에 단점

2. 개발환경 설정

• 파이썬 프로젝트는...

다양한 파이썬 버전

다양한 버전의 다양한 라이브러리



프로젝트의 특징에 따라

다양한 환경을 가짐

버전 별 호환성 문제 다수

2. 개발환경 설정

- 각 프로젝트마다 다른 버전의 파이썬과 모듈을 사용하는 경우 많음
→ 가상환경 구축 권장
- 가상환경을 지원하는 도구
 - VirtualEnv: 구버전의 파이썬에서부터 많이 사용되어 온 도구
 - Venv: 파이썬 3.4 부터 기본적으로 포함된 도구
 - Anaconda: 최근 가장 인기있는 파이썬의 배포 패키지

2. 개발환경 설정

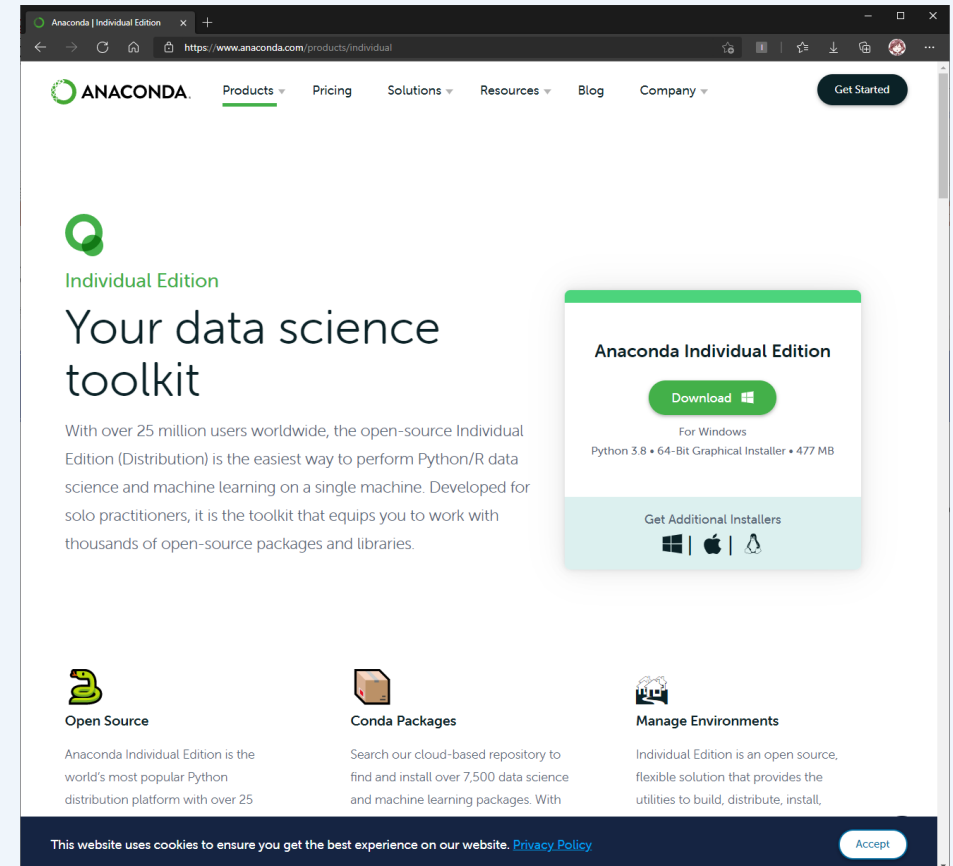
- Anaconda를 이용한 가상환경 구축 (for Windows)

- Anaconda 다운로드

- <https://www.anaconda.com/products/individual>

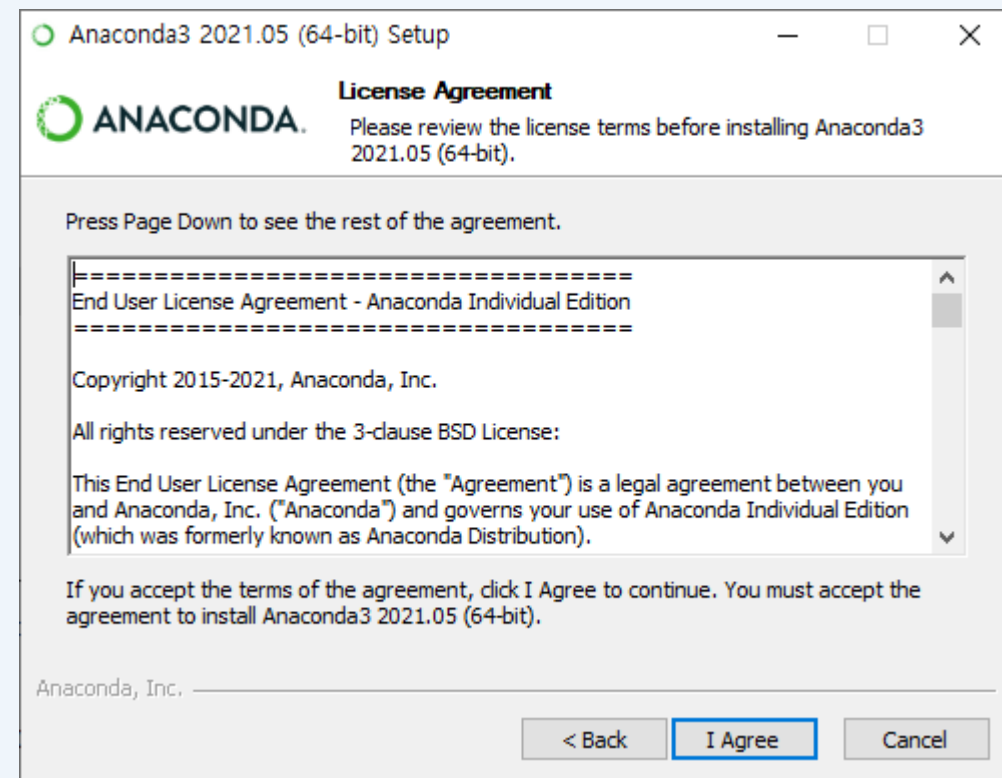
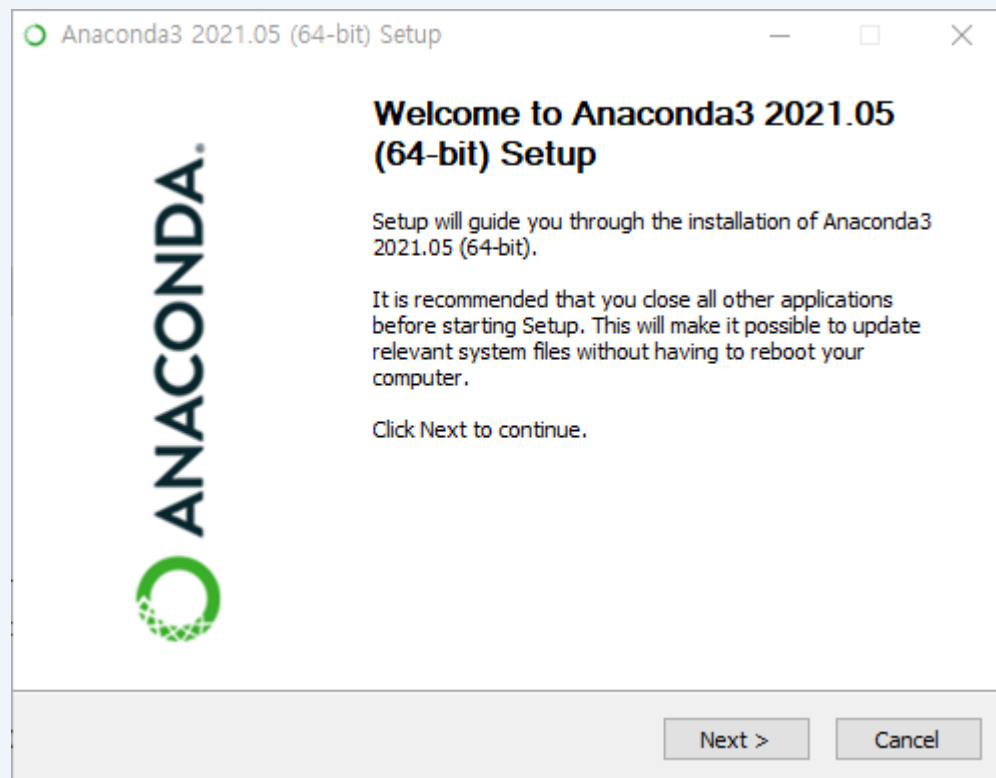
페이지에서 다운로드

(Individual 버전은 Open Source 버전)



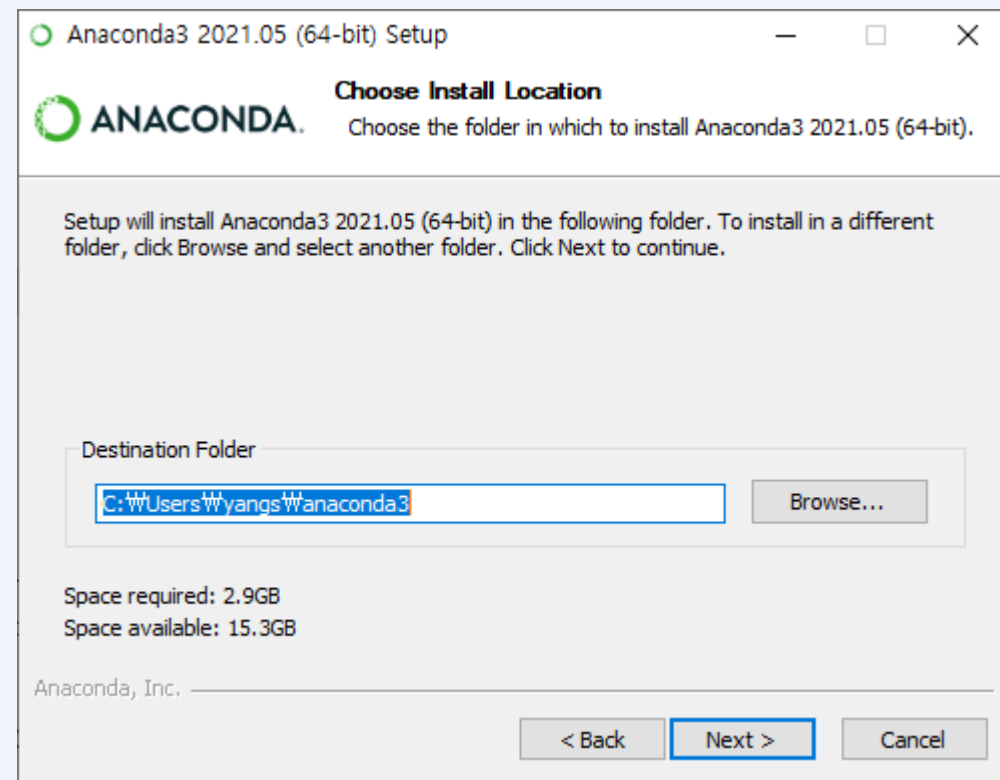
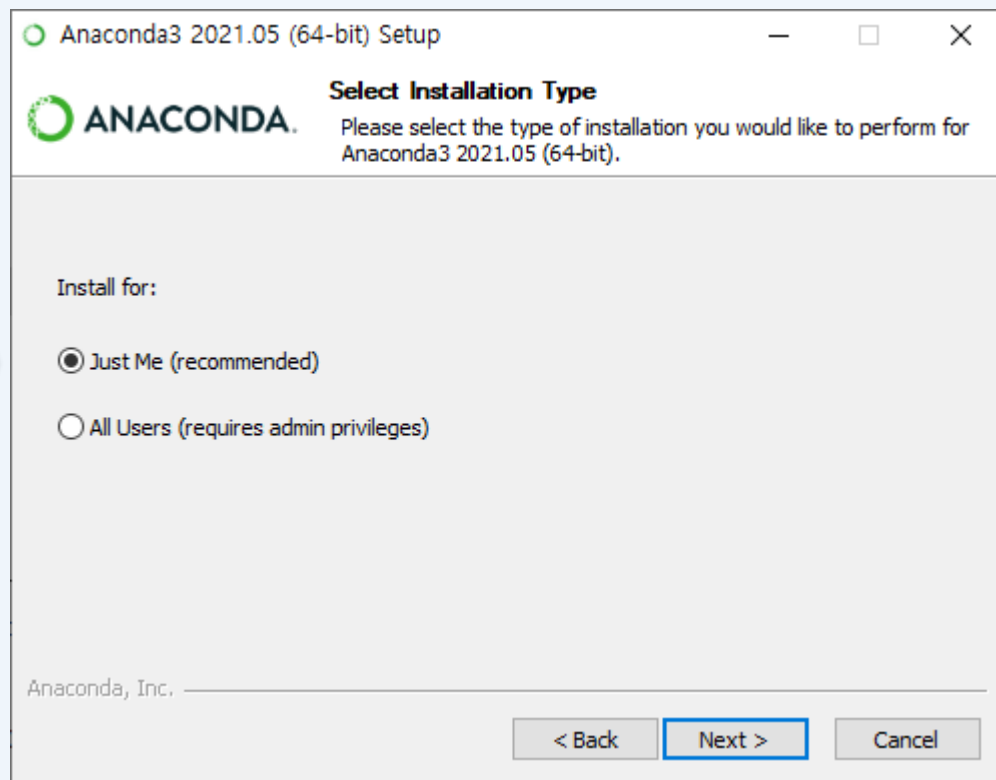
2. 개발환경 설정

• Anaconda 설치 프로그램 실행 (1)



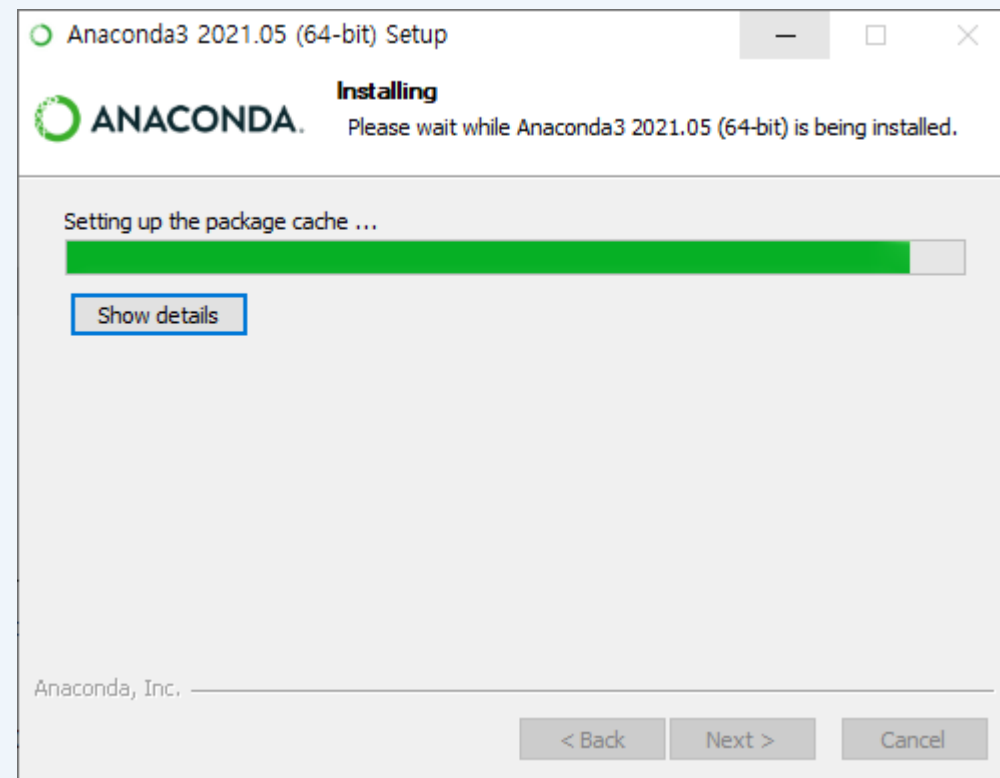
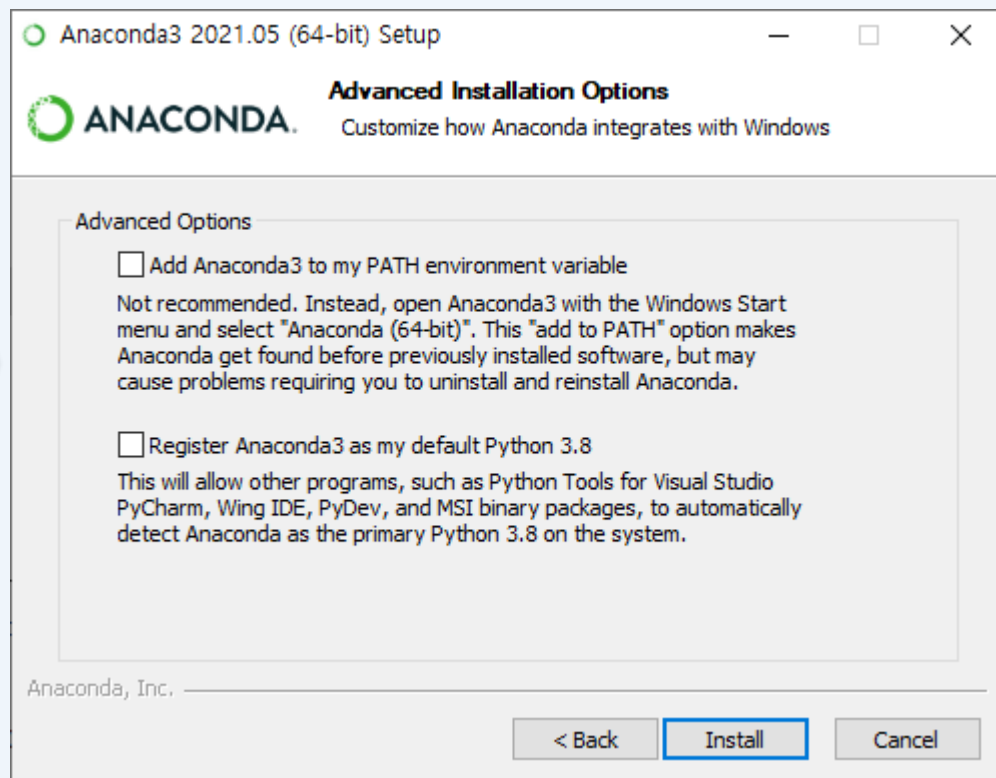
2. 개발환경 설정

• Anaconda 설치 프로그램 실행 (2)



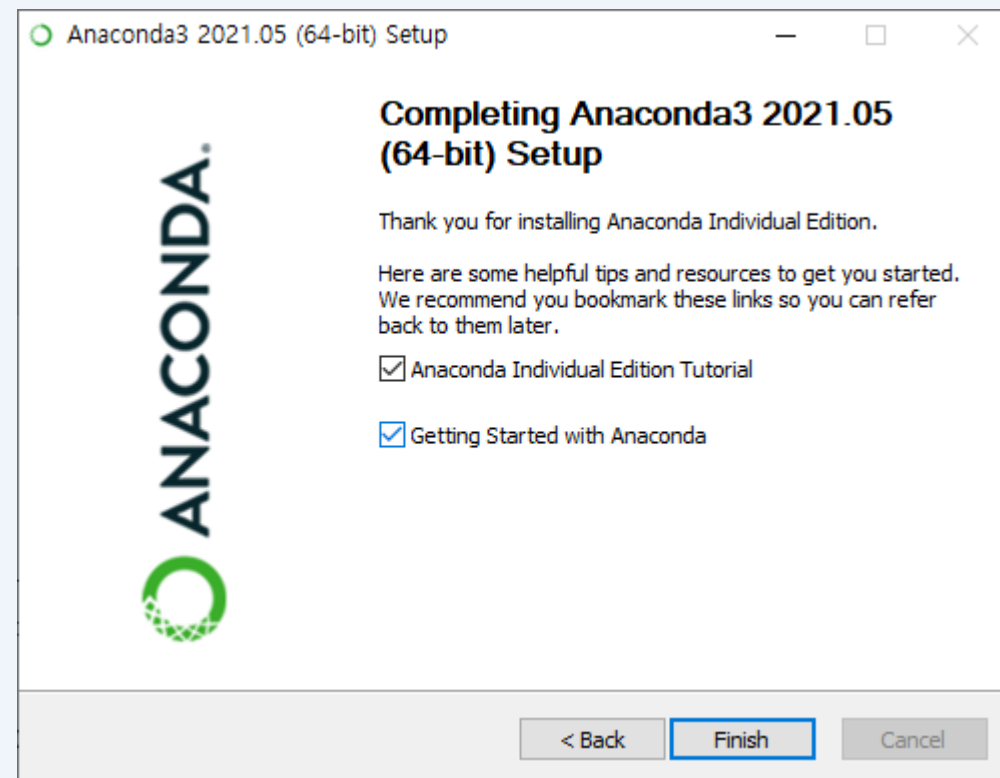
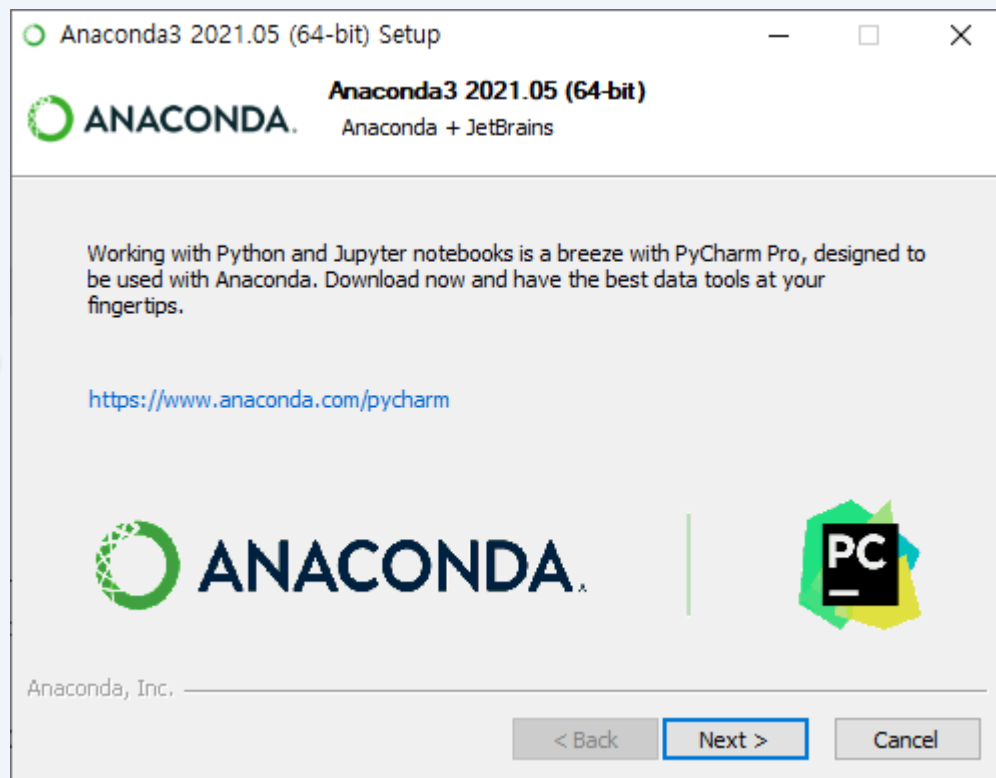
2. 개발환경 설정

• Anaconda 설치 프로그램 실행 (3)



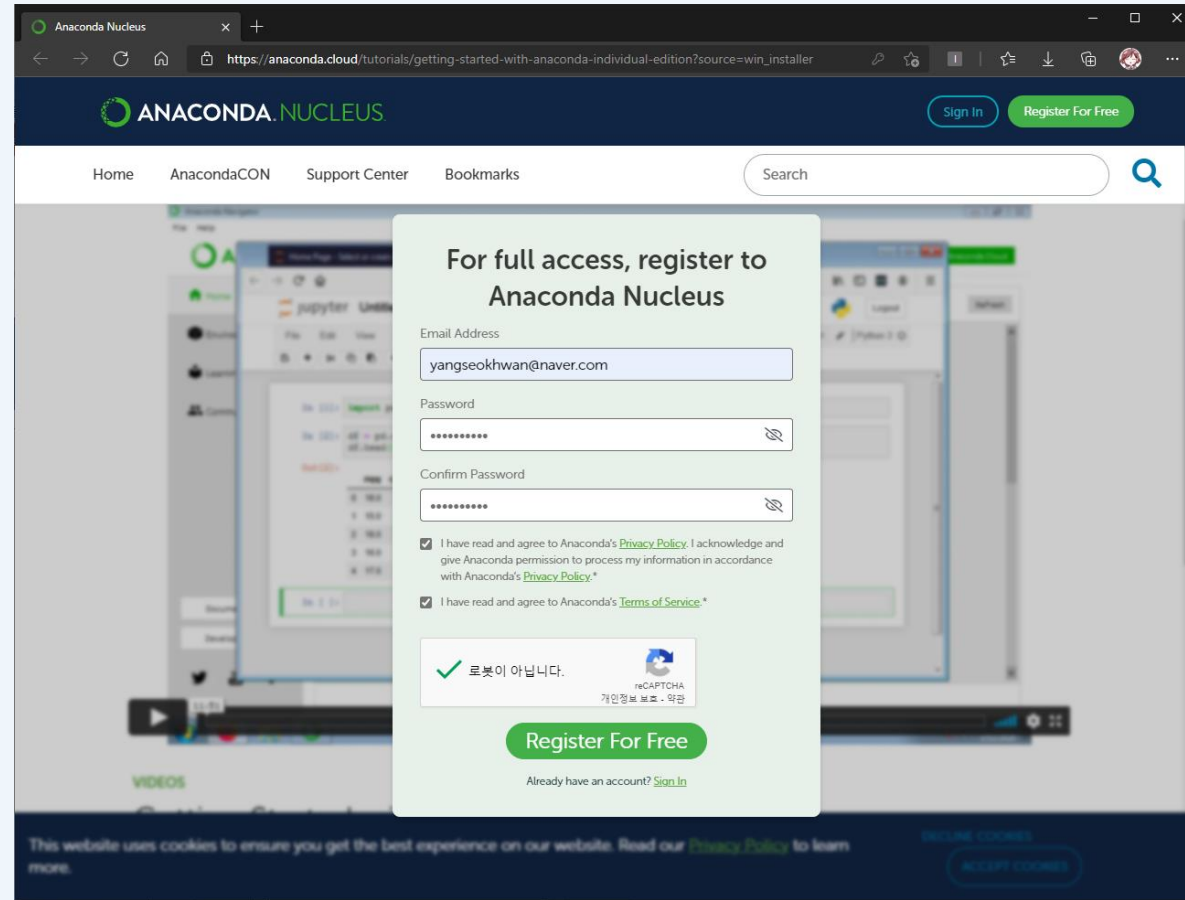
2. 개발환경 설정

• Anaconda 설치 프로그램 실행 (4)



2. 개발환경 설정

- Anaconda 설치 프로그램 실행 (5)



The screenshot shows the Anaconda Nucleus registration page. The browser address bar displays the URL: https://anaconda.cloud/tutorials/getting-started-with-anaconda-individual-edition?source=win_installer. The page header includes the Anaconda Nucleus logo, a 'Sign In' button, and a 'Register For Free' button. The navigation bar contains links for Home, AnacondaCON, Support Center, and Bookmarks, along with a search bar. The main content area features a registration form with the following fields and options:

- Title:** For full access, register to Anaconda Nucleus
- Email Address:**
- Password:**
- Confirm Password:**
- Agreements:**
 - ☒ I have read and agree to Anaconda's [Privacy Policy](#). I acknowledge and give Anaconda permission to process my information in accordance with Anaconda's [Privacy Policy](#).
 - ☒ I have read and agree to Anaconda's [Terms of Service](#).
- reCAPTCHA:** A checkbox with the text "로봇이 아닙니다." (I am not a robot.) and a reCAPTCHA logo.
- Buttons:** A large green 'Register For Free' button and a smaller 'Sign In' link below it.

At the bottom of the page, there is a cookie consent banner that reads: "This website uses cookies to ensure you get the best experience on our website. Read our [Privacy Policy](#) to learn more." with 'ACCEPT COOKIES' and 'DECLINE COOKIES' buttons.

2. 개발환경 설정

- Anaconda 기반 가상환경 만들기

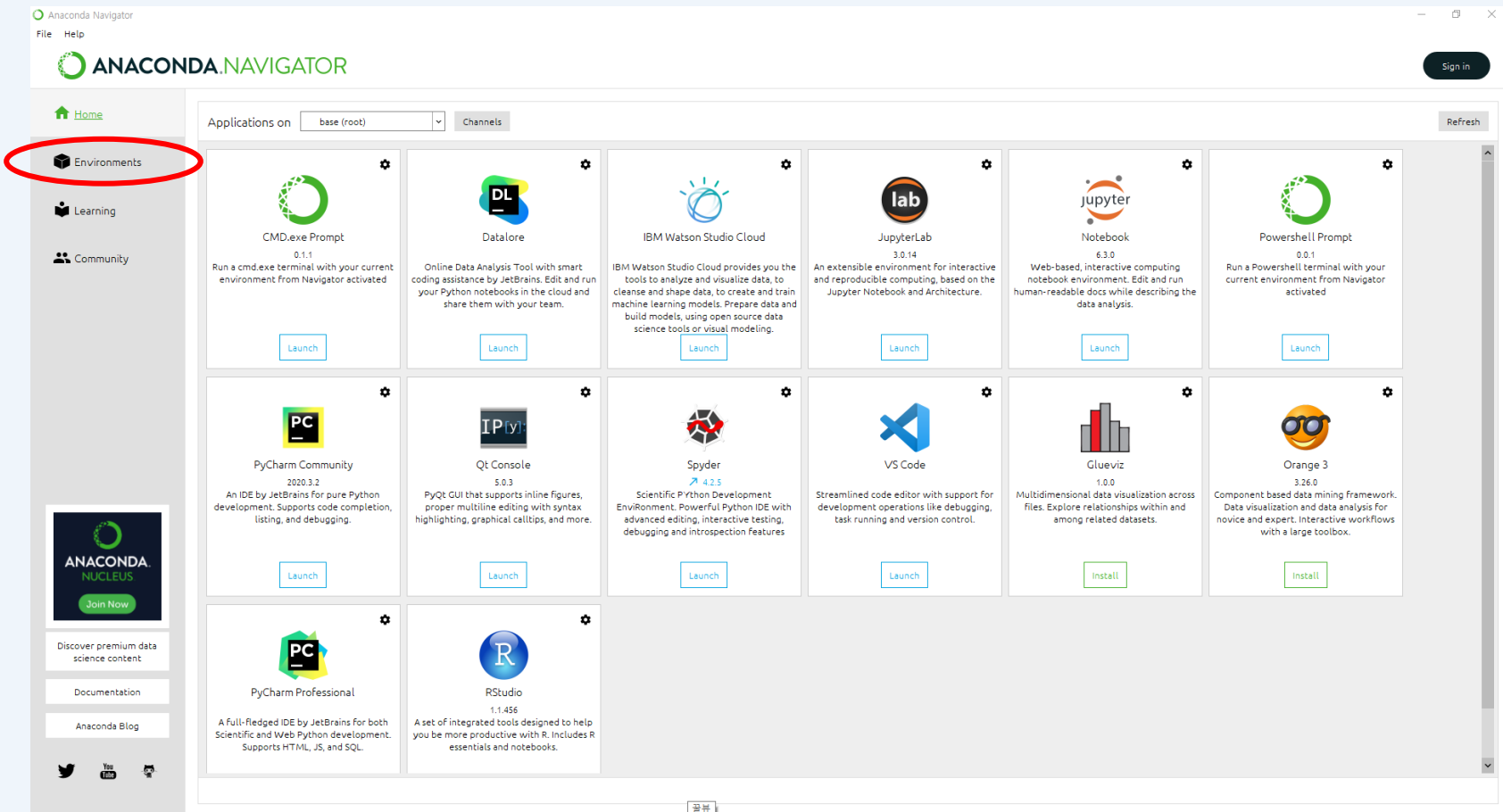
Anaconda Navigator
실행



2. 개발환경 설정

• Anaconda 기반 가상환경 만들기

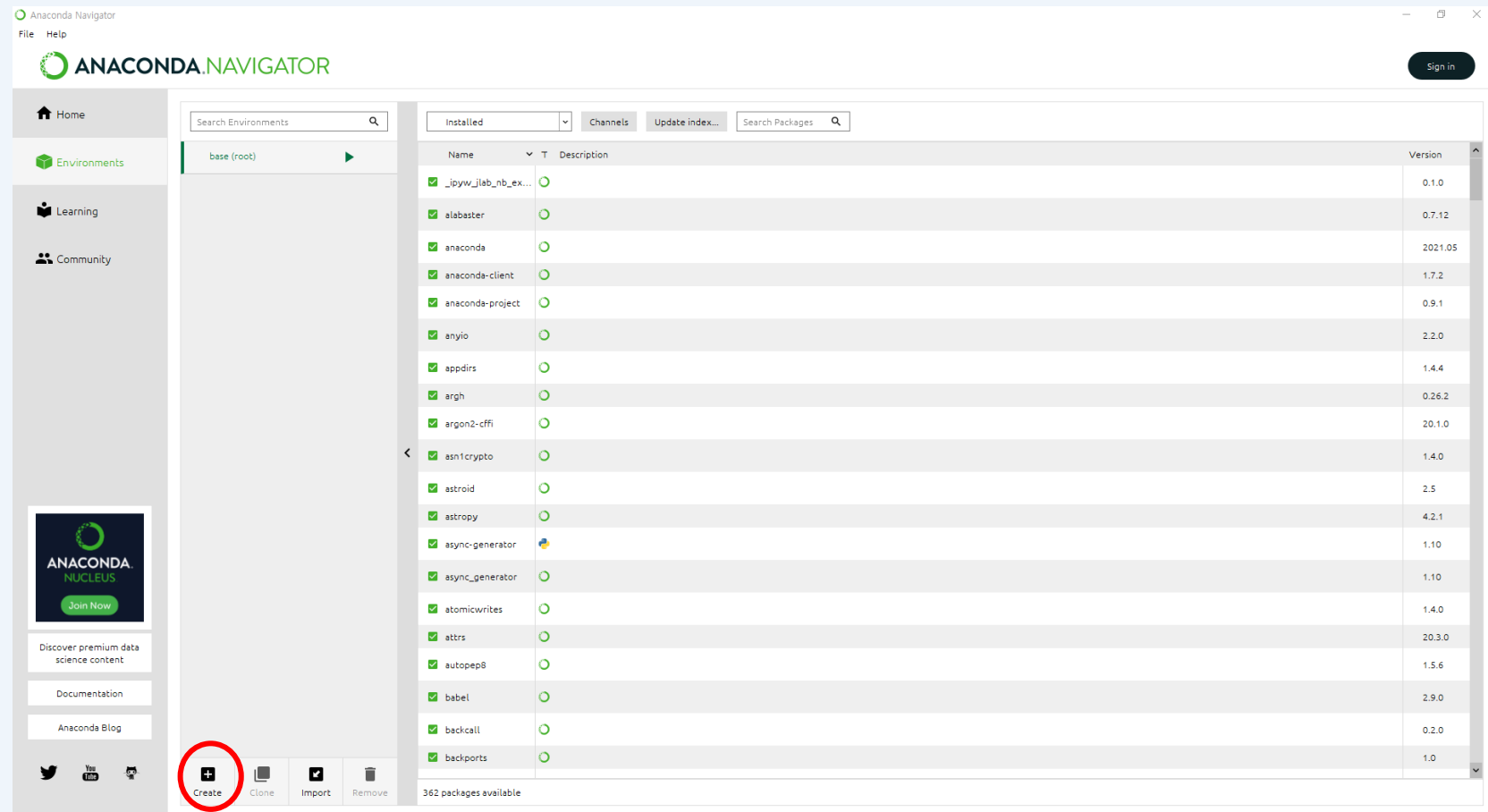
Environments
선택



2. 개발환경 설정

• Anaconda 기반 가상환경 만들기

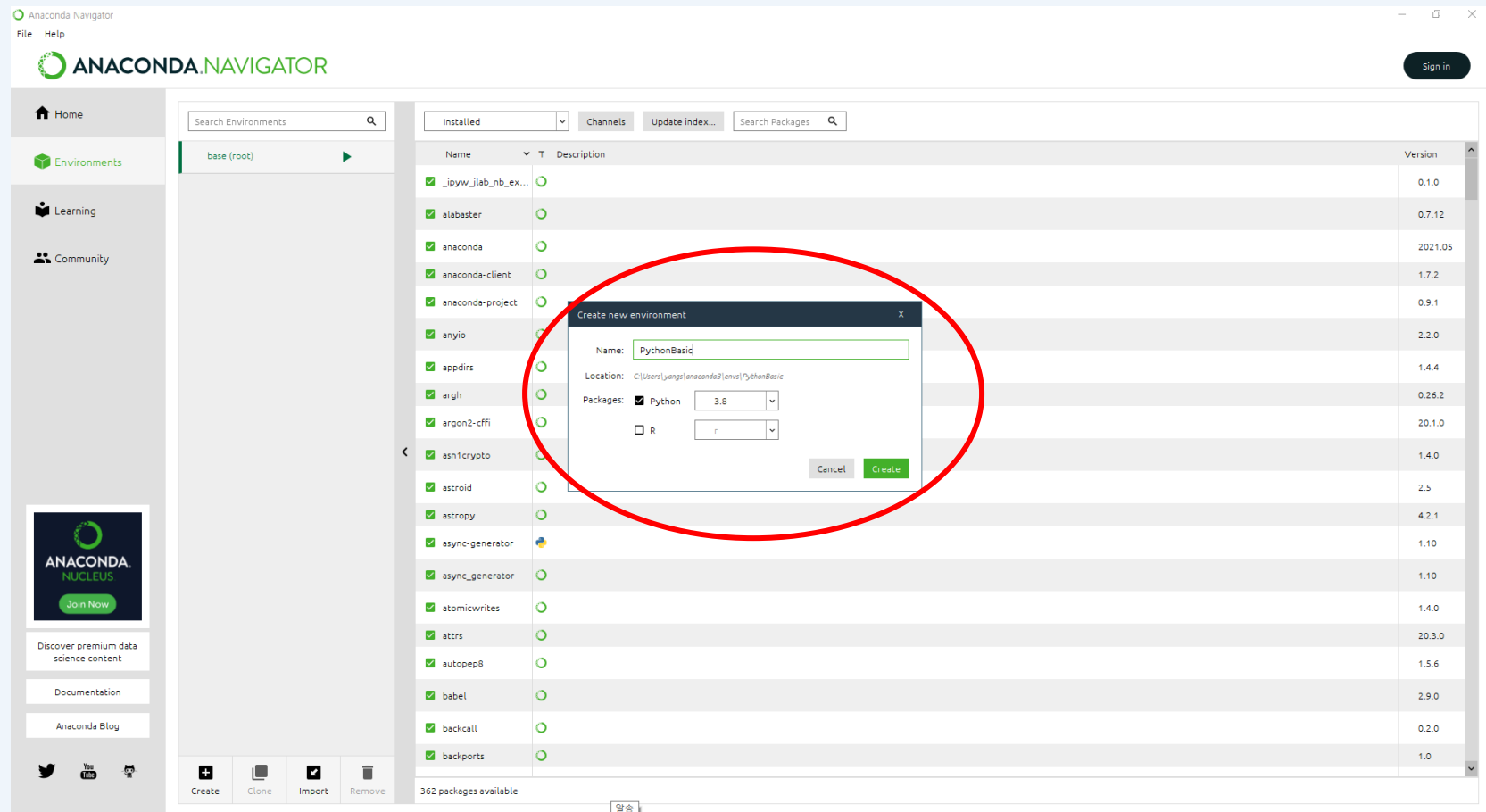
가상환경
추가버튼



2. 개발환경 설정

• Anaconda 기반 가상환경 만들기

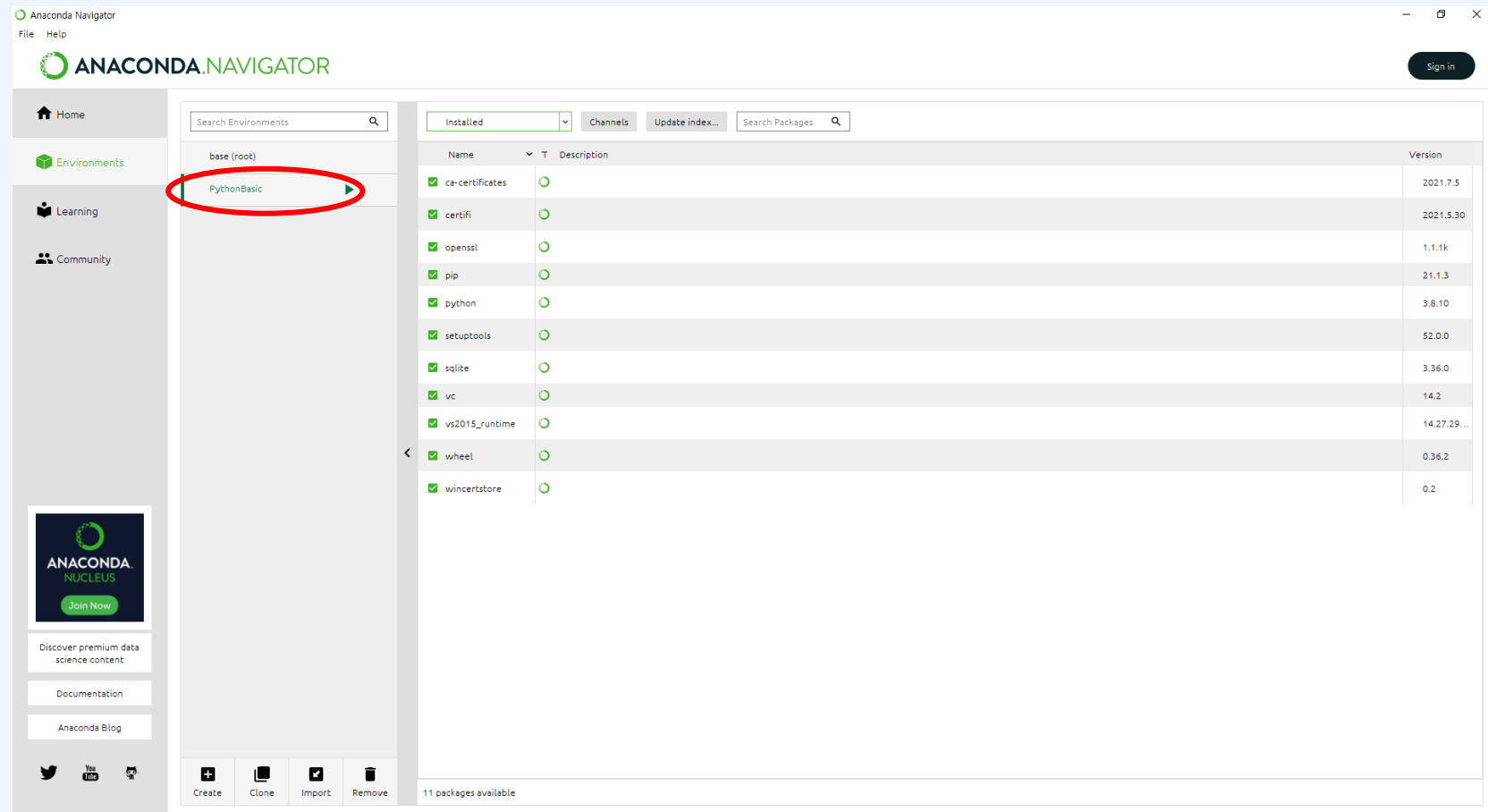
가상환경 이름 및
파이썬 버전 선택
후
"Create"



2. 개발환경 설정

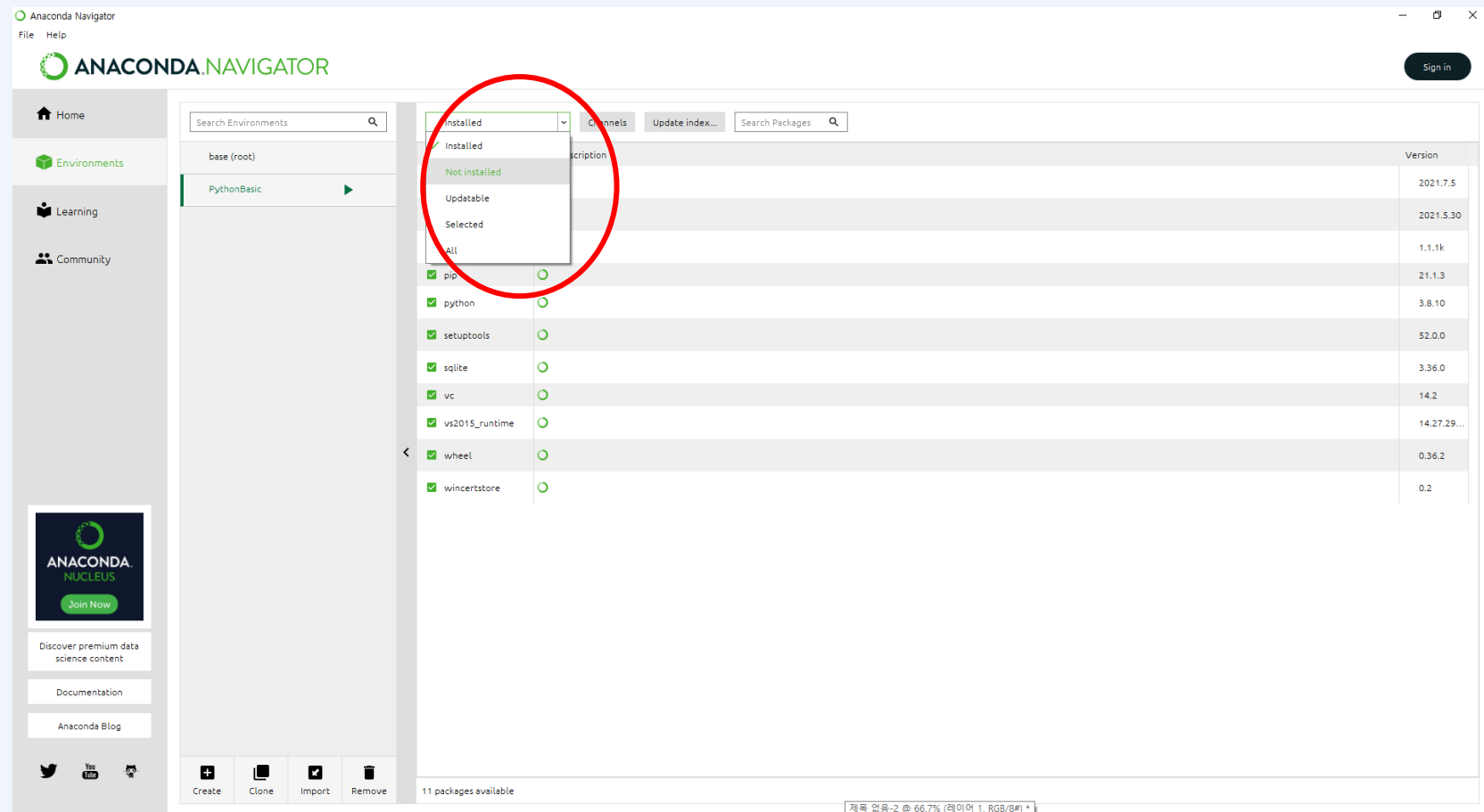
• Anaconda 기반 가상환경 만들기

생성완료



• 가상환경에 모듈 설치하기

Not Installed
또는
All 선택

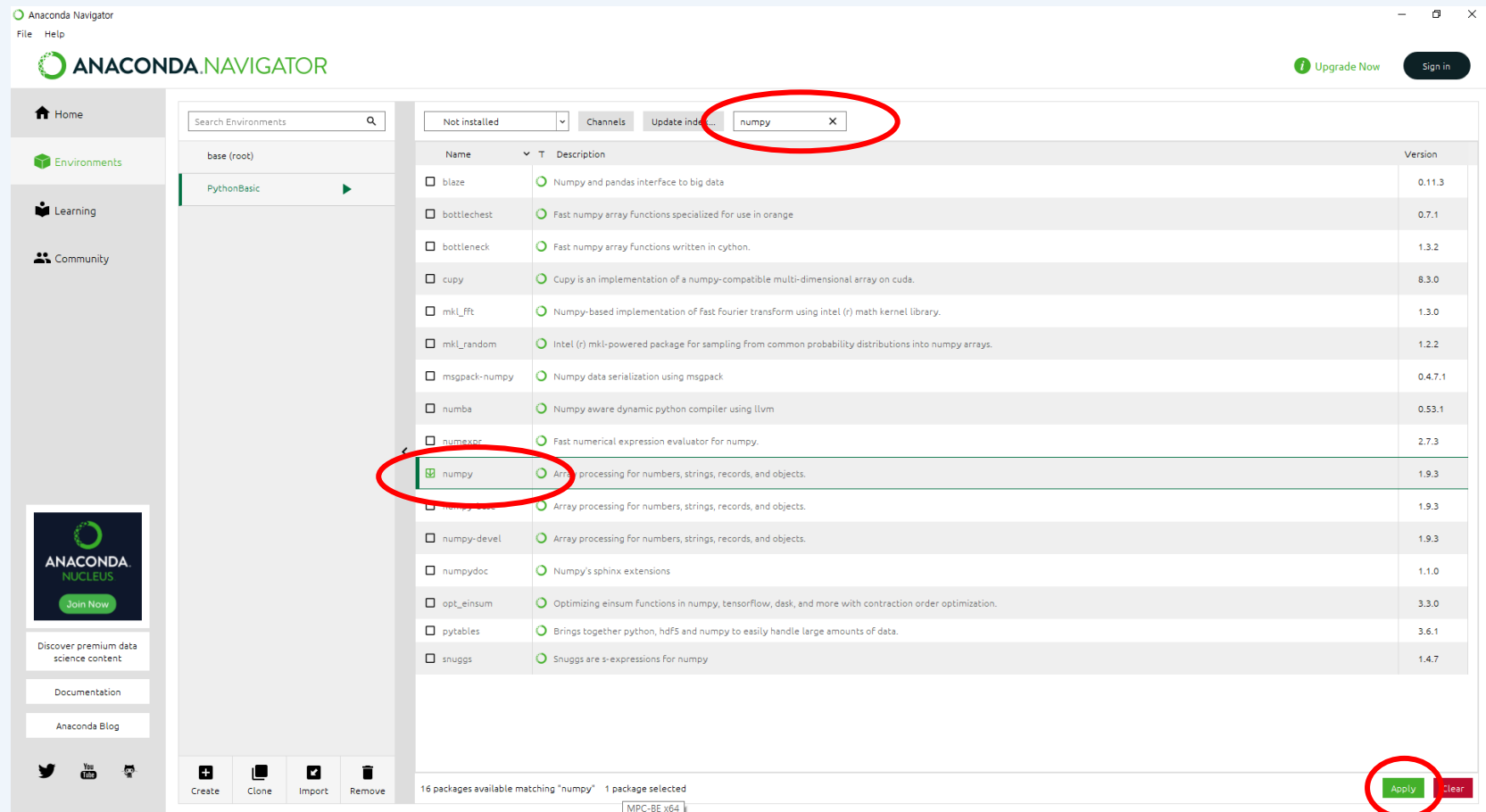


2. 개발환경 설정

• 가상환경에 모듈 설치하기

설치하려는 모듈
검색 후 선택,
"Apply" 버튼

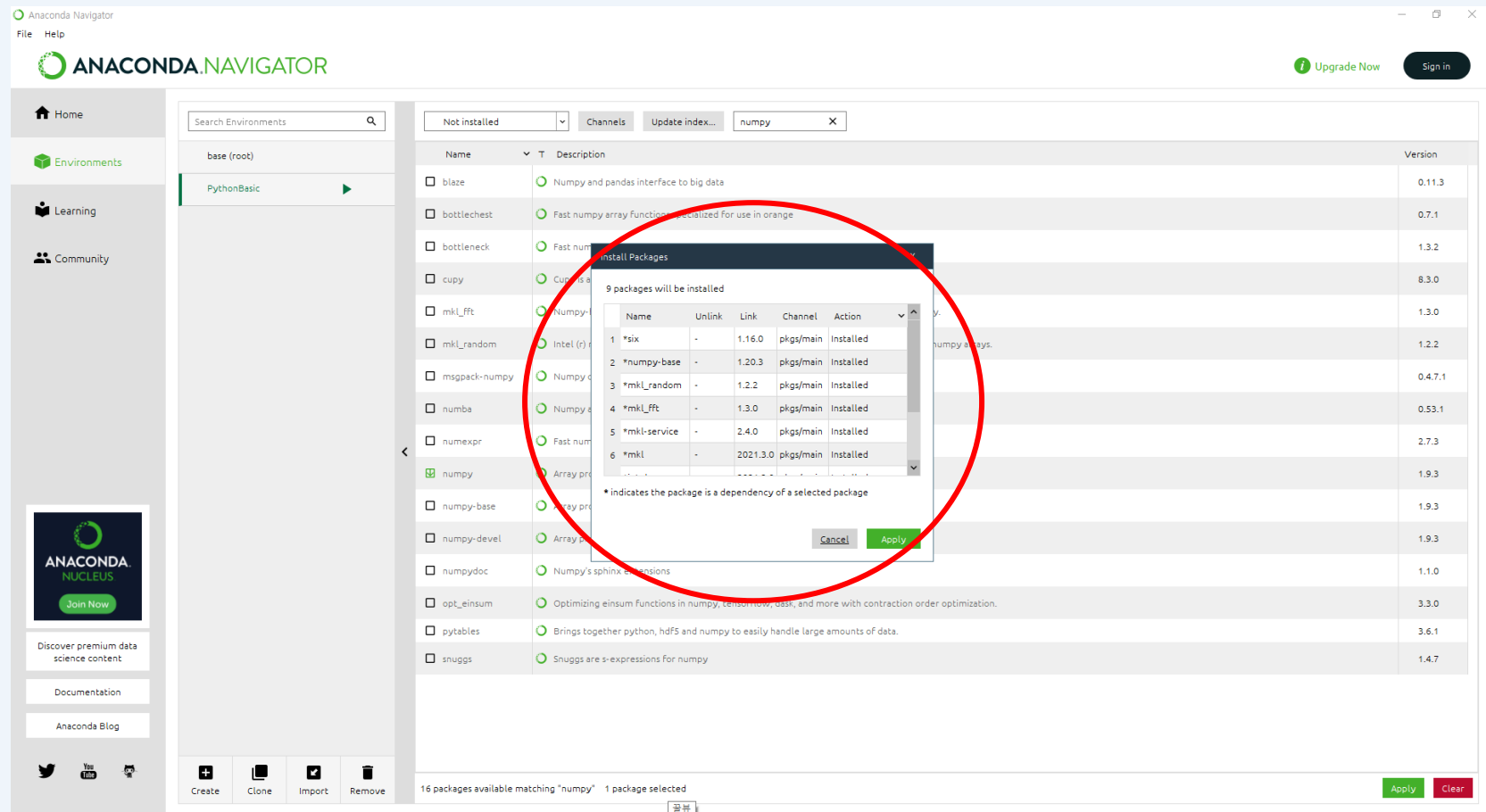
만약 검색결과에 없으면
"Update index" 버튼을
눌러서 항목 갱신



2. 개발환경 설정

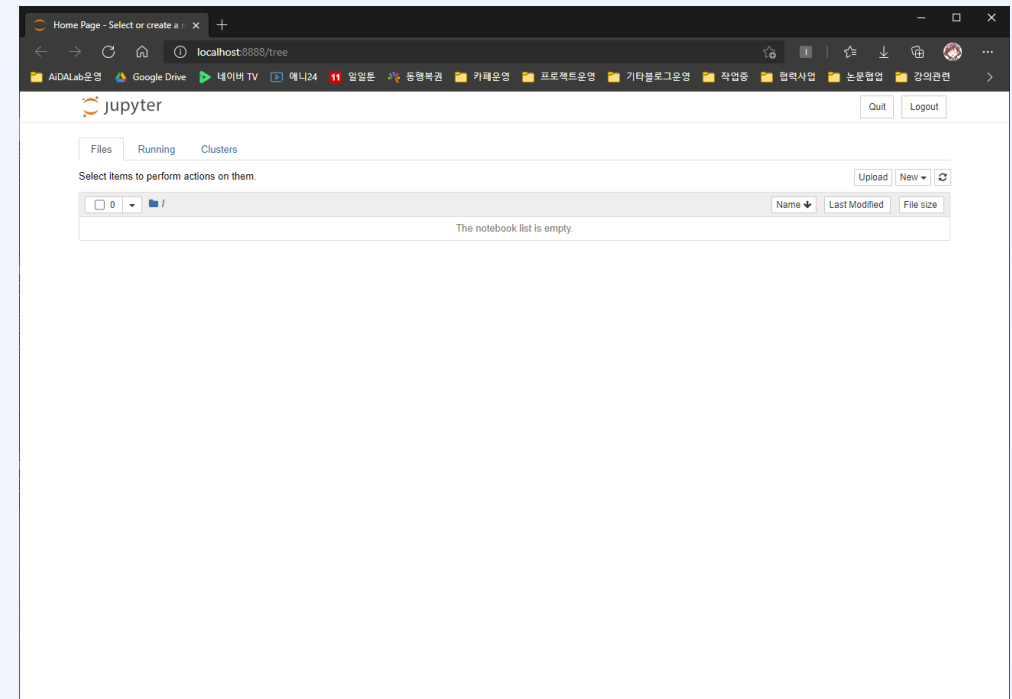
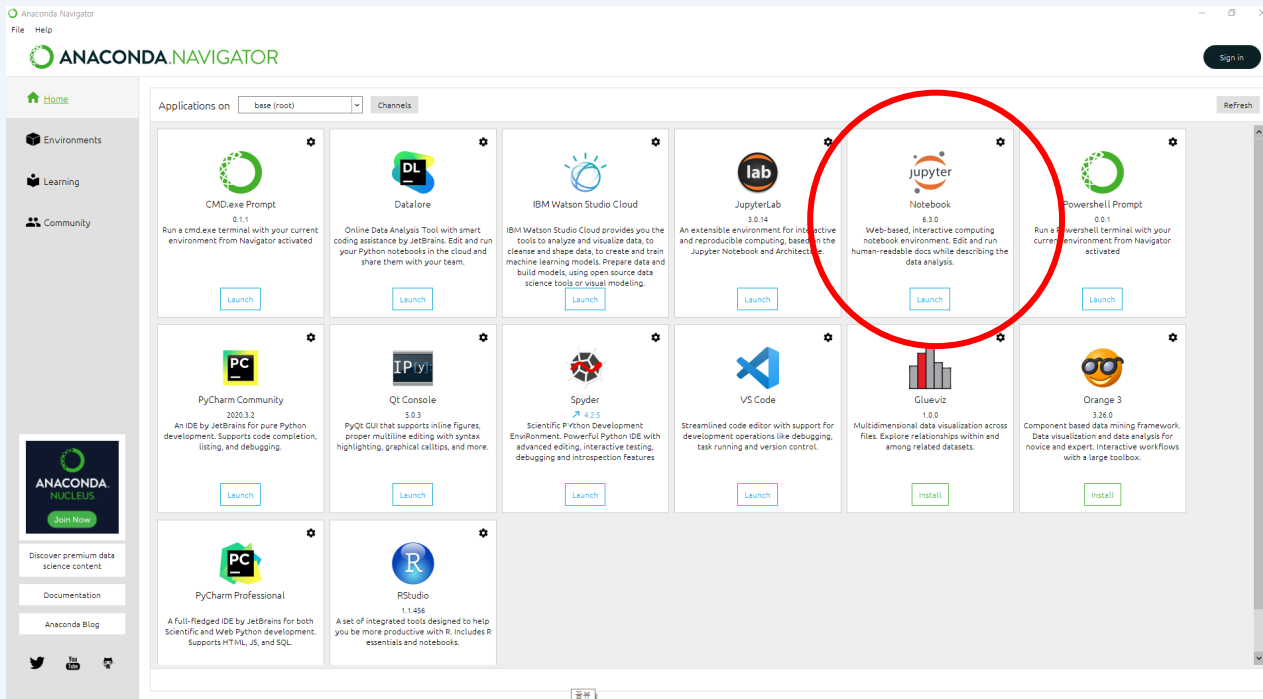
• 가상환경에 모듈 설치하기

설치하려는 모듈의
패키지 확인 후
"Apply" 버튼



3. Jupyter Notebook

- 프로그램 코드를 셀 단위로 수행할 수 있는 인터랙티브 개발 도구



2. 개발환경 설정

• Virtualenv 를 사용할 경우

[Ubuntu]

```
$ sudo apt install python3-virtualenv
$ cd workspace
$ virtualenv tf-gpu
$ cd tf-gpu
$ source bin/activate
(tf-gpu) $ pip install numpy
```

[Windows] 파이썬 다운로드, 설치 후

```
C:\> pip install virtualenv
C:\> cd workspace
C:\workspace> virtualenv tf-gpu
C:\workspace> cd tf-gpu\Scripts
C:\workspace\Scripts> activate
(tf-gpu) C:\workspace\Scripts> cd ..
(tf-gpu) C:\workspace\Scripts> pip install numpy
```

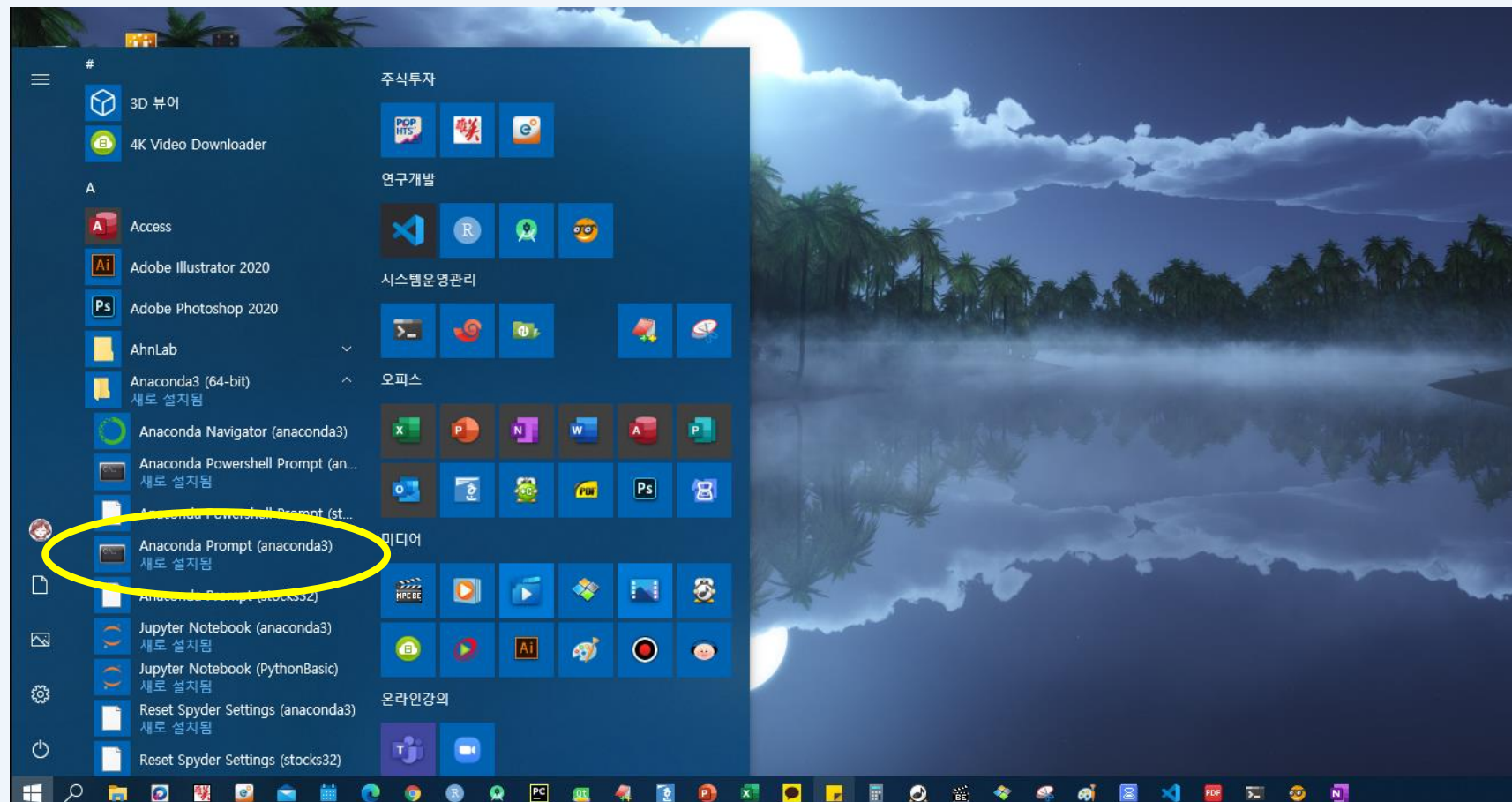
```
seokhwan@deep: ~/workspace
seokhwan@deep:~/ $ sudo apt install python3-virtualenv
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
libbsd0:i386 libdrm2:i386 libexpat1:i386 libfprint-2-tod1 libglapi-mesa:i386 libglvnd0:i386 libnvidia-common-440
libx11-6:i386 libx11-xcb1:i386 libxau6:i386 libxcb-dri2-0:i386 libxcb-dri3-0:i386 libxcb-glx0:i386
libxcb-present0:i386 libxcb-sync1:i386 libxcb1:i386 libxdamage1:i386 libxdmcp6:i386 libxext6:i386 libxf86vm1:i386
libxshmfence1:i386 libxxf86vm1:i386
'sudo apt autoremove'를 이용하여 제거하십시오.
다음 새 패키지를 설치할 것입니다:
python3-virtualenv
0개 업그레이드, 1개 새로 설치, 0개 제거 및 4개 업그레이드 안 함.
63.4 k바이트 아카이브를 받아야 합니다.
이 작업 후 362 k바이트의 디스크 공간을 더 사용하게 됩니다.
받기:1 http://mirror.kakao.com/ubuntu focal/universe amd64 python3-virtualenv all 20.0.17-1 [63.4 kB]
내려받기 63.4 k바이트, 소요시간 0초 (735 k바이트/초)
Selecting previously unselected package python3-virtualenv.
(데이터베이스 읽는중 ...현재 214439개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack .../python3-virtualenv_20.0.17-1_all.deb ...
Unpacking python3-virtualenv (20.0.17-1) ...
python3-virtualenv (20.0.17-1) 설정하는 중입니다 ...
Processing triggers for man-db (2.9.1-1) ...
seokhwan@deep:~/ $
```

```
seokhwan@deep: ~/workspace/tf-gpu
seokhwan@deep:~/workspace$ cd ~/workspace
seokhwan@deep:~/workspace$ virtualenv tf-gpu
created virtual environment CPython3.8.2.final.0-64 in 215ms
creator CPython3Posix(dest=/home/seokhwan/workspace/tf-gpu, clear=False, global=False)
seeder FromAppData(download=False, distro=latest, CacheControl=latest, requests=latest, colorama=latest, idna=late
st, six=latest, msgpack=latest, urllib3=latest, html5lib=latest, setuptools=latest, pip=latest, pyparsing=latest, re
trying=latest, distlib=latest, ipaddr=latest, wheel=latest, packaging=latest, progress=latest, contextlib2=latest, c
hardet=latest, pkg_resources=latest, webencodings=latest, appdirs=latest, pep517=latest, certifi=latest, pytoml=late
st, lockfile=latest, via=copy, app_data_dir=/home/seokhwan/.local/share/virtualenv/seed-app-data/v1.0.1.debian)
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
seokhwan@deep:~/workspace$ ls
tf-gpu
seokhwan@deep:~/workspace$ cd tf-gpu
seokhwan@deep:~/workspace/tf-gpu$ ls
bin lib pyvenv.cfg
seokhwan@deep:~/workspace/tf-gpu$
```

3. Jupyter Notebook

- Console 환경에서 가상환경 설치, Jupyter Notebook 설치

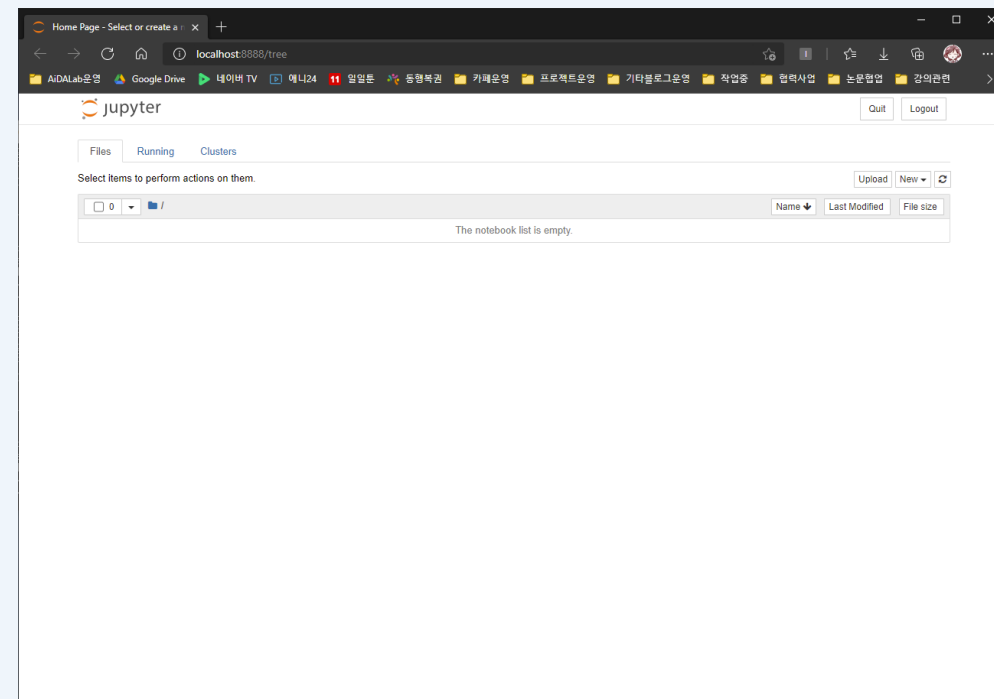
Anaconda Prompt
선택



3. Jupyter Notebook

• Console 환경에서 가상환경 설치

```
(base) PS C:\Users\Wyangs\> mkdir workspace
(base) PS C:\Users\Wyangs\> cd workspace
(base) PS C:\Users\Wyangs\workspace> conda create -n jupyter-env python=3.8
.....
(base) PS C:\Users\Wyangs\workspace> conda activate jupyter-env
(jupyter-env) PS C:\Users\Wyangs\workspace> conda install jupyter
.....
(jupyter-env) PS C:\Users\Wyangs\workspace> jupyter notebook
```



4. Colab

우리가 사용할 딥러닝용 PC를 직접 구매하려면... → **비싸다**

회사에서 사용했던 전용 딥러닝용 PC → 1,500만원



Google에서 제공하는 Colab(정식 명칭은 Colaboratory) 활용 권장

파이썬 / R 지원
Jupyter Notebook과 유사한 클라우드 기반 개발 환경 제공
브라우저 기반의 개발환경 제공 → 스마트폰에서도 사용 가능
GPU / TPU 지원

4. Colab

- Colab을 사용하려면

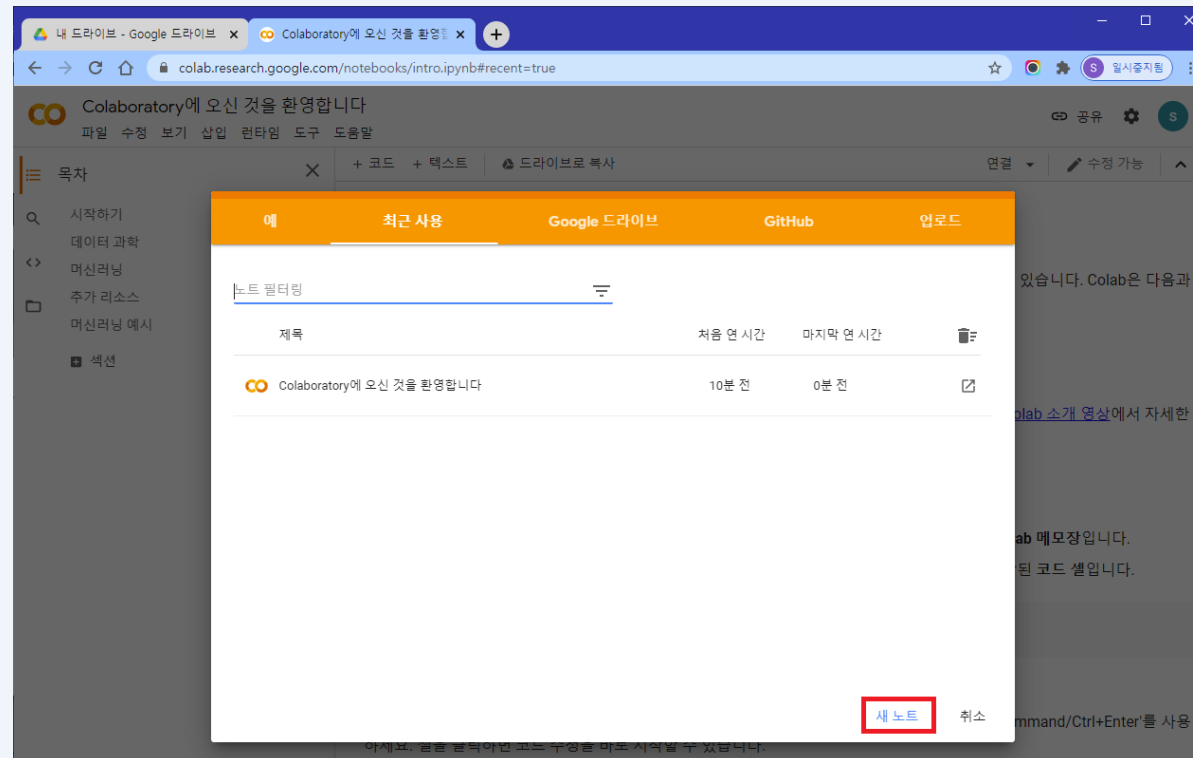
- Gmail 계정 생성(무료)

- Google Drive 확인

- Colab 서비스는 무료인 대신 12시간이 지나면 메모리에서 작업내용이 삭제됨
 - 작업 내용, 데이터 파일 등을 Google Drive와 연동하여 사용함으로써 해결 가능
 - 무료 용량: 최대 15GB

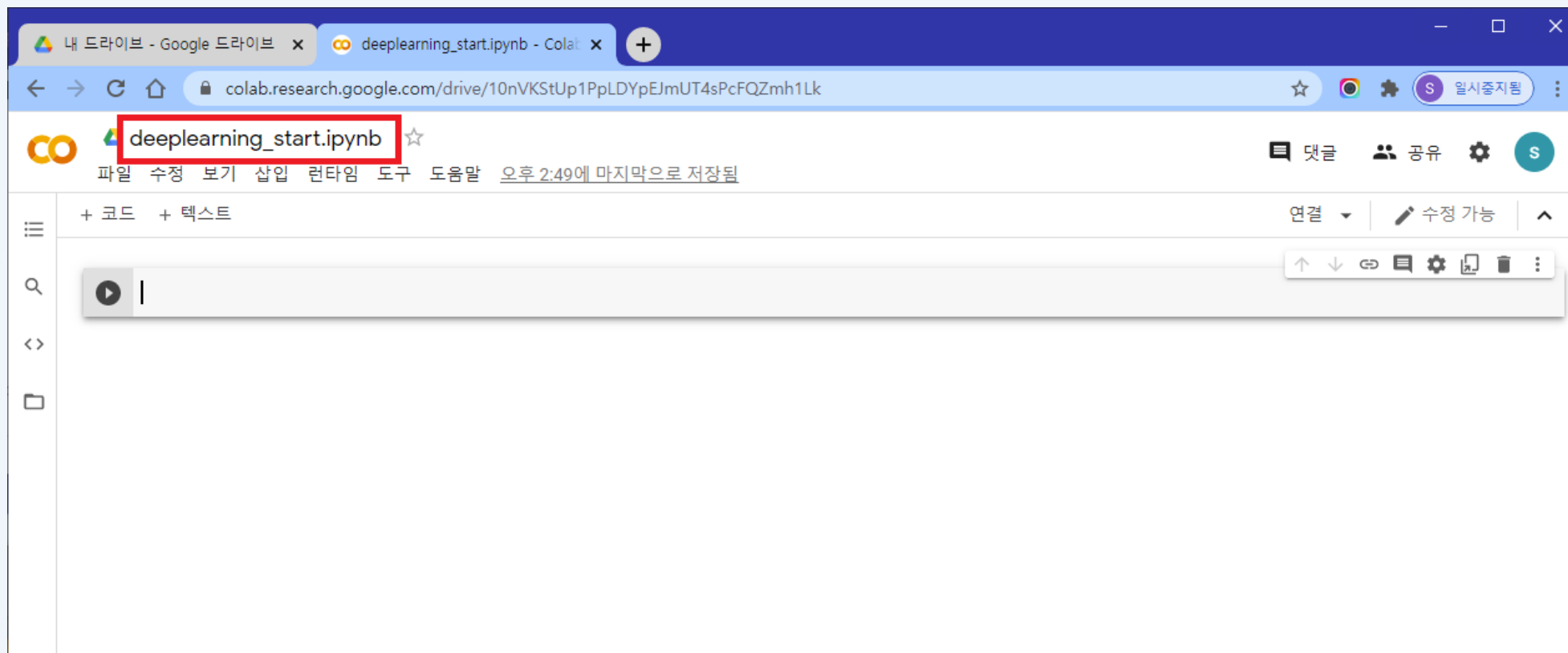
4. Colab

- Colab 환경 설정 (1)
 - <https://colab.research.google.com> 접속
 - 우측 하단 “새 노트” 선택하여
Note 생성



4. Colab

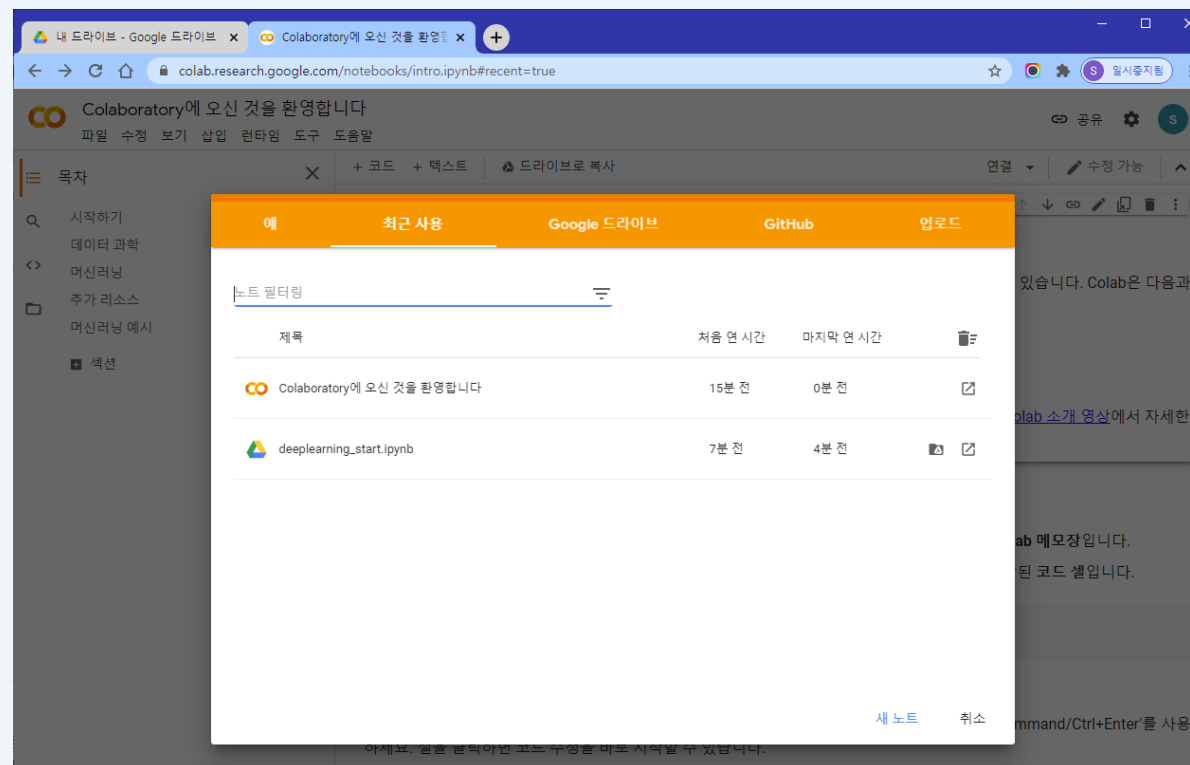
- Colab 환경 설정 (2)
 - 원하는 파일명 지정 후 작업 시작



4. Colab

- Colab 환경 설정 (3)

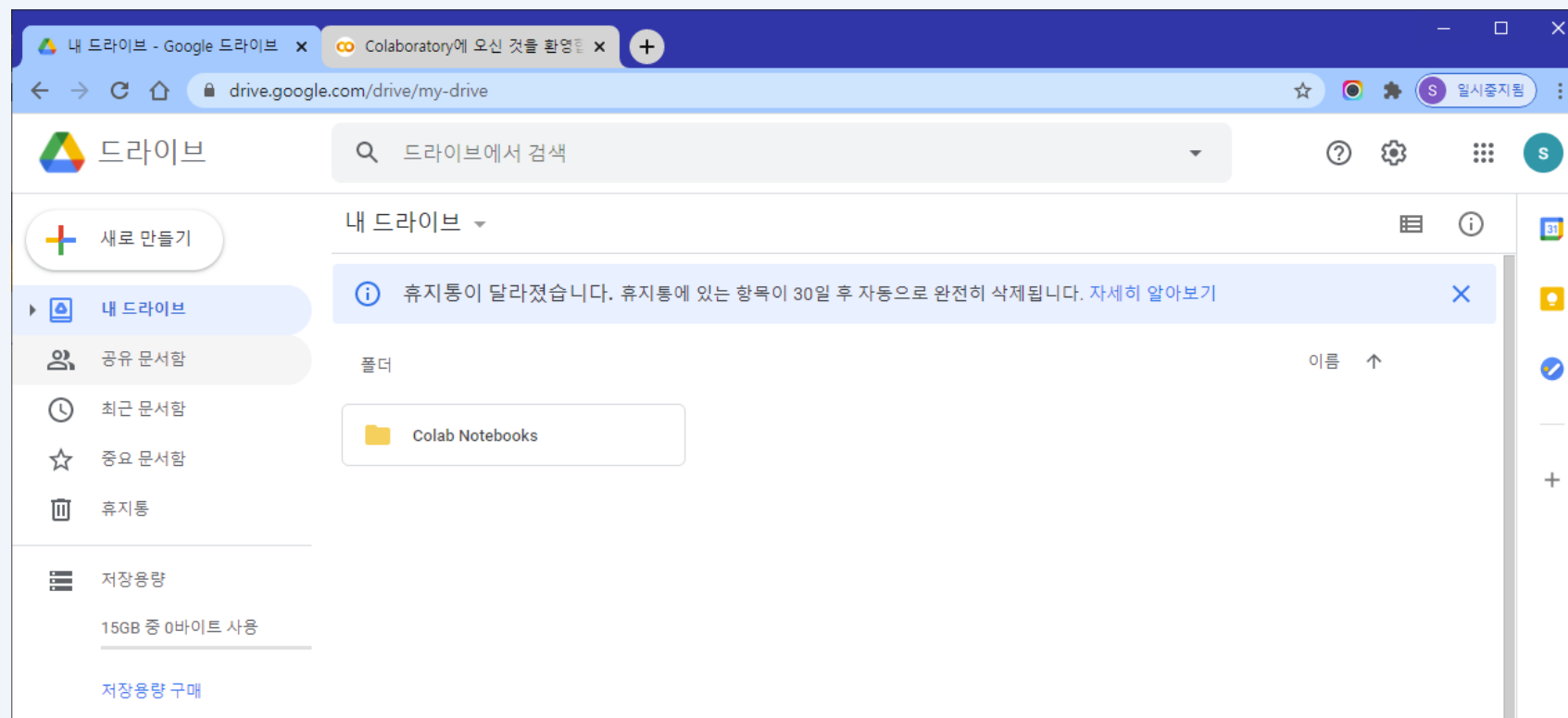
- 작업 내용은 자동 저장되며, 파일 메뉴에서 직접 저장도 가능
- 저장 후 Google Colab 링크로 돌아가서 작업 파일 저장 확인 가능



4. Colab

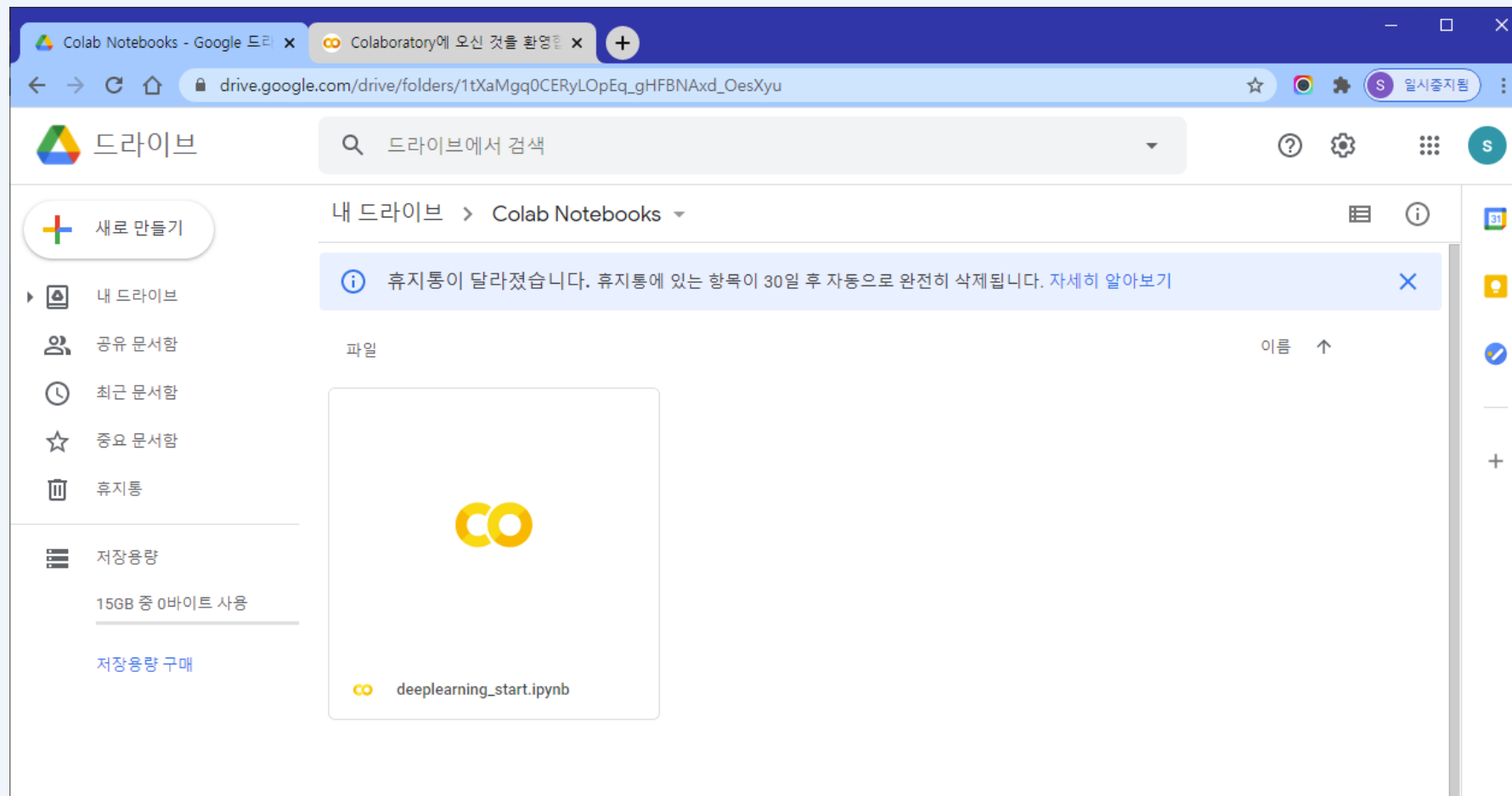
- Colab 환경 설정 (4)

- 한 번 설정하고 나면 Google Colab 링크 관계없이 자신의 Google Drive에서 접근 가능



4. Colab

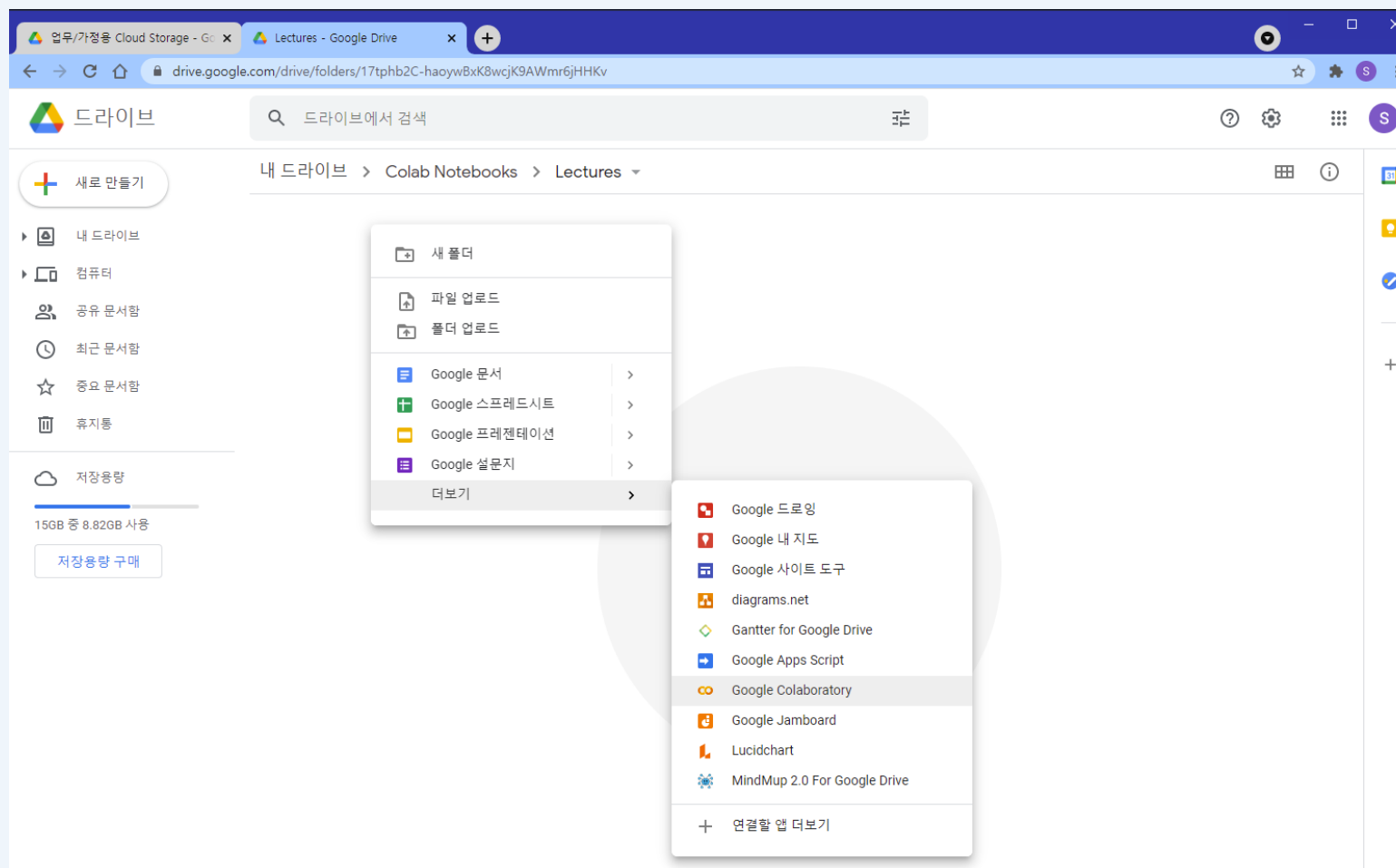
• Colab 환경 설정 (5)



4. Colab

• Colab 환경 설정 (6)

• 파일 새로 만들기



4. Colab

