

제 2편 파이썬 개발환경 구축

주요 학습 목표

- 개발환경 구축
- anaconda 가상환경

Chapter 02. Python 개발환경 구축하기

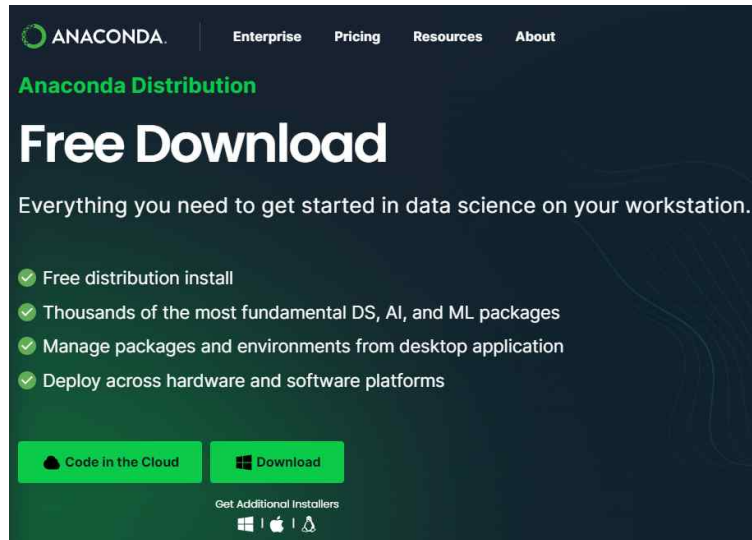
2.1 파이썬의 시작 Anaconda로 한다.

기본 python을 직접 설치하지 않고 아나콘다를 통해 설치하는 이유는 관리의 용이성과 독립 환경 구성 때문이다. 이에 대해서는 이미 많은 곳에서 다루고 있으므로 생략하기로 한다.

만약 기존의 설치된 파이썬이 다수 존재한다면 필요에 따라 그냥 두기도 하고 필요에 따라 지워도 무관하다.

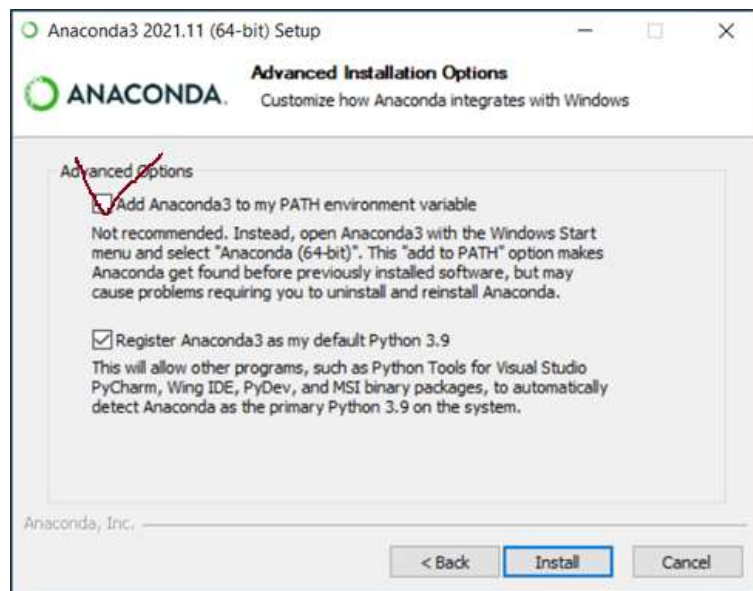
여기서는 설치 후 가상환경을 다루는 방법을 중심으로 설명한다. VScode에 연동하는 법은 뒤에서 다룬다. 다음의 공식 홈페이지에서 스크롤 내려서 64비트로 다운 및 설치를 진행한다. 아나콘다의 배포는 다음 URL을 따라가기 바란다.

www.anaconda.com/products/individual#windows



[그림 2-1] ANACONDA 다운로드

앞의 [그림 2-1] 다운로드 과정을 거친 뒤에는 설치 과정을 시작한다. 설치 과정 중에 가장 중요한 부분은 다음의 [그림 2-2]에서 보이는 PATH를 추가하는 옵션을 반드시 체크해야 한다는 것이다.



[그림 2-2] 콘다 설치시 PATH 입력 선택창

디폴트 옵션에서는 앞의 [그림 2-2]와 같이 path variable을 등록하지 말라고 한다¹⁷⁾. 이유는 보안의 이유이다. 체크인을 하고 Path를 등록하는 것이 반드시 필요하다. 그 밖의 설치 방법은 아나콘다 문서¹⁸⁾에 더 자세히 나와 있다. 용량이 부족한 경우 최소한만 설치하는 미니콘다라는 것이 있다.

2.2 설치된 Anaconda 확인

아나콘다 콘솔을 관리자 권한으로 켜 후 다음을 입력해 본다. 다음의 [그림1-2]는 설치가 완료된 후 확인을 위해서 윈도우에서 Anaconda Prompt (Anaconda3)를 오른쪽 버튼으로 눌러서 자세히를 선택하고 관리자 권한으로 실행하는 경로를 보여 주고 있다.



[그림 2-3] 콘다 콘솔을 여는 법

17) Anaconda Distribution 2022.05부터 모든 사용자 설치 중에 Anaconda를 PATH 환경 변수에 추가하는 옵션이 비활성화되었습니다. 이것은 보안 익스플로잇을 해결하기 위해 수행되었습니다. Just Me 설치 중에 Anaconda를 PATH 환경 변수에 추가할 수 있습니다.

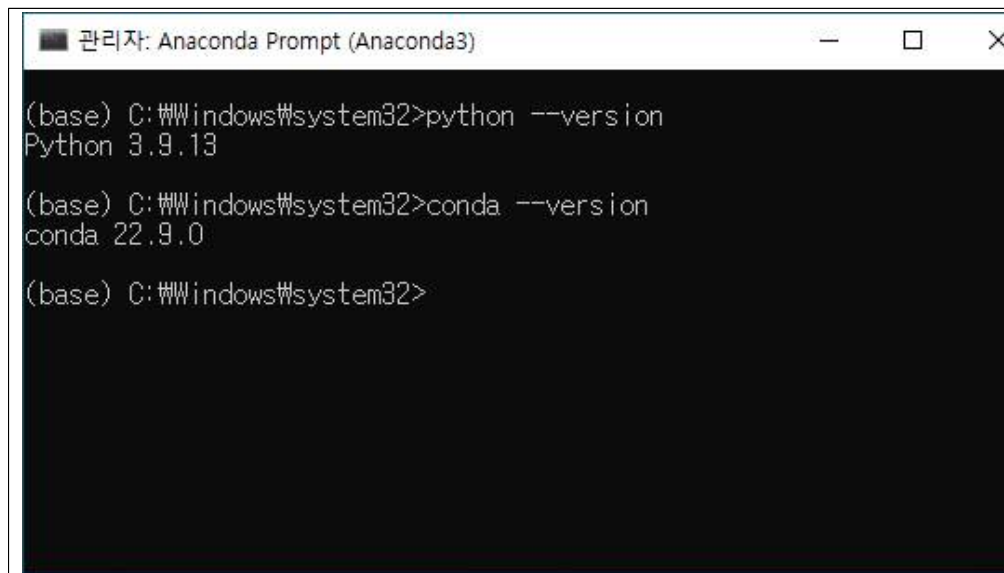
18) 아나콘다 문서 <https://docs.anaconda.com/anaconda/install/windows/>

다음의 [그림 2-4]는 콘다 콘솔에 입력으로, 아나콘다 Prompt를 관리자 권한으로 켜 후 다음을 입력해 본다.

```
python --version  
conda --version
```

[그림 2-4] 콘다 콘솔에 입력

앞의 [그림 2-4]를 입력하면 다음의 [그림 2-5]에서 보이는 것처럼 터미널 옆에 (base)라고 표시가 된다. 아나콘다 설치 시 기본으로 주어지는 virtual env 이름이 base인데, 이게 보이는 것. 만약 윈도우 터미널을 확인하여 그림과 같이 확인한다.

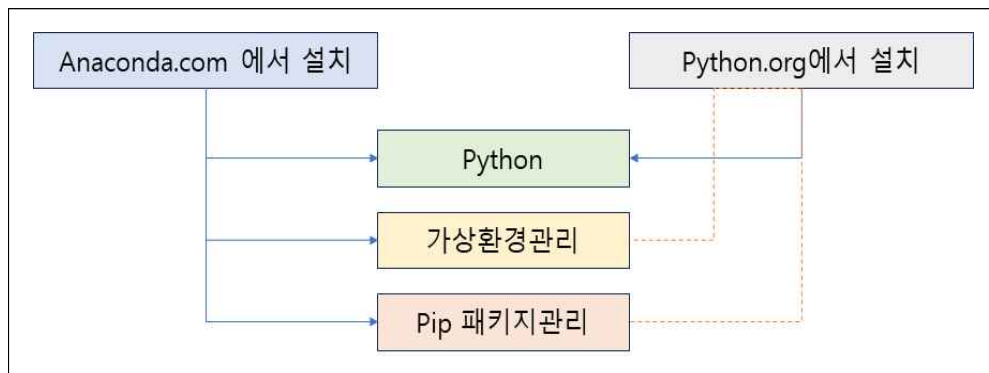


[그림 2-5] 콘다 콘솔 출력 상태

python과 anaconda 버전을 확인했다면 설치 시에 path 추가 옵션에 체크인을 하지 않았기 때문이다. 이런 경우 당황하지 않고 설치된 anaconda를 삭제하고 재설치할 것을 권한다. 물론 환경 변수 등을 조작해서 추가하는 방법도 있지만 그런 방법보다는 새로 설치하는 편이 더 빠르고 간결하다.

여기서 anaconda 패키지를 설치한다는 것은 무엇을 의미하는가 살펴보자. 사실상 anaconda 패키지를 설치하는 것은 원래 목적과 다르게 단순한 가상환경을 사용하기 위한 것이다. 아나콘다는 파이썬 보다 덩치가 큰 뱀의 종류라는 것은 알고 있을 것이다. 즉, 파이썬을 포함하고 가상환경을 지원하는 패키지라는 뜻이다. 그러나 실질적으로 파이썬 개발 과정에서 아나콘다의 패키지 관리를 사용하지는 않을 것이다. 이유는 pip 라고 하는 파이썬에 포함된 패키지 관리 툴이 따로 있기 때문이다. 즉 conda 라고 하는 아나콘다의 패키지 관리 툴을 사용하면 아나콘다 폴더에 미리 설치된 패키지를 우선적으로 사용하게 되어 네트워크 환경 등이 좋지 않은 환경에서의 개발 시 유용할 수는 있으나 사용하지 않는다. 보통은 pip 패키지 관리 툴을 사용함으로써 호환성을 극대화 할 수 있는 장점이 훨씬 크기 때문이다.

다음의 [그림 2-6]은 2가지 방법으로 설치가 가능한 파이썬 개발환경에 대한 모식도 이다. 이 책에서는 왼쪽의 Anaconda.com을 통해 설치하고 conda 패키지 관리 툴이 아닌 pip 패키지 관리 툴을 사용하는 개발환경을 구축 할 것이다.\



[그림 2-6] 개발환경 구축의 2가지 방향

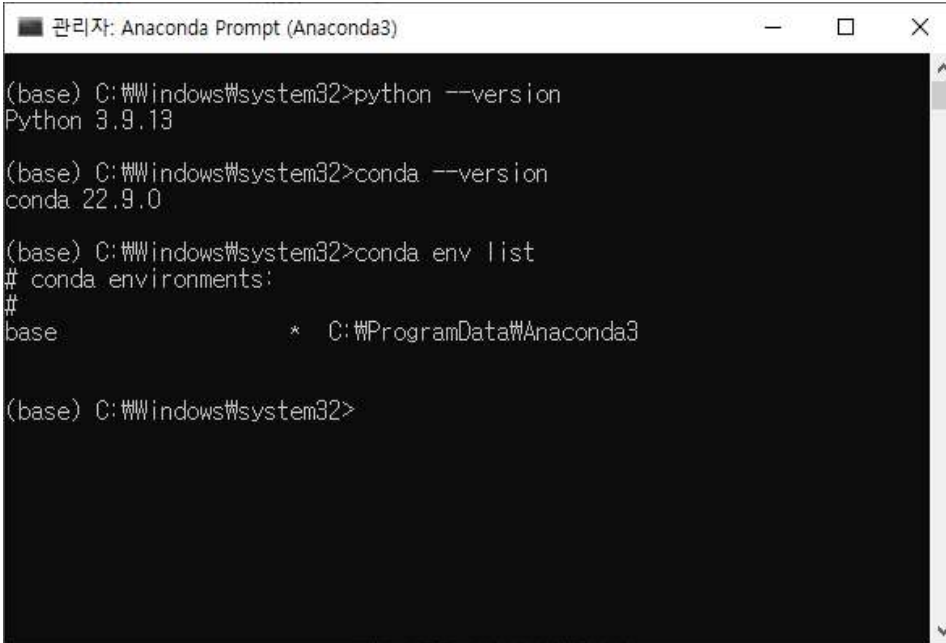
2.3 Base 가상 환경 이해하기

아나콘다 콘솔에서 다음을 입력한다. 기본으로 깔아주는 base라는 이름의 virtual env가 표시된다.

```
conda env list
```

[그림 2-7] 콘다 콘솔에 입력

앞의 [그림 2-7] 콘다의 가상환경의 목록을 보는 명령어이다. 다음의 [그림 2-8]은 콘다 가상환경 조회 화면이다.



```
(base) C:\Windows\system32>python --version
Python 3.9.13

(base) C:\Windows\system32>conda --version
conda 22.9.0

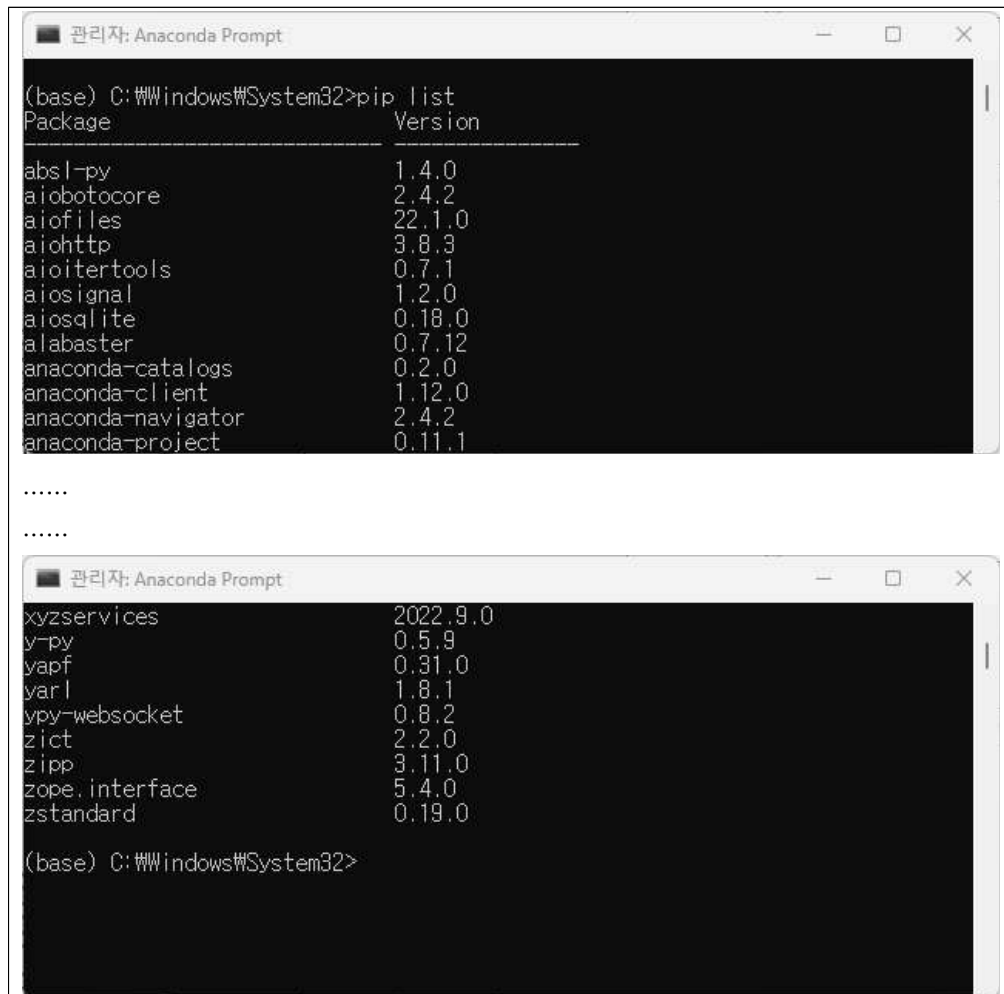
(base) C:\Windows\system32>conda env list
# conda environments:
#
base                  * C:\ProgramData\Anaconda3

(base) C:\Windows\system32>
```

[그림 2-8] 콘다 가상환경 조회

이렇게 설치된 base 이름을 가지고 있는 환경은 아나콘다의 모든 패키지를 포함한 base 환경이 된다. 이를 조회해 보면 다음과 같다.

다음의 [그림 2-9]는 pip list 명령어를 이용해서 base 환경의 설치된 패키지 목록을 보여주고 있다.



```
(base) C:\Windows\System32>pip list
Package                      Version
-----
absl-py                      1.4.0
aiobotocore                  2.4.2
aiofiles                     22.1.0
aiohttp                      3.8.3
aioitertools                 0.7.1
aiosignal                    1.2.0
aiosqlite                    0.18.0
alabaster                    0.7.12
anaconda-catalogs            0.2.0
anaconda-client              1.12.0
anaconda-navigator           2.4.2
anaconda-project             0.11.1
.....
.....
xyzservices                  2022.9.0
y-py                         0.5.9
yapf                         0.31.0
yarl                         1.8.1
ypy-websocket                0.8.2
zict                         2.2.0
zipp                         3.11.0
zope.interface               5.4.0
zstandard                    0.19.0

(base) C:\Windows\System32>
```

[그림 2-9] pip list 명령으로 설치된 패키지 목록 보기

그렇다면 내가 나만의 가상환경에서 수많은 패키지가 설치되지 않은 깨끗한 개발환경을 만드는 방법에 대해서 알아보자.

2.4 내가 사용할 가상 환경(virtual env)

프로그램 개발을 시작하게 되면 시스템의 안정성을 위해 기본으로 깔아주는 base라는 이름의 virtual env는 말고 독립된 가상환경을 구축하는 것이 좋다. 새로운 가상환경 생성을 위해 다음과 같이 입력했다.

```
conda create -n mine python=3.9
```

[그림 2-10] 콘다 콘솔에 입력

앞의 [그림 2-10]의 conda create은 새로운 가상환경을 만든단 얘기고 -n은 mine option이며, mine은 임의로 지은 이름, python 버전은 글 작성하는 지금 최신인 3.9를 입력했다. python 3.9.2와 함께 몇개 패키지가 설치된다. 기존에 base에 있던 다양한 패키지는 하나도 안 깔리므로 나중에 필요한 것 위주로 깔아줘야 한다. 바로 이점이 가상환경을 사용하는 이유이기도 하다.

다양한 애플리케이션을 개발하다 보면 상호 충돌 되는 상황이 발생할 수 있어서 가상환경이라는 테두리에 가두어 두고 독립적으로 개발을 할 수 있는 것이다.

다음 [그림 2-11]은 새로운 가상 환경 생성을 진행 중인 화면이다.

```
관리자: Anaconda Prompt (Anaconda3) - conda create -n mine python=3.9
(base) C:\Windows\system32>conda create -n mine python=3.9
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3\envs\mine

added / updated specs:
- python=3.9

The following NEW packages will be INSTALLED:

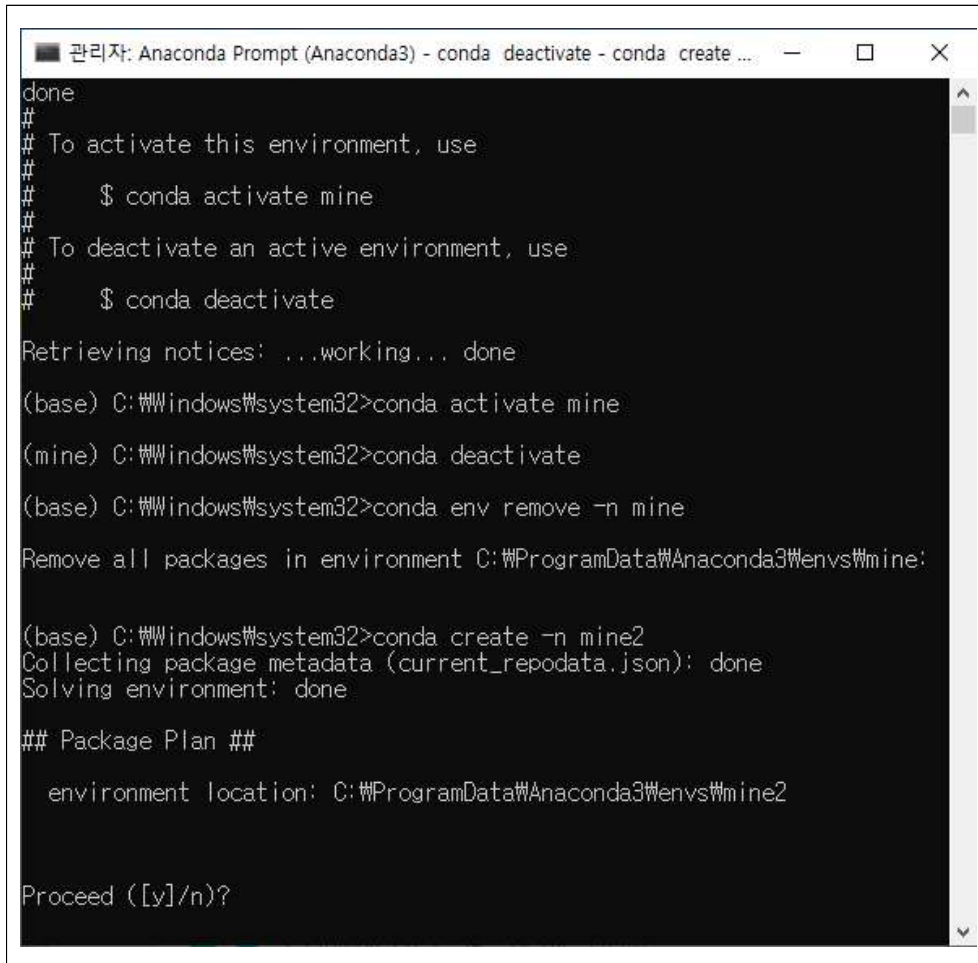
  ca-certificates pkgs/main/win-64::ca-certificates-2022.10.11-haa95532_0 None
  certifi         pkgs/main/win-64::certifi-2022.9.24-py39haa95532_0 None
  openssl        pkgs/main/win-64::openssl-1.1.1q-h2bbff1b_0 None
  pip            pkgs/main/win-64::pip-22.2.2-py39haa95532_0 None
  python         pkgs/main/win-64::python-3.9.13-h6244533_2 None
  setuptools     pkgs/main/win-64::setuptools-65.5.0-py39haa95532_0 None
  sqlite         pkgs/main/win-64::sqlite-3.39.3-h2bbff1b_0 None
  tzdata         pkgs/main/noarch::tzdata-2022f-h04d1e81_0 None
  vc             pkgs/main/win-64::vc-14.2-h21ff451_1 None
  vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2 None
  wheel          pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0 None
  wincertstore   pkgs/main/win-64::wincertstore-0.2-py39haa95532_2 None

Proceed ([y]/n)?
```

[그림 2-11] 콘다 가상 환경 생성

다음의 [그림 2-12]는 새로운 가상환경 mine을 생성시킨 후 conda activate mine 명령을 이용하여 생성된 가상환경을 동작(진입)시키는 과정과 동작 중인 가상환경을 conda deactivate 명령으로 정지시키는 과정

정지된 가상환경 mine을 삭제하는 과정 그리고 다시 mine2를 생성하는 과정을 보여 주고 있다. 시간을 갖고 차근 차근 따라해 보기 바란다.



```
done
##
## To activate this environment, use
##
##     $ conda activate mine
##
## To deactivate an active environment, use
##
##     $ conda deactivate
Retrieving notices: ...working... done
(base) C:\Windows\system32>conda activate mine
(mine) C:\Windows\system32>conda deactivate
(base) C:\Windows\system32>conda env remove -n mine
Remove all packages in environment C:\ProgramData\Anaconda3\envs\mine:

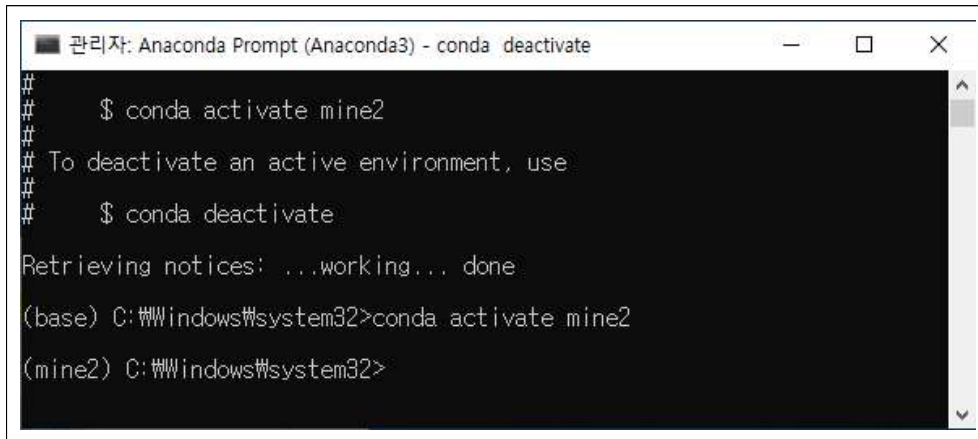
(base) C:\Windows\system32>conda create -n mine2
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3\envs\mine2

Proceed ([y]/n)?
```

[그림 2-12] 콘다 가상환경 가상, 활성화, 비활성, 삭제과정



```
관리자: Anaconda Prompt (Anaconda3) - conda deactivate
#
# $ conda activate mine2
#
# To deactivate an active environment, use
#
# $ conda deactivate
Retrieving notices: ...working... done
(base) C:\Windows\system32>conda activate mine2
(mine2) C:\Windows\system32>
```

[그림 2-13] mine2 가상환경 활성화

이렇게 해서 가상환경을 사용하기 위한 아나콘다의 설치 과정과 다루는 방법에 대해서 학습하였다. 이제부터 사용하게 되는 우리의 가상환경 이름은 mine2가 되었다.



```
관리자: Anaconda Prompt
# $ conda deactivate

(base) C:\Windows\System32>conda activate mine2
(mine2) C:\Windows\System32>python --version
Python 3.9.18

(mine2) C:\Windows\System32>pip list
Package      Version
-----
pip          23.3
setuptools   68.0.0
wheel        0.41.2

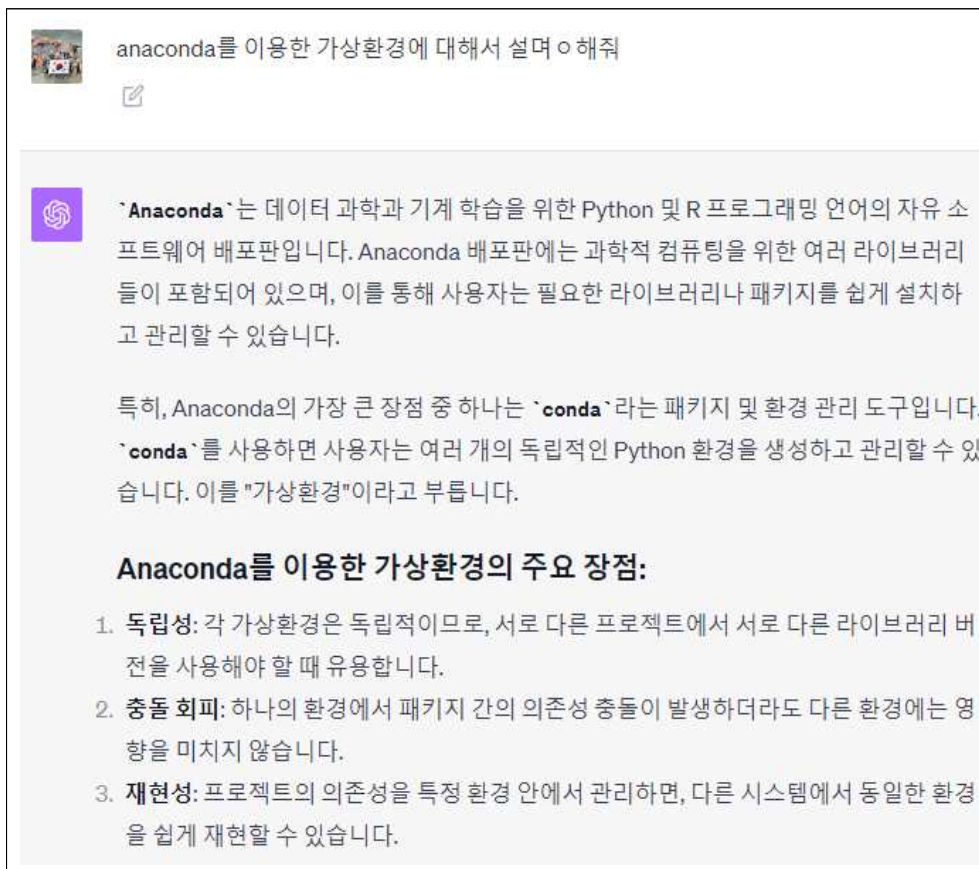
(mine2) C:\Windows\System32>
```

[그림 2-14] mine2 가상환경

앞의 [그림 2-14]를 보면 mine2 가상환경이 activated 되었으며 여기서 지정된 python의 version 정보를 알 수 있다. 또한, pip list 명령어를 이용하여 설치된 패키지를 볼 수 있다. 앞서 base 환경과 비교하면 꼭 필요한 패키지 관리 툴과 셋업 툴 정도만 설치되고 모든 환경이 깨끗한 상태인 것을 알 수 있다.

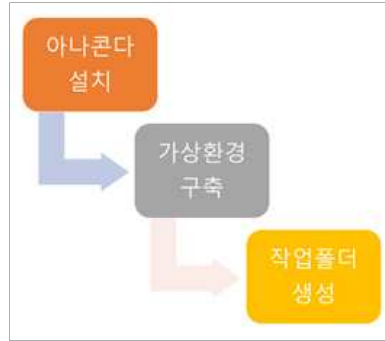
2.5 chatGPT에게 묻는 가상환경

다음의 [그림 2-15] chatGPT에게 묻는 가상환경은 지금까지 수업한 내용에 대해서 같은 방법으로 chatGPT를 이용해 묻고 답하는 과정을 통해 복습을 해보자 한다.



[그림 2-15] chatGPT에게 묻는 가상환경

다음의 [그림 2-16]은 아나콘다 설치 후 개략적인 프로그램 시작 과정을 나타내는 것으로 가상환경을 구축하고 본인의 작업 폴더를 생성하는 것을 시작으로 한다는 점을 강조하고 있다.



[그림 2-16] 프로그램 개발 작업의 시작

다음의 [그림 2-17]은 가상환경 관리에 대한 사용방법을 chatGPT를 통해 다시 한번 복습하는 내용이다. 가상 경은 매우 중요하고 개발에 있어 첫걸음이다.

가상환경 생성 및 사용 방법:

1. 환경 생성:


```
bash
conda create --name [환경이름] python=[버전]
```
2. 환경 활성화:


```
bash
conda activate [환경이름]
```
3. 환경 내 패키지 설치: 환경이 활성화된 상태에서 패키지 설치. 예를 들어, numpy를 설치하려면:


```
bash
conda install numpy
```
4. 환경 비활성화:


```
bash
conda deactivate
```
5. 환경 목록 확인:



```
bash
conda env list
```
6. 환경 삭제:



```
bash
conda remove --name [환경이름] --all
```

Regenerate

[그림 2-17] chatGPT에게 듣는 가상환경 관리

다음의 [그림 2-18]은 아나콘다를 설치하였지만 conda를 사용하지 않고 pip라는 기본 패키지 관리 툴을 사용하는 것에 대한 chatGPT의 의견이다. 여기서 파이썬의 기본 가상환경 관리 툴인 “virtualenv”에 대한 언급이 있으나 무시하기로 한다.


패키지 관리를 pip를 사용하면 어떤 장점 ◦ | 있어? conda에 비해서



``pip``은 Python에서 가장 널리 사용되는 패키지 관리 도구입니다. ``conda``와 비교했을 때 ``pip``의 주요 장점 및 특징은 다음과 같습니다:

1. **범용성:** ``pip``은 Python의 표준 패키지 관리 도구이므로 거의 모든 Python 환경에서 작동합니다. 반면 ``conda``는 Anaconda 및 Miniconda 배포판에 특화된 도구입니다.
2. **PyPI의 지원:** ``pip``는 Python Package Index (PyPI)와 직접 연동되어 있습니다. PyPI는 수많은 Python 패키지를 호스팅하는 대표적인 저장소로, ``pip``를 사용하면 PyPI에 있는 거의 모든 패키지를 쉽게 설치할 수 있습니다.
3. **간단한 설치 스크립트:** 패키지 제작자는 ``setup.py``를 사용하여 패키지를 ``pip``-호환 가능하게 만들 수 있습니다. 이를 통해 사용자 정의 패키지나 비공개 패키지도 쉽게 분배 및 설치할 수 있습니다.
4. **가볍고 빠르다:** ``pip``는 순수한 Python 패키지 관리 도구로, ``conda``보다 가볍습니다. 따라서 간단한 의존성 관리나 빠른 패키지 설치가 필요할 때 유용합니다.
5. **가상환경 지원:** ``pip``는 ``virtualenv``와 함께 사용되면 Python 가상환경을 만들고 관리할 수 있습니다.

하지만, ``pip``와 ``conda``에는 각각의 장점이 있기 때문에 상황에 따라 적절한 도구를 선택하는 것이 중요합니다. 예를 들어, 데이터 과학이나 기계 학습 관련 라이브러리(특히 복잡한 의존성을 가진 것들)를 설치할 때는 ``conda``가 더 우수할 수 있습니다. 반면, 순수한 Python 개발 환경이나 웹 개발 환경에서는 ``pip``가 더 적합할 수 있습니다.

[그림 2-18] chatGPT에게 듣는 패키지 관리