

1교시 - 파이썬 프로그래밍 입문

컴퓨터의 구조

- 컴퓨터는 크게 다음 세 부분으로 나뉘어져 있음
 - CPU - 중앙 처리 장치
 - 저장장치
 - 주저장장치 - Memory
 - 보조저장장치 - Disk
 - 입출력장치
 - 입력장치 - 키보드와 마우스
 - 출력장치 - 모니터와 프린터
- 컴퓨터를 사용하는 입장에서는 CPU, 저장장치, 입출력장치는 모두 일종의 '자원'(Resource)에 해당
 - 여러 프로그램이 동시에 동작하는 멀티태스킹 환경에서는 CPU, 저장장치, 입출력장치를 프로그램들이 나눠 쓰거나 돌려 씀
 - 프로그램을 만들 때는 항상 자원이 유한하다는 것을 염두에 두어야 함

운영체제 (Operating System)

- 자원을 관리
 - 효율적으로 나눠쓰거나 돌려쓸 수 있는 방법을 제공
 - 쉽게 쓸 수 있는 방법을 제공
- Windows, Linux, MacOS 등이 운영체제에 해당
- 운영체제가 자원을 관리하는 방식
 - 컴퓨터의 CPU, Memory, Disk, 입출력장치 등 개별 장치를 모니터링
 - 개별 프로그램은 운영체제를 거치지 않고 자원을 직접 사용할 수 없음 (일부 예외는 존재)
 - 각 장치와 자원을 요청하는 프로그램의 특성에 맞게 자원을 할당
 - CPU라면 - 각 프로그램 별로 CPU를 사용하는 스케줄 표를 만들고 이에 따라 CPU를 사용하게 함

- Memory라면 - 각 프로그램 별로 배타적으로 사용 가능한 Memory 공간을 배정
- Disk라면 - 각 프로그램이 요청에 따라 각 시점에 Disk 내의 파일을 독점적으로 사용할 수 있도록 관리
 - 운영 체제에 포함된 다양한 프로그램(Ex. 메모장, 그림판 등)은 사용자 편의를 위한 부가 기능
- 보안 관리 측면, 가상화 측면에서 OS의 역할이 점차 증대됨

프로그램 (Application)

- 실행파일(.exe 확장자)은 '명령어의 집합'
- 프로그램을 실행하고 종료할 때 벌어지는 일
 1. 사용자가 프로그램을 실행
 2. 운영체제가 실행 요청을 접수
 3. (OS)프로세스를 생성
 - 프로세스는 어떤 프로그램이 실행되는 환경
 - 프로그램이 필요로 하는 자원을 포함
 4. (OS)프로그램을 메모리로 로드
 - .exe 파일에 포함된 명령어
 - 실행에 필요한 ('의존성이 있는') 파일도 메모리에 로드
 5. (CPU)명령어를 순서대로 실행
 - 여기서 순서는 '선형적 순서'를 의미하지 않음
 - 실행하는 과정에서 프로세스에 할당된 자원을 사용
 6. 프로그램이 OS에 종료를 요청
 - 종료의 상태에 대한 정보가 포함 (정상 종료, 비정상 종료)
 7. (OS)프로세스에 할당된 자원을 해제
 8. (OS)프로세스를 종료
- 프로그램의 구조
 - 바이너리 (0과 1)

프로그램 언어의 수준

- 기계수준 언어 - 실행 가능한 바이너리 (executable binary, executable)
 - 0과 1, 이진수로 표현
 - machine code(컴퓨터가 직접 처리할 수 있는 형태의 명령)과 메타데이터, 데이터 파트, 기타 실행에 필요한 정보 등이 포함된 파일
 - machine code는 같은 종류의 CPU에서는 동일하지만, 메타데이터, 데이터 파트, 기타 실행에 필요한 정보 등은 OS별로 다르므로 윈도우, 리눅스, MacOS의 프로그램이 서로 호환되지 않음
- 저수준 언어 - 어셈블리
 - 사람이 읽을 수 있는 형태로 표현한 기계어
 - CPU의 명령과 1:1로 매치 - CPU가 100개의 명령을 처리한다면 어셈블리 코드 100줄을 처리하는 것 (예외는 존재함)
- 고수준 언어 - 일반 프로그래밍 언어로 표현된 코드
 - 사람이 프로그램을 효율적으로 만들 수 있도록 형식이 정해진 언어
 - CPU 명령과 1:1로 매치되지 않음
 - 대부분의 경우 한 줄의 고수준 언어 프로그램 코드가 여러 줄의 어셈블리 언어와 매칭

```
# 고수준 언어 - python
```

```
2 + 3
```

```
# 저수준 언어 - 어셈블리 (x86)
```

```
mov eax, 2
```

```
add eax, 3
```

```
# 기계수준 언어 - 기계어 (x86)
```

```
10111000 00000010
```

```
00000101 00000011
```

- 예전에는 저수준 언어로 프로그램을 개발하는게 일반적이었으나, 꽤 오랜기간 동안 고수준 언어를 사용해 프로그램을 개발
 - 1950년대 발표된 고수준 언어 - FORTRAN(1957), LISP, COBOL, ALGOL
 - 1960년대 발표된 고수준 언어 - BASIC

- 1970년대 발표된 고수준 언어 - C (1972), Pascal(1970), Smalltalk, Prolog
- 1980년대 발표된 고수준 언어 - C++(1985), Objective-C, Perl
- 1990년대 이후 발표된 고수준 언어 - Python(1991), Java(1995), JavaScript(1995)

컴파일러와 인터프리터

- 고수준 언어를 OS가 이해할 수 있는 형태의 프로그램으로 바꾸는 중간 역할자가 필요
 - 프로그램 코드를 실행 파일로 만드는 '통역자'의 역할이 컴파일러
 - 프로그램 코드를 한 줄 한 줄 CPU가 실행할 수 있게 전달하는 '번역자'의 역할이 인터프리터
 - 다만, 경우에 따라 인터프리터 언어도 실행 파일을 만들 수 있고, 컴파일러 언어도 인터프리터 언어 형태로 실행할 수도 있음. 하지만 프로그래밍 언어의 본질적인 제작 철학에 따라 뚜렷하게 이 둘이 구분되는 것이 일반적
- 컴파일러
 - FORTRAN, COOL, C, C++, Pascal 프로그래밍 언어가 속함
 - 장점
 - 미리 기계어로 변환해 두므로 실행 속도가 빠름
 - 전체 프로그램을 읽은 후 기계어로 변환하므로 전체 코드를 분석해 최적 실행 구조로 재구조화
 - OS와 의존성 문제만 충족하면 프로그래밍 언어 없이 독립적으로 실행 가능
 - 소스 코드 노출 없이 프로그램 배포 가능
 - 단점
 - 컴파일 시간이 필요
 - 디버깅이 어려움
 - 다른 플랫폼으로 이식이 복잡하거나 어려움
- 인터프리터
 - LISP, BASIC, **Python**, JavaScript 프로그래밍 언어가 속함
 - 장점
 - 프로그램 작성 후 즉시 실행 가능 - 빠른 프로토타이핑, 개발 중 디버깅에 유리

- 디버깅 용이
 - 이식성 - 인터프리터가 설치된 컴퓨터라면 어떤 컴퓨터에서도 실행 가능
- 단점
 - 느린 실행 속도
 - 최적화 부족
 - 실행 환경 의존
- 혼합형 언어
 - Java, C# - 소스 코드를 바이트코드 또는 중간 언어로 컴파일한 후, 이를 JVM 또는 CLR이 실행 (인터프리터 또는 JIT 컴파일 방식)
 - 최근 많은 언어들이 성능 향상을 위해서 JIT 컴파일 방식을 채택하는 추세
 - JIT(Just-in-time)는 컴파일러와 인터프리터의 장점을 결합
 - 실행 시점에 컴파일 (코드의 일부 또는 전체) - 초기 실행 속도는 느리지만, 반복 또는 중요한 부분은 빠르게 실행
 - 실행 환경에 따라 최적의 프로그램 구조로 실행 구조를 작성

파이썬

- 네덜란드 프로그래머 귀도 반 로섬이 처음 개발해 1991년 최초 발표
- 인터프리터 언어
- 윈도우, 리눅스, 맥오에스 등의 운영체제와 Intel, AMD, ARM 등의 CPU 환경을 포함한 대부분의 CPU, 대부분의 OS에서 동작

다른 플랫폼을 위한 Python 다운로드

파이썬은 아래에 알파벳 순으로 나열된 여러 전문 및/또는 오래된 플랫폼으로 이식되었습니다. 이러한 이식은 종종 최신 파이썬 릴리스보다 훨씬 뒤떨어져 있습니다.

AIX를 위한 파이썬

Python3 및 Python2용 AIX 바이너리 패키지는 AIXTOOLS에서 무료로 제공됩니다. 패키지는 install 형식 (RPM 형식 아님)으로 제공됩니다. openssl.base의 최신 버전 외에는 사전 요구 사항이 없습니다.

자세한 내용은 AIXTOOLS 위키를 참조하세요. Python 3의 다운로드 페이지는 <http://www.aixtools.net/index.php/python3> 에 있고 Python 2의 다운로드 페이지는 <http://www.aixtools.net/index.php/python2> 에 있습니다.

Python의 AIXTOOLS 패키징 설치와 관련된 문제가 있으면 <http://forums.rootvg.net/aixtools> 에 문제를 게시하세요.

IBM i용 Python(이전 AS/400, iSeries)

Python 2와 Python 3은 모두 IBM에서 RPM 형태로 제공됩니다. yum 패키지 관리자 또는 IBM i Access Client Solutions 제품으로 설치할 수 있습니다. IBM i용 RPM 기반 오픈 소스 패키지를 시작하려면 <http://ibm.biz/ibmi-rpms> 를 방문하세요. 이러한 RPM 패키지에는 활성(확장되지 않음) 지원의 IBM i 버전이 필요합니다.

iOS 및 iPadOS용 Python

Pythonista는 iPad나 iPhone에서 타사 라이브러리와 시스템 통합을 포함하여 Python 스크립트를 작성하기 위한 완벽한 개발 환경입니다.

Pyto는 또한 iPad나 iPhone에서 많은 타사 라이브러리와 시스템 통합을 포함하여 Python 3을 실행하기 위한 완전한 개발 환경을 제공합니다.

z/OS용 Python

Rocket Software는 z/OS용 Python 포트를 제공합니다.

Python for z/OS는 IBM에서 라이선스 비용 없이 제공됩니다. Early Programs Web Tool의 PAX 형식 또는 Shopz의 SMP/E 형식으로 제공됩니다. 선택 사항인 무료 구독 및 지원(S&S)은 Shopz 주문 과정에서 제공됩니다. 자세한 내용은 [IBM Open Enterprise SDK for Python 제품 페이지](#) 를 방문하세요.

RISC OS용 Python

Python은 RISC OS에서 사용할 수 있으며, PackMan 패키지 관리자를 사용하여 얻을 수 있습니다.

Solaris용 Python

[ActivePython](#) (과학적 컴퓨팅 모듈을 포함한 상용 및 커뮤니티 버전, 오픈 소스 아님)을 구매하거나, C 컴파일러가 있는 경우 [소스에서 빌드할 수 있습니다](#).

UNIX Packages에는 다양한 Solaris 버전을 위한 다양한 Python 버전이 있습니다. 이들은 표준 Sun pkgadd를 사용합니다.

UEFI 환경을 위한 Python

UEFI(Unified Extensible Firmware Interface) 셸 환경을 위한 표준 CPython 버전 3.6.8 포트는 Tianocore 오픈 소스 프로젝트를 통해 제공됩니다. 이는 UEFI 환경에서 표준 Python 스크립팅 기능을 제공하여 UEFI 기반 펌웨어 및 플랫폼 개발자 커뮤니티가 플랫폼, 펌웨어 검증, 디버그 등에 사용할 수 있도록 돕습니다. UEFI용 Python 소스 코드와 빌드 지침은 [여기에서](#) 제공됩니다.

현재 x86 및 x64 비트 플랫폼에 대한 VS2019 및 GCC5 도구 체인을 사용하여 빌드 지원이 활성화됩니다.

HP-UX를 위한 파이썬

[ActivePython](#) (과학 컴퓨팅 모듈을 포함한 상용 및 커뮤니티 버전, 오픈 소스 아님)을 구매할 수 있습니다.

Linux용 대체 Python 패키지

[ActivePython](#) (과학 컴퓨팅 모듈을 포함한 상용 및 커뮤니티 버전, 오픈 소스 아님)을 구매할 수 있습니다.

- 2008년 파이썬 3.0 발표
 - 그 이전의 파이썬 2.0은 파이썬이 아니라고 간주해도 무방
 - 가장 쉬운 구분 법 - print 사용법

```
# Python 2.X 스타일
print "Hello, World!"

# Python 3.X 스타일
print("Hello, World!")
```

- 파이썬 버전에 대해서
 - 숫자 3개로 구성
 - 첫 번째 숫자 (Python 3.12.6에서 3에 해당) - 메이저 버전
 - 주 버전이 변경되면 호환성에 중요한 변화가 발생했음을 의미
 - 파이썬 2와 파이썬 3은 '거의' 다른 언어이고 '하위 호환'이 보장되지 않음
 - 파이썬 2.X 환경에서 만들어진 프로그램이 파이썬 3.X 인터프리터에서 동작 하지 않음
 - 두 번째 숫자 (Python 3.12.6에서 12에 해당) - 마이너 버전
 - 새로운 기능이 추가되었을 때 바뀌는 숫자
 - 일부 예외를 제외하고 하위 호환성을 유지
 - 파이썬 3.9.X 환경에서 만들어진 프로그램이 파이썬 3.12.6 인터프리터에서는 정상 동작(할 가능성이 매우 높음)
 - 파이썬 3.12.6 환경에서 만들어진 프로그램이 파이썬 3.9.X 환경에서 정상 동작하지 않을 수 있음 (신규 기능을 사용하는 경우)
 - 세 번째 숫자 (Python 3.12.6에서 6에 해당) - 패치 버전
 - 기능상의 차이 없이 버그 수정이나 보안 패치 등의 소소한 변경 사항이 있을 때 바뀌는 숫자
 - 동일한 호환성을 보장
 - 파이썬 3.12.5 환경에서 만들어진 프로그램을 파이썬 3.12.6에서 실행할 수 있으며 반대도 가능

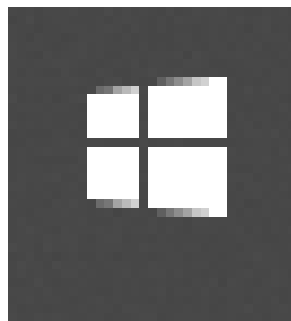
- 단 가장 최신 버전이 가장 안정적으로 동작

실습 환경 구성

- 파이썬 실습 환경
 - 터미널 설치 (옵셔널)
 - miniconda - 3.12.X 버전 이후의 파이썬 인터프리터 포함된 버전
 - 셸 환경에서 miniconda 설정하고 가상환경 만들기
 - 비주얼 스튜디오 코드
- 먼저 결정해 뒀어야 할 것
 - 실습 최상위 디렉토리 - 만약 디렉토리가 없다면 디렉토리를 만들어 둡시다.
 - 저는 C:\python_study를 사용하기로 했습니다. 아래 설명은 이 디렉토리를 기준으로 진행합니다.
 - 디렉토리는 윈도우 탐색기에서 생성할 수 있습니다.

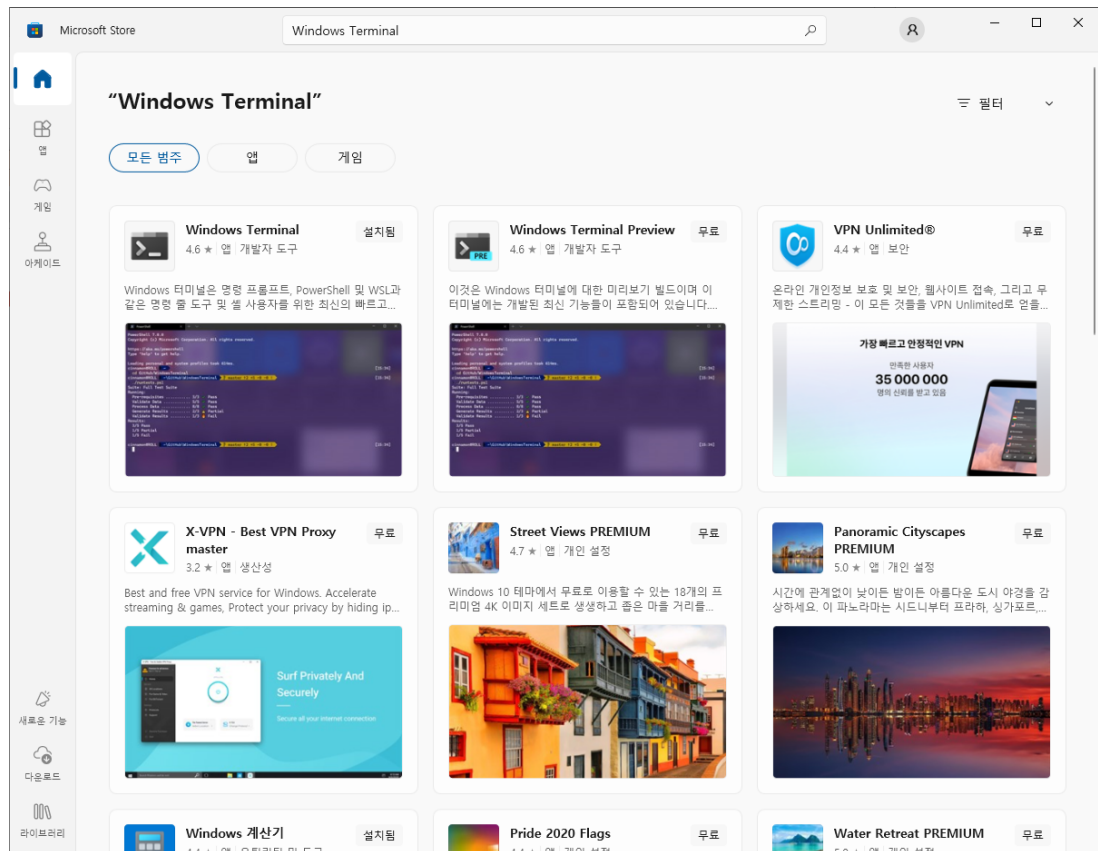
Windows OS

- 터미널 설치
 - 시작버튼 → 검색 → “Windows PowerShell” 프로그램 실행해도 되지만 **터미널 앱을 추천**하며, 터미널 앱을 사용하는 것을 기본으로 설명 드립니다.

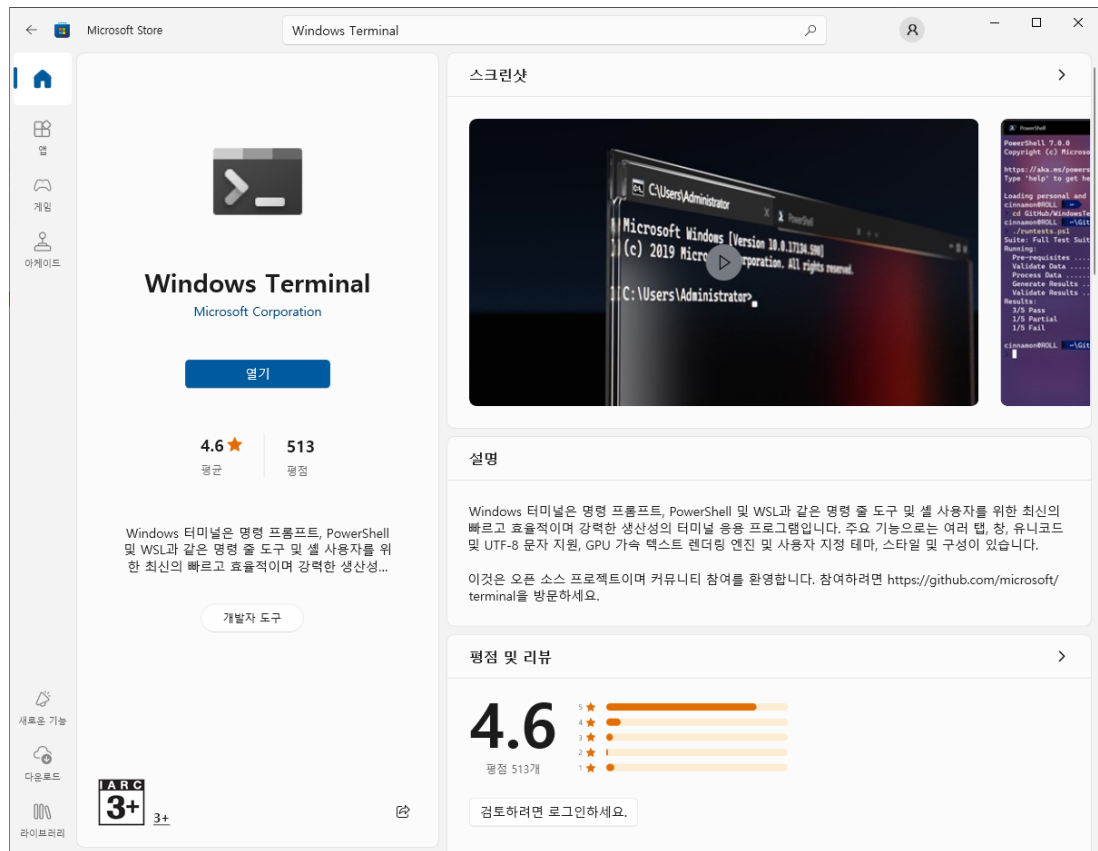


윈도우즈 시작 버튼

- 시작버튼 → 검색 → “Microsoft Store” 프로그램 실행 후 Windows Terminal 검색해 설치

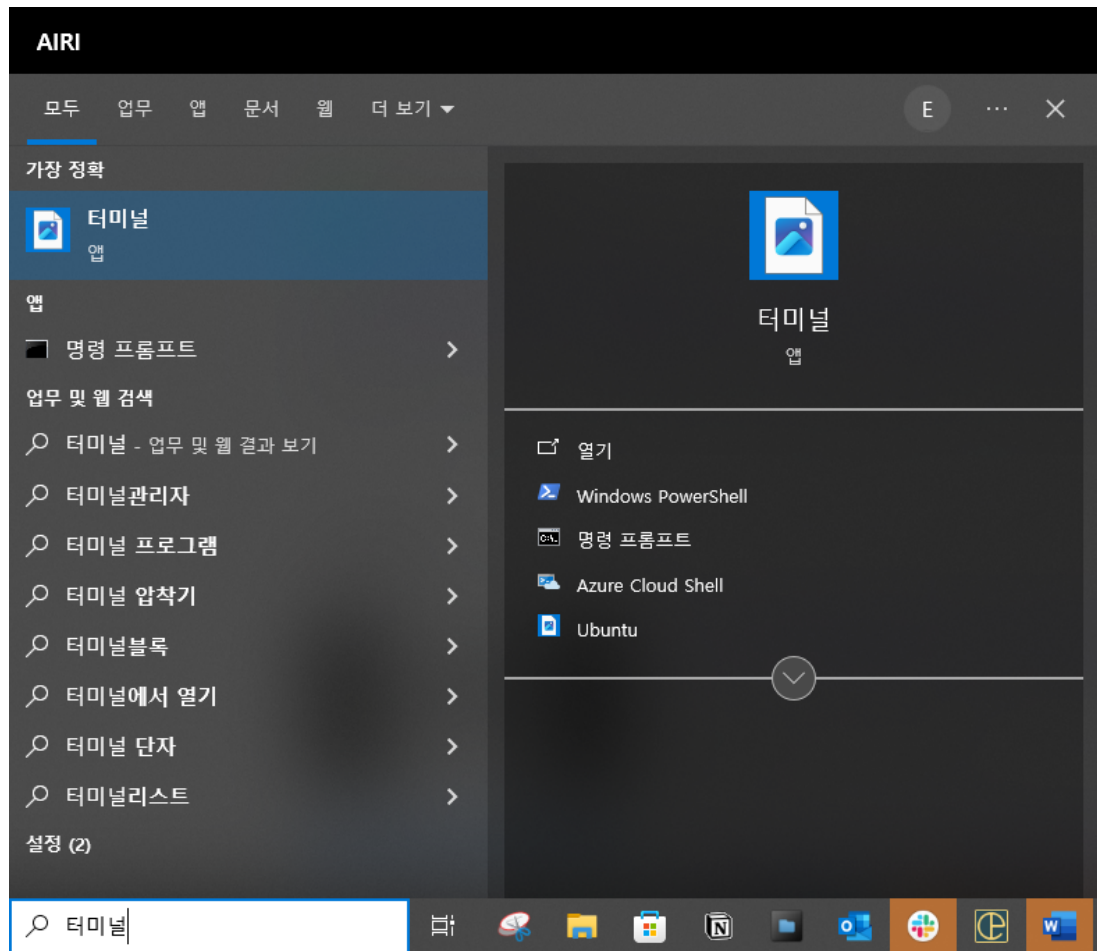


Windows Terminal을 검색하면 위와 같은 화면이 나옵니다. 첫 번째 windows Terminal 항목을 클릭합니다.




클릭하면 위와 같은 화면이 나옵니다. 이미 설치되어 있으므로 '열기' 버튼이 나오는데, 설치되어 있지 않다면 대신 설치 버튼이 나옵니다. 이를 클릭해 설치합니다.

- 설치 후 시작버튼 → 검색 → 터미널 또는 Windows Terminal 검색하면 나오는 한글 이름 '터미널' 앱을 실행합니다.



- 미니콘다 설치 (설치할 디스크에 500MB 정도의 빈 공간 필요)
 - 미니콘다를 내려 받습니다.
 - 웹 브라우저로 <https://docs.anaconda.com/miniconda/> 접속
 - 페이지의 오른쪽에서 Latest Miniconda installer links에서 Windows를 선택 (그림 참조)



[More Resources](#)
[Download Anaconda](#)
[Pricing](#)

Ctrl + K

Section Navigation

Getting Started

- Should I use Anaconda Distribution or Miniconda?
- Anaconda Distribution
- Miniconda**
 - System requirements
 - Latest Miniconda installer links by Python version
 - Installing Miniconda
 - Miniconda release notes
 - Other resources
 - Miniconda hash information
- Anaconda Learning

Package Tools

- Working with conda (CLI)
- Navigator (GUI)
- Anaconda Notebooks
- Anaconda.org
- Python in Excel

Business Solutions

- Package Security Manager (Cloud)
- Package Security Manager (On-prem)
- Data Science & AI Workbench
- Anaconda AI Navigator
- AE4 (Legacy)

Reference

- API
- Glossary
- Help and support
- Troubleshooting
- Disabling the anaconda-anon-usage package
- Cloud Changelog
- Download our documentation
- Legal

Miniconda

Miniconda is a free minimal installer for conda. It is a small bootstrap version of Anaconda that includes only conda, Python, the packages they both depend on, and a small number of other useful packages (like pip, zlib, and a few others).

If you need more packages, use the `conda install` command to install from thousands of packages available by default in Anaconda's public repo, or from other channels, like conda-forge or bioconda.

Is Miniconda free for me?

Miniconda is free for anyone to use! However, access to Anaconda's public repository of packages is only free to individuals and small organizations (<200 employees). A paid license is required for larger organizations and anyone embedding or mirroring Anaconda's repository. See the [TOS](#) for details.

Should I use Miniconda or Anaconda?

The [Anaconda or Miniconda](#) page lists some reasons why you might want one installation over the other.

Latest Miniconda installer links

This list of installers is for the latest release of Python: 3.12.4. For installers for older versions of Python, see [Other installer links](#). For an archive of Miniconda versions, see <https://repo.anaconda.com/miniconda/>.

Latest - Conda 24.7.1 Python 3.12.4 released Aug 22, 2024

Platform	Name	SHA256 hash
Windows	Miniconda3 Windows 64-bit	ff8ab50f0303c7b9097387967ac2a721016d0200691870ff4e172fc14930eb7
macOS	Miniconda3 macOS Intel x86 64-bit bash	5cfb85d81d94df3ef3265f2247aef32a35aeb450ea71c3a204ced384fb87d
	Miniconda3 macOS Intel x86 64-bit pkg	e31844adec03a69e274538a796c3b4183cd2cbc7b90fd0ea98b591a6313a330b

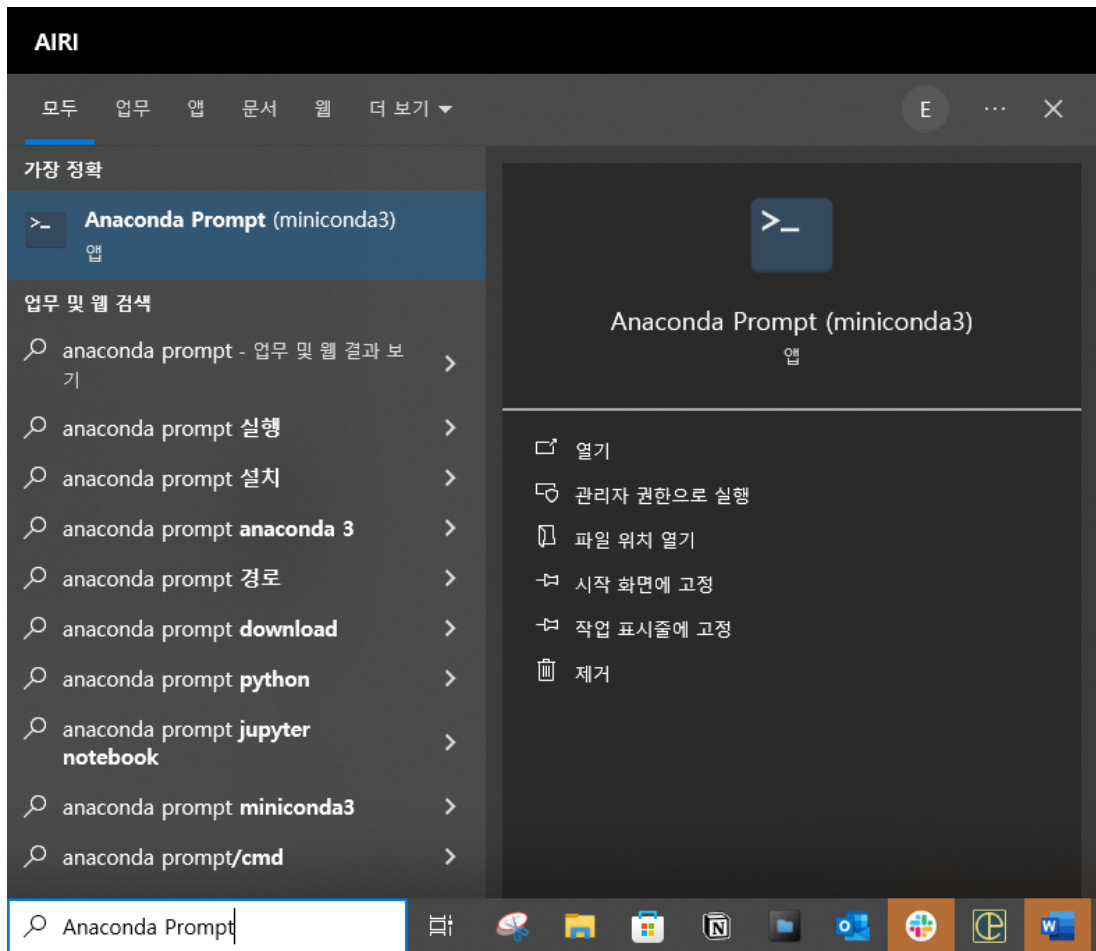
On this page

[Latest Miniconda installer links](#)

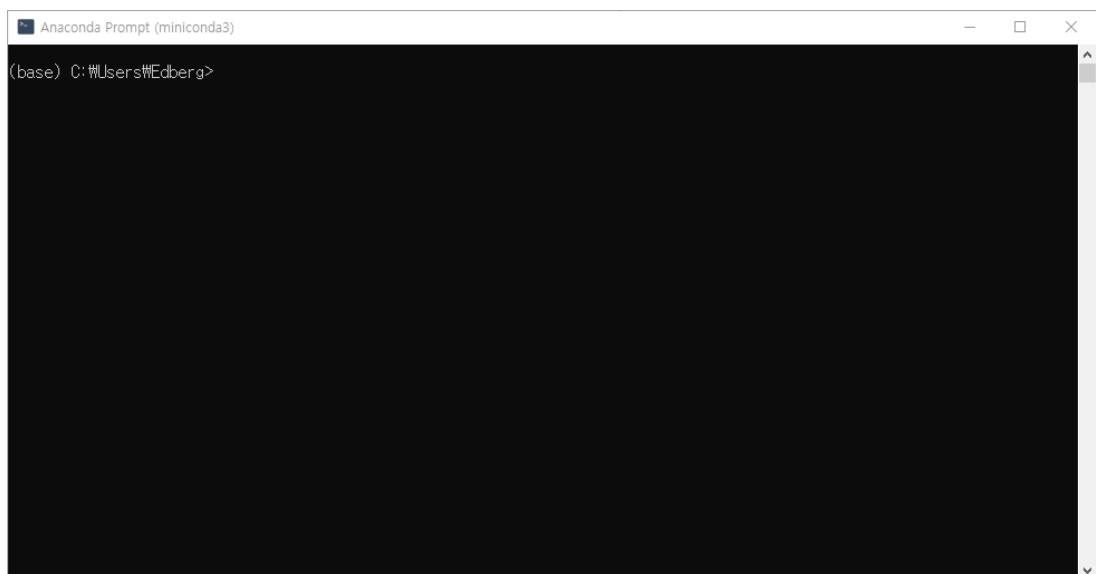
[Quick command line install](#)

Windows 옆의 붉은색 박스의 링크를 클릭하면 미니콘다 실행파일이 다운로드된다.

- 내려받은 설치 파일 (.exe 파일)을 탐색기에서 더블클릭해서 실행합니다.
 - 특별히 바꿀 설정이 없으며 Next를 클릭합니다.
 - 다만, 설치할 디렉토리를 물어보는데, 이 디렉토리를 어디엔가 기록해 둡시다.
 - 이하 이 디렉토리가 C:\Users\Edberg\miniconda3 (저의 경우) 이라고 가정하고 설명합니다.
 - 여기서 설치할 디스크나 디렉토리를 변경할 수 있습니다.
- 설치가 성공적으로 되었다면 시작버튼 → 검색 → Anaconda Prompt를 검색해서 나오는 Anaconda Prompt (miniconda3) 앱을 실행해 봅시다.

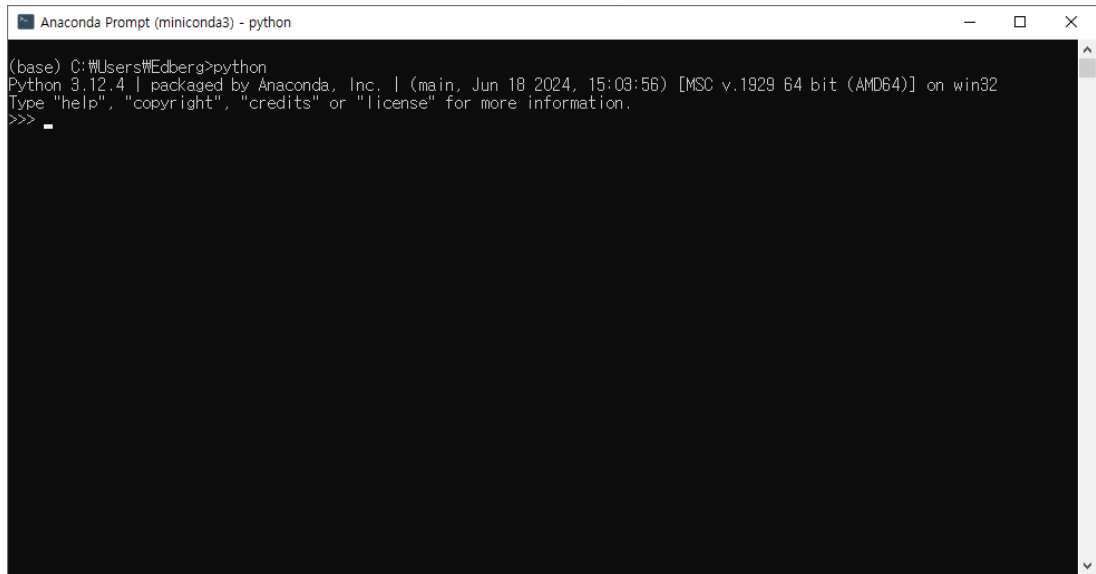


- 아래 그림처럼 (base) 로 시작하는 프롬프트가 뜨는 터미널 창이 실행되면 잘 설치된 것입니다.

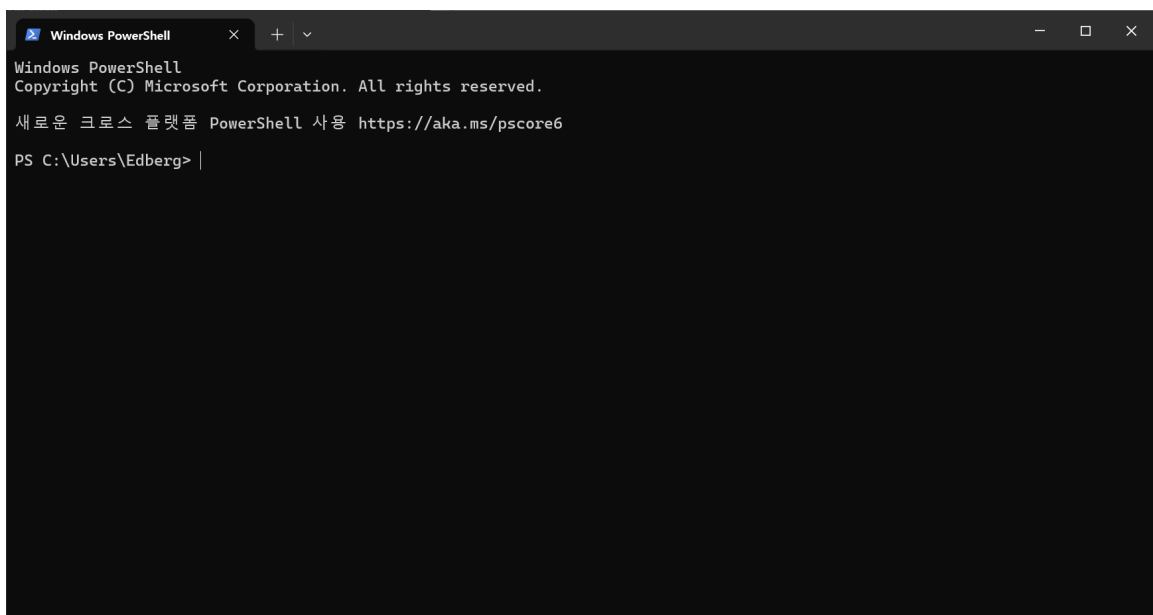


- 이 화면에서 `python` 을 입력하고 엔터키를 누르면 다음과 같이 파이썬 인터프리터가 실행됩니다. 이 때 파이썬 버전이 나오는데, 이 버전이 3.12.X 보다 큰 값이어야 함

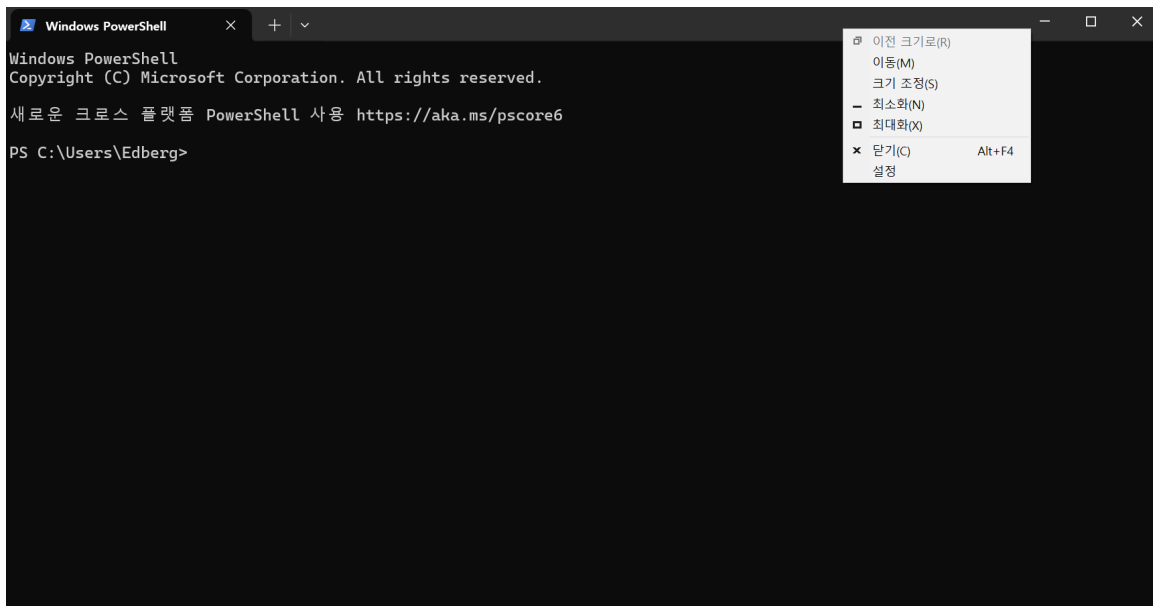
니다.



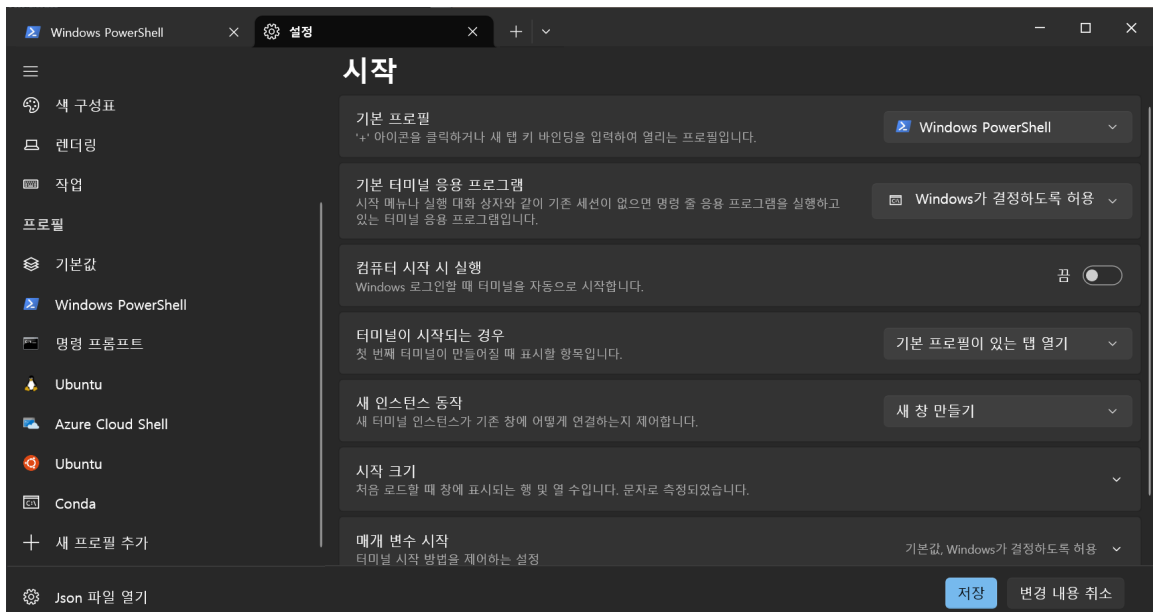
- 이 화면이 바로 **"파이썬 대화형 환경" 실습 환경**입니다.
 - 파이썬 인터프리터를 종료할 때는 Ctrl키와 Z키를 동시에 누르고 엔터키를 입력합니다. (윈도우의 경우, 리눅스나 MacOS에서는 Ctrl키와 D키를 동시에 누르고 엔터키를 입력)
- 윈도우즈 터미널 환경에서 miniconda 설정하고 가상환경 만들기
 - 앞에서 설치한 터미널 앱을 실행합니다.



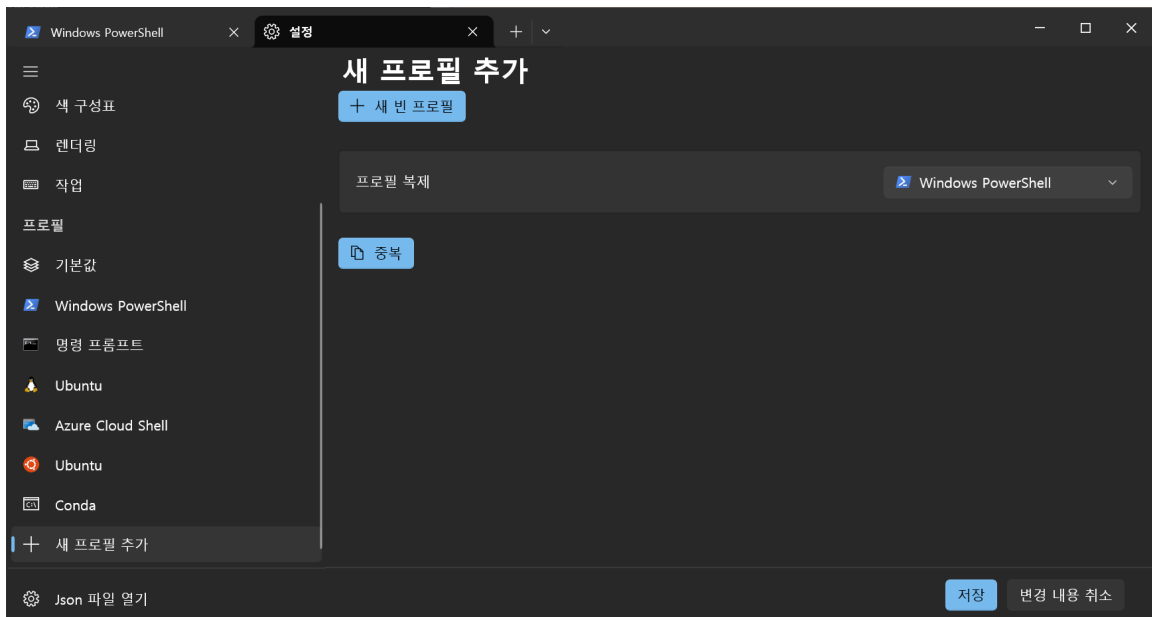
- 터미널앱의 상단 탭 바의 빈 공간에서 마우스 오른쪽 버튼을 클릭하면 팝업 메뉴가 나옵니다. 여기서 '설정'을 선택합니다.



- 설정의 왼쪽 메뉴에 보면 프로필 그룹이 있습니다. 이 프로필 그룹의 제일 아래 **+ 새 프로필 추가**를 선택합니다.



- 새 빈 프로필을 선택합니다.

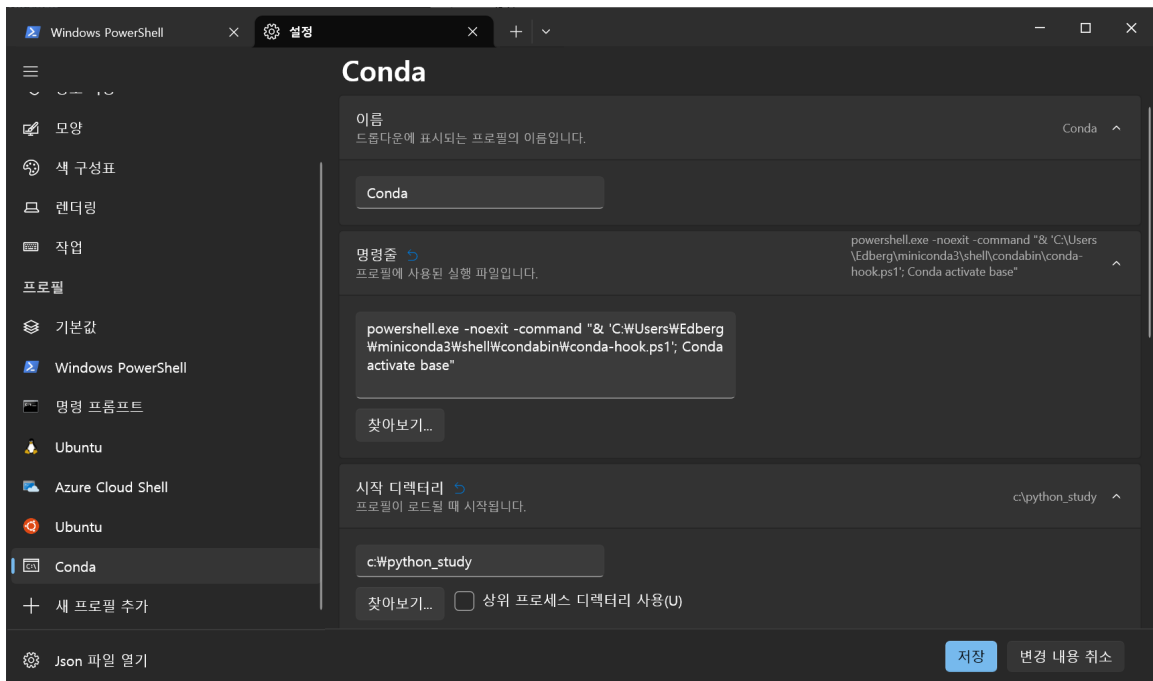


◦ 다음을 설정합니다.

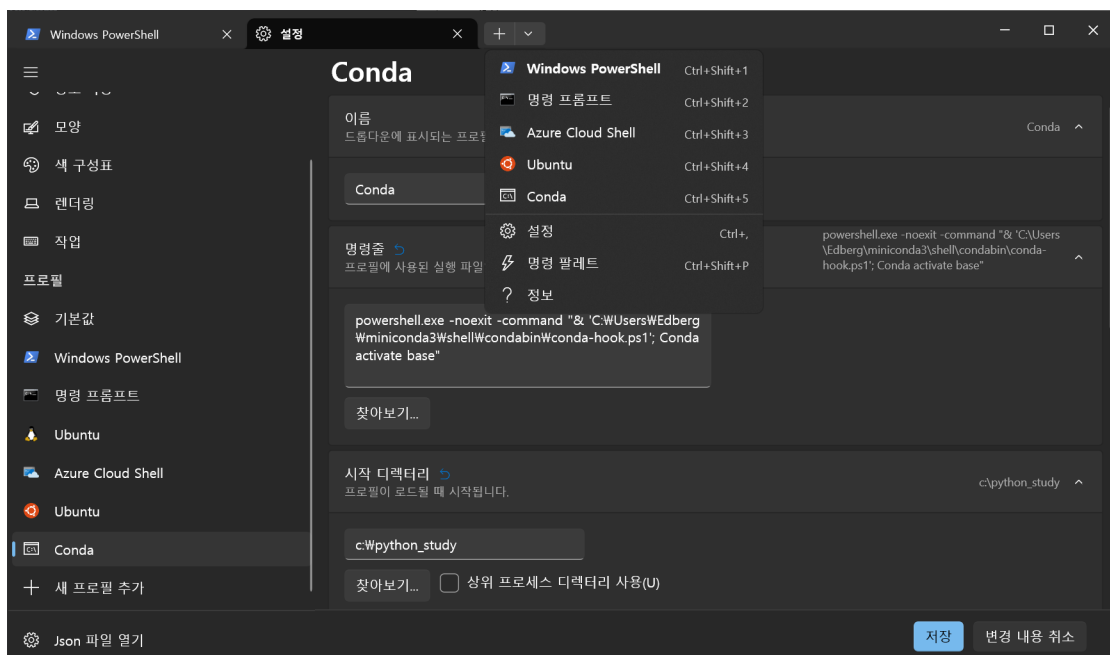
- 이름 - 적당한 프로필 이름을 선택합니다. (저는 Conda로 입력합니다.)
- 명령줄 - 다음과 같이 입력합니다.

```
powershell.exe -noexit -command "& 'C:\Users\Edberg\miniconda3\shell\condabin\conda-hook.ps1'; Conda activate base"
```

- 여기서 C:\Users\Edberg\miniconda3은 그대로 입력하지 마시고 미니콘다를 설치할 때 각자 선택한 디렉토리로 바꾸어 입력합니다.
- 시작 디렉터리 - 미리 결정하고 만들어 둔 실습 최상위 디렉토리를 입력합니다. 저는 C:\python_study 입니다.
- 상위 프로세스 디렉터리 사용(U)는 클릭을 해제해야 입력할 수 있습니다.

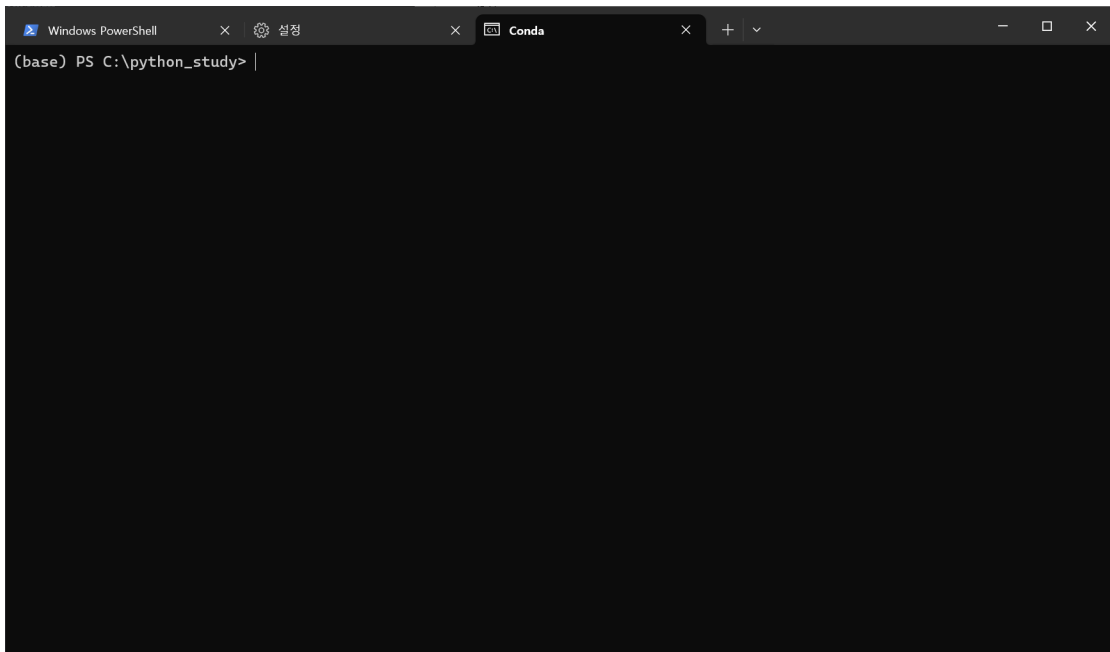


- 제일 아래 저장 버튼을 클릭합니다. 그러면 위에 그림 처럼 왼쪽 프로필 그룹에 입력한 프로필 이름 (저는 Conda) 가 나타납니다.
- 상단 탭 옆에 + 버튼과 v 버튼이 있습니다. v 버튼을 선택하면 그림과 같이 프로필을 선택할 수 있습니다.



- 선택 가능한 프로필 중 Conda를 선택하면 다음과 같이 miniconda를 사용할 수 있는 터미널 창이 나옵니다.

- 아래 그림처럼 프롬프트에 (base)와 지정한 실습 디렉토리 위치를 확인할 수 있어야 합니다.



- 파이썬 가상환경이란?
 - 파이썬 프로그래밍을 하다보면 수많은 외부 라이브러리를 가져다 쓰게 됩니다. 경우에 따라서는 라이브러리끼리 충돌할 수도 있고, 라이브러리의 버전 차이로 인한 문제가 발생할 수도 있습니다.
 - 이러한 문제를 회피하기 위해서 파이썬은 프로그램 또는 프로젝트마다 서로 다른 라이브러리를 설치해서 동작할 수 있는 프로그램 또는 프로젝트 별 별도의 환경을 만들 수 있는 방법을 제공합니다.
 - 이렇게 만들어진 환경을 파이썬 가상환경이라고 합니다.
- 미니콘다에서 파이썬 가상환경 만들기
 - 미니콘다를 사용하도록 설정한 터미널 창에서 다음과 같이 입력합니다. (여기서 `p_study` 는 지정 가능한 가상환경의 이름이며, 변경해도 됩니다.)

```
conda create --name p_study
```

 - 다음과 같이 가상환경이 설치되는 것을 확인할 수 있습니다. (y + 엔터키를 한 번 눌러줘야 진행됩니다.)

```
Windows PowerShell  X  설정  X  Conda  +  v  -  □  X
(base) PS C:\python_study> conda create --name p_study
Retrieving notices: ...working... done
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Edberg\miniconda3\envs\p_study

Proceed ([y]/n)? y
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate p_study
#
# To deactivate an active environment, use
#
#   $ conda deactivate
(base) PS C:\python_study>
```

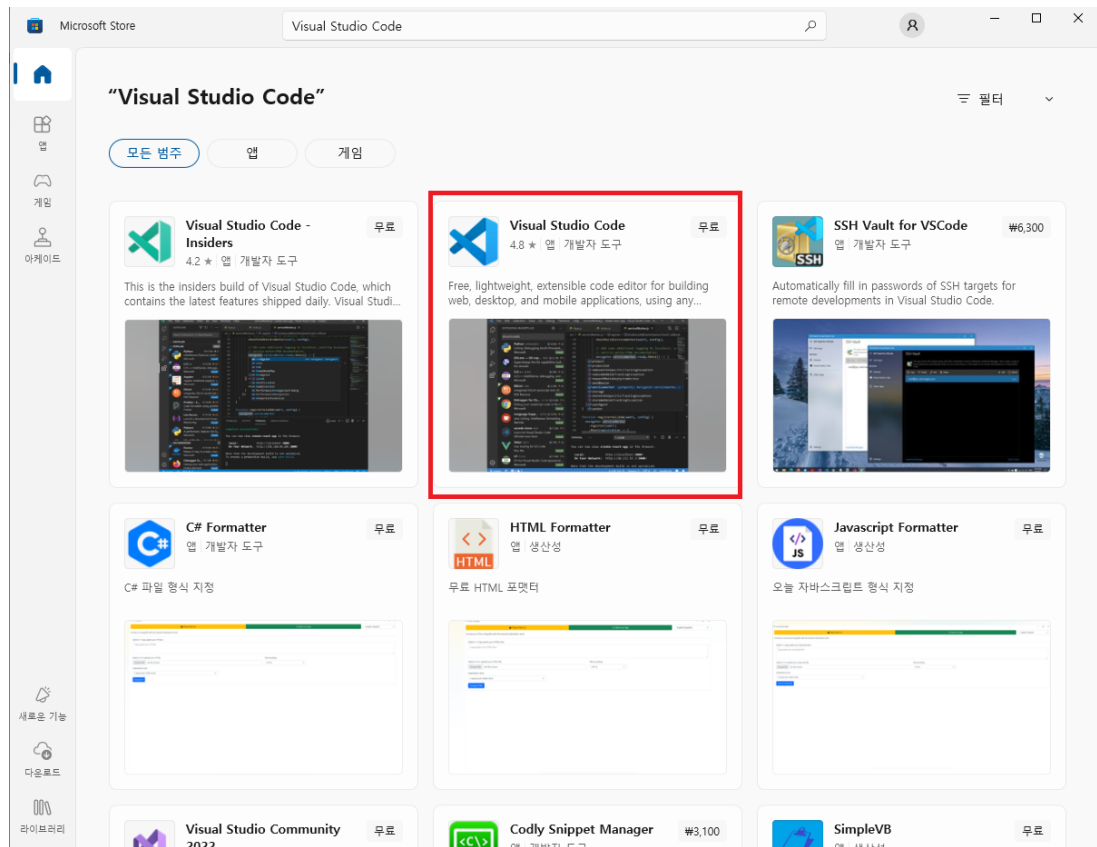
- 설치한 가상 환경은 `conda activate p_study` 명령으로 활성화 할 수 있습니다. (만약 `p_study` 대신 다른 이름을 사용했다면, 그 이름을 사용합니다.)

```
(base) PS C:\python_study> conda activate p_study
(p_study) PS C:\python_study>
```

- 위에서 처럼 (base)가 지정한 가상환경의 이름으로 바뀌게 됩니다.
- 작업을 종료하고 가상환경을 종료하려면 `conda deactivate` 명령을 사용합니다.

```
(p_study) PS C:\python_study> conda deactivate
(base) PS C:\python_study>
```

- 다시 (base) 환경으로 돌아왔습니다.
- 비주얼 스튜디오 코드 설치
 - 비주얼 스튜디오 코드는 마이크로소프트에서 오픈소스로 개발해 공개한 프로그램 통합 개발 환경(IDE)입니다. 마이크로소프트에서 개발해 상용으로 판매하는 비주얼 스튜디오와 착각하지 맙시다.
 - Windows Store 앱을 실행하고 Visual Studio Code를 검색합니다.

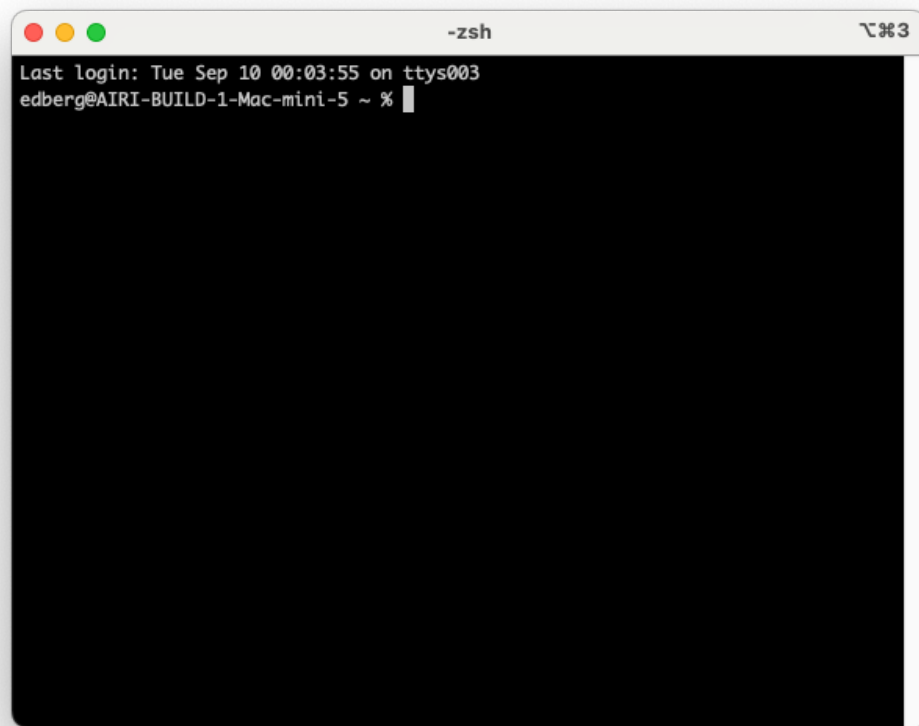


검색 결과에서 붉은색 박스 프로그램이 비주얼 스튜디오 코드입니다.


- Visual Studio Code를 선택한 후, 해당 페이지에서 설치 버튼을 눌러 설치합니다.

MacOS

- 이하의 설명에서 실습 최상위 디렉토리는 ~/python_study로 하고, 이를 기준으로 설명 드립니다.
- iTerm2 설치
 - 기본 터미널 앱을 사용해도 좋습니다. 다만 iTerm 등 풍부한 기능을 제공하는 좋은 터미널 프로그램이 많이 있습니다. 저는 iTerm을 기준으로 설명 드립니다.
 - <https://iterm2.com/> 사이트에서 iTerm2를 다운로드 받아 설치할 수 있습니다. iTerm2는 GPL v2 라이선스로, 상업적으로 이를 판매용으로 사용하지 않는다면 자유롭게 사용할 수 있습니다.
 - 다운로드 받은 압축 파일을 클릭해 압축을 해제하면 iTerm 앱이 다운로드 폴더에 저장됩니다. 이 iTerm 앱을 응용 프로그램 폴더로 이동하면 설치가 완료됩니다.
 - 설치가 완료되면 Finder의 응용 프로그램 폴더 또는 Launchpad에서 iTerm을 실행할 수 있습니다.



- 터미널 화면에서 `mkdir -p ~/python_study` 를 입력해서 실습 디렉토리를 만들 수 있습니다.
 - `~/python_study` 대신 각자 정한 실습 최상위 디렉토리를 입력하시기 바랍니다.
- 미니콘다 설치 (설치할 디스크에 500MB 정도의 빈 공간 필요)
 - 미니콘다를 내려 받습니다.
 - 웹 브라우저로 <https://docs.anaconda.com/miniconda/> 접속
 - 페이지의 오른쪽에서 Latest Miniconda installer links에서 macOS의 다운로드 링크 중 적절한 하나를 선택합니다. (다운받는 파일이 bash가 아닌 pkg 인지 확인합니다.)
 - 만약 인텔 CPU를 사용하는 구형 맥북 또는 아이맥인 경우는 [Miniconda3 macOS Intel x86 64-bit pkg](#)를 선택합니다. (아래 그림의 위쪽 박스 참고)
 - 애플 실리콘 CPU를 사용하는 신형 맥북 또는 아이맥인 경우는 [Miniconda3 macOS Apple M1 64-bit pkg](#)를 선택합니다. (아래 그림의 아래쪽 박스 참고)



[More Resources](#)
[Download Anaconda](#)
[Pricing](#)

Section Navigation

Getting Started

- Should I use Anaconda Distribution or Miniconda?
- Anaconda Distribution
- Miniconda**
- System requirements
- Latest Miniconda installer links by Python version
- Installing Miniconda
- Miniconda release notes
- Other resources
- Miniconda hash information
- Anaconda Learning

Package Tools

- Working with conda (CLI)
- Navigator (GUI)
- Anaconda Notebooks
- Anaconda.org
- Python in Excel

Business Solutions

- Package Security Manager (Cloud)
- Package Security Manager (On-prem)
- Data Science & AI Workbench
- Anaconda AI Navigator
- AEA (Legacy)

Reference

- API
- Glossary
- Help and support
- Troubleshooting
- Disabling the anaconda-anaor-usage package
- Cloud Changelog
- Download our documentation
- Legal

Latest Miniconda installer links

This list of installers is for the latest release of Python: 3.12.4. For installers for older versions of Python, see [Other installer links](#). For an archive of Miniconda versions, see <https://repo.anaconda.com/miniconda/>.

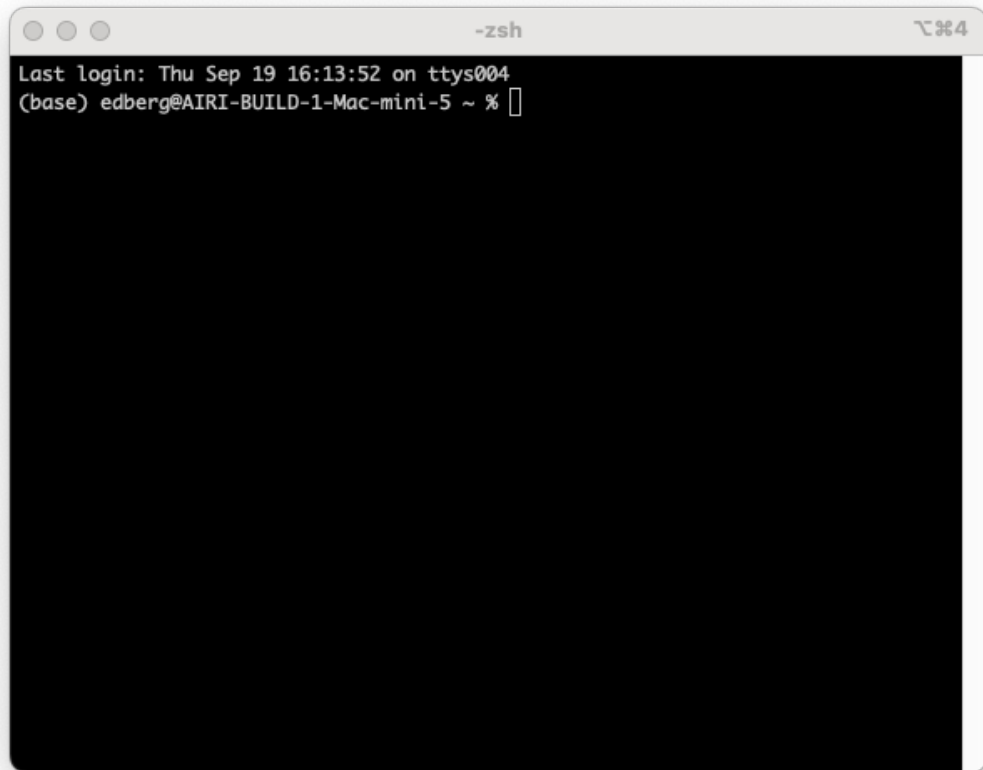
Latest - Conda 24.7.1 Python 3.12.4 released Aug 22, 2024

Platform	Name	SHA256 hash
Windows	Miniconda3-Windows-64-bit	ff8ab58f0383c7b9897387967ac2a721816d82860187eff4e172fc14938e
	Miniconda3-macOS-Intel-x86_64-bit-bash	5cfb85d83d94dfe3ef3265f2247aef32a35aeb450ea77c3a704cefd384fb
macOS	Miniconda3-macOS-Intel-x86_64-bit-pkg	e31844adec83a60e274538a796c3b4183cd2c7b09f08ea9eb591a6313a3
	Miniconda3-macOS-Apple M1-64-bit-bash	e7ef5a899f9383d14d5b15ae761d54a8cd9bf3c4de18a372ef0405d8f5f78
Linux	Miniconda3-Linux-64-bit	33442cd38138f33dcb4a932b938ee95398be98344d1f4c38f7e757cd2118
	Miniconda3-Linux-aarch64-64-bit	b40ce1e233cda38ce37185de627e646ae8e04b36373eacfc07fa8c35949f
Linux	Miniconda3-Linux-s390x-64-bit	5a454c59314f63a9b060e2e27d58f4a2516c77a7bde919fc11d3c083c6b

On this page

- [Latest Miniconda installer links](#)
- [Quick command line install](#)

- 내려받은 설치 파일 (.pkg파일)을 Finder에서 더블클릭해서 실행합니다.
 - 특별히 바꿀 설정이 없으며 '계속'과 '동의'를 클릭하면 설치가 완료됩니다.
 - 설치가 완료된 후 위에서 설치한 iTerm을 실행하면 다음과 같이 프롬프트에 (base)라고 나옵니다.



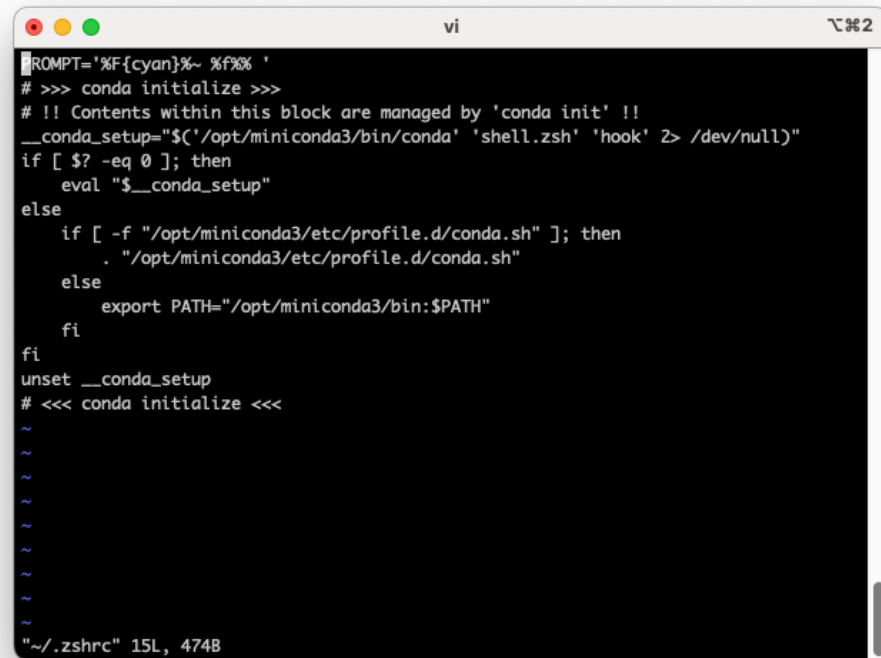
- 실습 디렉토리로 이동할 때는 cd 명령을 이용합니다.
 - `cd ~/python_study`
- 다만, 설치할 디렉토리를 물어보는데, 이 디렉토리를 어디엔가 기록해 둡시다.
 - 이하 이 디렉토리가 C:\Users\Edberg\miniconda3 (저의 경우) 이라고 가정하고 설명합니다.
 - 여기서 설치할 디스크나 디렉토리를 변경할 수 있습니다.

```
(base) edberg@AIRI-BUILD-1-Mac-mini-5 ~ % cd ~/pytho
(base) edberg@AIRI-BUILD-1-Mac-mini-5 python_study %
```

- 팁 - 프롬프트 변경 방법
 - `(base) edberg@AIRI-BUILD-1-Mac-mini-5 python_study %` 부분을 셸에서 프롬프트라고 부릅니다. 유닉스 시스템의 프롬프트에는 계정, 작업기계 이름, 디렉토리 등의 정보가 포함되어 있습니다.

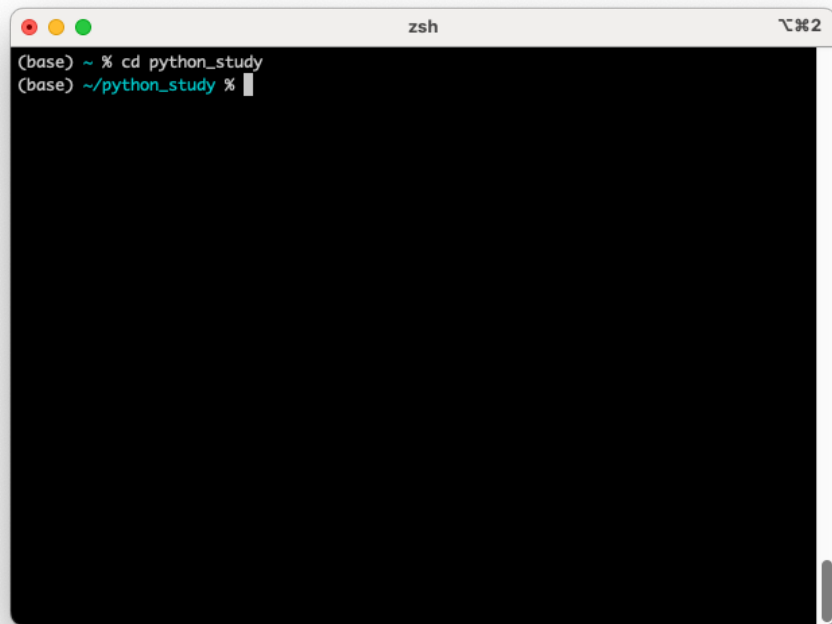
- 다만 이게 너무 길어서 작업에 방해가 된다면, 프롬프트를 짧게 변경할 수 있습니다.
- `~/.zshrc` 파일을 열어보면 `# >>> conda initialize >>>` 로 시작하는 행이 있습니다. 그 바로 앞 줄에 다음 한 줄을 추가하세요.

```
PROMPT='%F{cyan}%~ %f% '
```



```
PROMPT='%F{cyan}%~ %f% '
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$(('/opt/miniconda3/bin/conda' 'shell.zsh' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/opt/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/opt/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/opt/miniconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
~
~
~
~
~
~
~
~
~
~
"~/.zshrc" 15L, 474B
```

- 이후 터미널을 다시 실행하면 다음과 같이 간소화된 프롬프트를 확인할 수 있습니다.



- iTerm 또는 터미널 앱 환경에서 miniconda 가상환경 만들기
 - 파이썬 가상환경이란?
 - 파이썬 프로그래밍을 하다보면 수많은 외부 라이브러리를 가져다 쓰게 됩니다. 경우에 따라서는 라이브러리끼리 충돌할 수도 있고, 라이브러리의 버전 차이로 인한 문제가 발생할 수도 있습니다.
 - 이러한 문제를 회피하기 위해서 파이썬은 프로그램 또는 프로젝트마다 서로 다른 라이브러리를 설치해서 동작할 수 있는 프로그램 또는 프로젝트 별 별도의 환경을 만들 수 있는 방법을 제공합니다.
 - 이렇게 만들어진 환경을 파이썬 가상환경이라고 합니다.
 - 미니콘다에서 파이썬 가상환경 만들기
 - 미니콘다를 사용하도록 설정한 터미널 창에서 다음과 같이 입력합니다. (여기서 `p_study` 는 지정 가능한 가상환경의 이름이며, 변경해도 됩니다.)

```
conda create --name p_study
```
 - 다음과 같이 가상환경이 설치되는 것을 확인할 수 있습니다. (y + 엔터키를 한번 눌러줘야 진행됩니다.)

```
zsh
(base) ~/python_study % conda create --name p_study
Channels:
- defaults
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/miniconda3/envs/p_study

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate p_study
#
# To deactivate an active environment, use
#
#   $ conda deactivate
#
(base) ~/python_study %
```

- 설치한 가상 환경은 `conda activate p_study` 명령으로 활성화 할 수 있습니다. (만약 `p_study` 대신 다른 이름을 사용했다면, 그 이름을 사용합니다.)

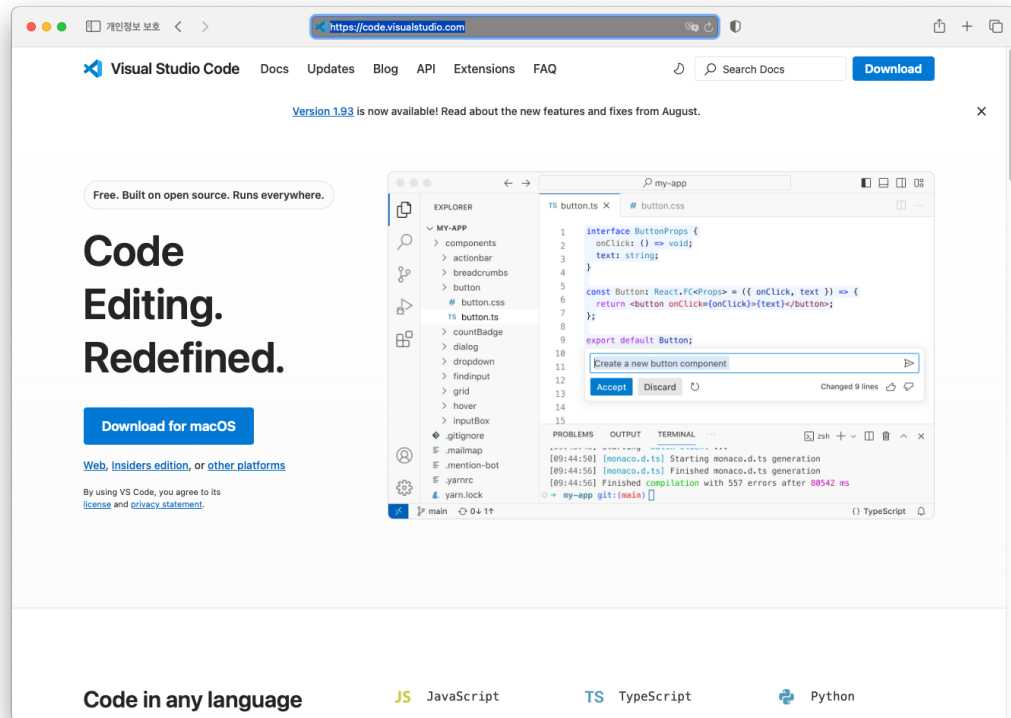
```
(base) ~/python_study % conda activate p_study
(p_study) ~/python_study %
```

- 위에서 처럼 (base)가 지정한 가상환경의 이름으로 바뀌게 됩니다.
- 작업을 종료하고 가상환경을 종료하려면 `conda deactivate` 명령을 사용합니다.

```
(p_study) ~/python_study % conda deactivate
(base) ~/python_study %
```

- 다시 (base) 환경으로 돌아왔습니다.
- 비주얼 스튜디오 코드 설치
 - 비주얼 스튜디오 코드는 마이크로소프트에서 오픈소스로 개발해 공개한 프로그램 통합 개발 환경(IDE)입니다. 마이크로소프트에서 개발해 상용으로 판매하는 비주얼 스튜디오와 착각하지 맙시다.

- 윈도우 OS와 달리 맥오에스용 비주얼 스튜디오는 직접 다운로드 받아서 설치해야 합니다. 비주얼 스튜디오 코드 홈페이지 (<https://code.visualstudio.com>)에 접속합니다.



- 왼쪽의 Download for macOS 버튼을 클릭합니다. 자동으로 비주얼 스튜디오 코드의 설치 파일(압축 파일)이 다운로드됩니다. 이 설치 파일을 클릭해 압축을 해제한 후, 비주얼 스튜디오 코드 앱을 응용 프로그램 폴더로 이동시키면 설치 완료입니다.

실습 디렉토리에 관하여

- 실습에 사용할 파일을 어디에 저장할 지를 결정합니다.
 - 디렉토리 구조는 각자 자유롭게 정하면 되지만 다음과 같이 강의 별로 별도의 디렉토리를 만드는게 좋을 것 같습니다.

(실습 최상위 디렉토리)

lec_01
lec_02
lec_03
...
240925

→ 동영상 강의 1강

→ 동영상 강의 2강

→ 동영상 강의 3강

→ 9월 25일 오프라인 강의

- 저는 앞에서 C:\python_study를 실습 최상위 디렉토리로 정했습니다.(윈도우 기준) 각자 적당한 디렉토리를 결정하시면 됩니다.

- 참고로 윈도우에서 \ 대신 원화 기호로 표시될 수도 있습니다.

비주얼 스튜디오 코드에서 프로그램 만들어보기

- 설치한 비주얼 스튜디오를 실행합니다.
 - 시작버튼 → Visual Studio Code 검색 → Visual Studio Code 앱 실행
- 동영상 강의를 참고합니다.
 - 동영상 강의에는 실습 디렉토리에 공장 sample.py 파일을 만드는데, 디렉토리를 만들고 그 아래에 sample.py 파일을 만들어도 됩니다.
 - 동영상 강의는 miniconda를 기준으로 작성되지 않았습니다. miniconda 기준의 동작은 실습 강의에서 확인할 수 있습니다.