1. 파이썬 환경 기본

복습	☐
개념	
블로그	☐
페이지	

이번 토픽에서는 터미널을 통해서 문서 파일을 Sublime Text를 여는 방법을 자주 사용할 건데요. 이걸 하기 위해서는 설정이 필요합니다. 별로 어렵지는 않은데요. 그냥 터미널을 열고 아래 나온 커맨드를 실행하세요.

```
sudo ln -s /Applications/Sublime\ Text.app/Contents/SharedSupport/bin/subl /usr/local/bin/subl
```

이 설정을 하고 나서는 터미널으로도 Sublime Text를 실행할 수 있습니다. 바로 `subl` 라는 커맨드를 사용하면 되는데요. `file.txt` 라는 파일이 있으면,
`subl file.txt`
이렇게 해당 파일을 열 수 있습니다.

4. 파이썬 파일을 실행하는 법

```
touch hello.py
# hello.py 파일 생성
subl hello.py
# hello.py 파일을 sublimetext로 열어서 작성한 이후
python3 hello.py
# 이렇게 하면 실행된다.
```

5. 파이썬 인터렉티브 모드

터미널에서 python3 입력하면 파이썬 인터렉티브 모드(반응형 모드, 파이썬쉘)이 실행된다.
여기서는 줄 단위로 파이썬 코드 실행 가능
여러 줄이 필요한 코드를 작성할 수 있다. 실행하려면 엔터 두번 치면 된다.
참고로 control + l 을 누르면 화면에 있는걸 지워주는데 지정한 변수 등과 같은 걸 지우는건 아니다. 그냥 안보이게만 한다.
종료하고 싶으면 quit()함수를 작성하고나 control + D를 누르면 된다.

6. 파이썬 인터프리터

파이썬에 경우 코드를 머신코드로 바꿔주는 프로그램을 파이썬 인터프리터(python interpreter)라고 부른다.

파이썬 인터프리터(Python Interpreter)

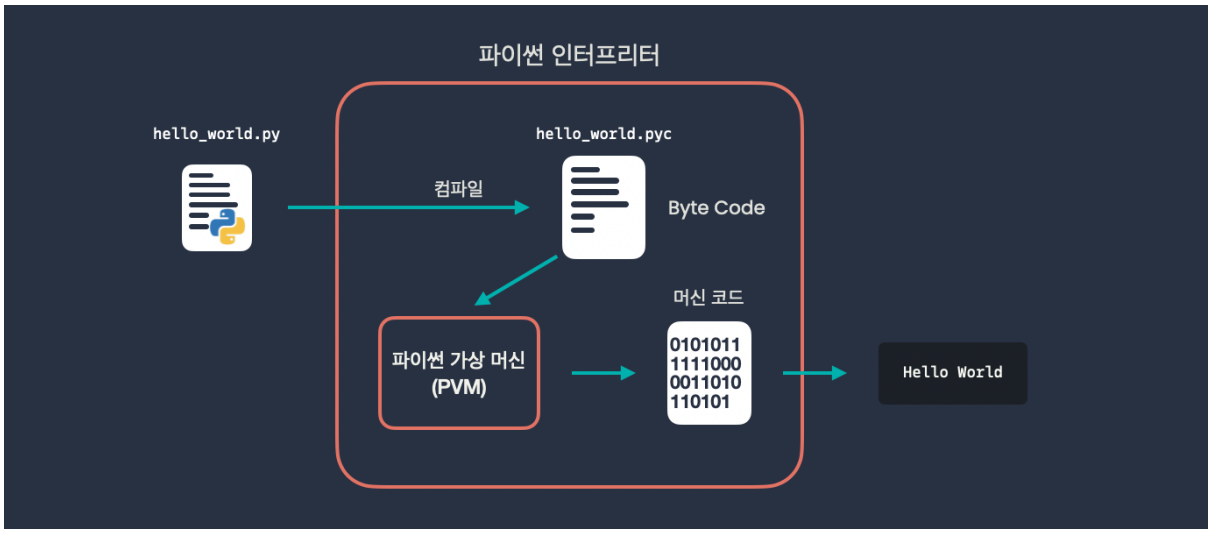
- 아무 문서 편집 도구에서도 파이썬 코드는 쓸 수 있다
- 중요한 건 파이썬 코드를 머신 코드로 바꿔주는 인터프리터다
- 파이썬 언어와 인터프리터를 각각, 그리고 합쳐서 그냥 파이썬이라고 부른다

7. 파이썬 인터프리터 심화

사실 파이썬 환경을 이해하고 다루기 위해서는 그냥 파이썬 인터프리터라는 프로그램이 파이썬 코드를 컴퓨터가 이해할 수 있는 머신 코드로 바꾸고, 실행시켜준다는 내용만 알면 되는데요. 더 자세하게 알고 싶은 분들을 위해서 파이썬 인터프리터가 내부적으로 어떻게 작동하는지 간단하게 정리해봤습니다.

파이썬 인터프리터 안에서 실제로 일어나는 일

`hello_world.py` 라는 파일에 `print('Hello World')` 라는 코드가 있고, 실행시킨다고 하겠습니다.



영상에서는 말씀드리지 않았지만, 사실 파이썬 인터프리터는 코드를 바로 머신 코드로 바꾸지 않고요. 먼저 코드 전체를 bytecode라는 머신 코드와는 다른 중간 단계의 코드로 바꿔줍니다. 이 단계를 컴파일이라고 표현하기도 하는데요. 이렇게 해서 바뀐 코드는 파일 이름 뒤에 .pyc 확장자 명이 붙고 (`hello_world.pyc` 나 유사한 형식), **pycache**라는 폴더에 저장됩니다.
이때 문법 확인 등 단순 검사를 통과하지 못하면 오류가 나고 bytecode가 만들어지지 않습니다.
그 다음에 인터프리터 안에 있는 파이썬 가상 머신(pvm/python virtual machine)이란 하위 프로그램이 파이썬 코드가 아니라 이 bytecode를 한 줄씩 보며, 머신 코드로 다시 바꿔줍니다. 한 줄을 바꿀 때마다 바로바로 실행시키죠.
위 코드에 대한 bytecode를 풀어서 보여드리면 이렇게 되는데요. 파이썬 가상 머신이 이해할 수 있는, 조금 더 머신 코드에 가까운 형태입니다. (실제로는 사람이 읽을 수 없는 형태로 저장됩니다)

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('Hello World')
            4 CALL_FUNCTION            1
            6 POP_TOP
            8 LOAD_CONST               1 (None)
           10 RETURN_VALUE
```

중요하진 않지만 간단히 설명드리면, 각 줄은 파이썬 가상 머신이 이해할 수 있는 코드입니다. 한 줄씩을 instruction, 명령이라고 부르죠.

1. **가장 위 왼쪽 1은 파일에서 코드 내용이 몇 번째 줄인지를 나타냅니다. 지금 같은 경우 파일에 한 줄밖에 없기 때문에 다른 내용 없이 1만 나와있습니다.**
2. **그 다음 숫자 열은 각 명령이 차지하는 바이트 주소를 의미합니다. 파이썬 3.6 이후로는 각 명령은 2byte기 때문에 2씩 차이가 나기 때문에 2의 배수로 커지는데요. 다른 버전들에서는 다르게 출력될 수 있습니다.**
3. **그 다음 열은 명령 이름들입니다. 각 명령은 파이썬 가상 머신이 이해할 수 있는 특정 일을 시키는데요. "어떤" 일을 하는지를 나타냅니다.**
4. **그 다음 두 열들은 명령들에 넘겨지는 아규먼트들에 대한 정보입니다. 쉽게 설명드리면 두 번째 줄에 `LOAD_CONST` 이라는 명령을 수행할 때, 어떤 내용에 대해서 해당 명령을 처리할 건지를 읽기 쉽게 표현해놓은 거죠. 지금 같은 경우 문자열 `'Hello World'` 을 사용해서 명령을 실행하라고 알려주고 있는 거죠.**

파이썬 인터프리터는 먼저 이 중간 단계 코드인 bytecode를 만든 후, 인터프리터 안에 있는 파이썬 가상 머신을 사용해서 bytecode의 내용을 한 줄 씩 머신 코드로 바꾸면서 바로 실행시킵니다.
특정 코드 전체를 한 번에 다 **머신 코드**로 바꾸고, 전부를 바꾼 파일을 컴퓨터에게 실행시키는 번역(translate)의 개념이 아니라, 중간 단계의 bytecode 코드를 한 줄씩 바꾸고 실행시키는 통역(interpret)의 개념이기 때문에 이걸 해주는 프로그램을 파이썬 인터프리터라고 부릅니다.

컴파일러 vs 인터프리터 언어

프로그래밍 언어들은 흔히 두 가지로 분류합니다. 컴파일러 언어와 인터프리터 언어인데요.
파이썬은 인터프리터 언어입니다.
컴파일러 언어들은 코드를 바로 컴퓨터가 이해할 수 있는 머신 코드로 '번역'합니다. 코드 전체를 다른 언어로 바꾸는 걸 컴파일이라고 부르죠. 그리고 이렇게 컴파일된 코드를 미리 만들어놓으면 이것만 사용해서 프로그램을 돌릴 수 있습니다. 가장 대표적인 컴파일러 언어가 C

와 C++ 같은 언어입니다.
인터프리터 언어들은 컴퓨터가 바로 이해할 수 있는 실행 코드를 만드는 게 아니라, 중간 매개체, 인터프리터를 사용해서 코드를 '통역'합니다. 파이썬에서는 파이썬 가상 머신이 이 역할을 합니다.
사실 파이썬의 경우 컴파일러와 인터프리터의 특징을 어느 정도씩 다 갖고 있습니다. 위에서 얘기했듯이 파이썬은 먼저 코드를 bytecode로 컴파일합니다. 하지만 이 파일은 머신 코드가 아니죠. 이 중간 단계의 코드를 파이썬 가상 머신이 한 줄씩 머신 코드로 통역해서 실행시킵니다. 단계가 두 개로 나눠지긴 하지만, 저희가 파이썬 코드 실행시킬 때는 한 단계처럼 느껴집니다.
컴파일러 언어들은 컴파일 하는 단계와 컴파일된 파일을 따로 실행시키는 단계, 두 개로 나눠져있고 인터프리터 언어들은 보통 한 단계로 코드를 실행할 수 있습니다.
일반적으로 컴파일러 언어로 작성한 코드들이 더 빠르게 실행이 됩니다. 컴파일러 언어들은 이미 컴퓨터가 바로 이해할 수 있는 실행 파일을 만들어주기 때문이죠. 비유를 하자면 영어로 된 글을 한글로 바꿔서 읽을 때 이미 한글로 번역된 글을 읽는 거랑, 영어로된 글을 한 줄씩 한글로 통역해가며 읽는 거랑 글을 읽는 속도가 차이가 날 수밖에 없겠죠?
프로그래밍 언어를 항상 컴파일러나 인터프리터 언어로 항상 나눠야 되거나 엄청 명확하게 나눠지는 건 아닙니다. 하지만 굳이 나눌 때는 코드를 실행시킬 때, 이미 머신 코드로 컴파일된 파일을 그대로 사용/실행 할 수 있으면 컴파일러 언어, 실행시키는 와중에 줄 단위로 한 번 더 머신 코드로 바꿔야 하면 인터프리터 언어라고 하죠.

9. 파이썬 버전

python 2.7.1
이렇게 버전은 major.minor.micro 버전으로 나뉜다.

major 버전

이전 버전과 호환성이 안 맞을 정도로 큰 변화가 있는 버전

minor 버전

새로운 기능 출시
낮은 버전의 코드를 높은 버전의 인터프리터로 실행해도 대부분 오류 없이 실행할 수 있다.

micro 버전

버그 수정 출시
같은 minor 버전 안에서 실수로 틀렸던 내용을 고쳐서 출시되는 버전
특정 minor 버전을 쓸 때는 항상 최신 micro 버전을 쓰는 게 좋다

파이썬 버전 선택

가장 최신 minor 버전보다 한 minor 버전 낮은 최신 micro 버전
예를 들어 현재 최신 버전이 python 3.9.6이라면 선택버전은 python 3.8 의 최신 버전을 선택하면 된다.

새로운 버전 사용

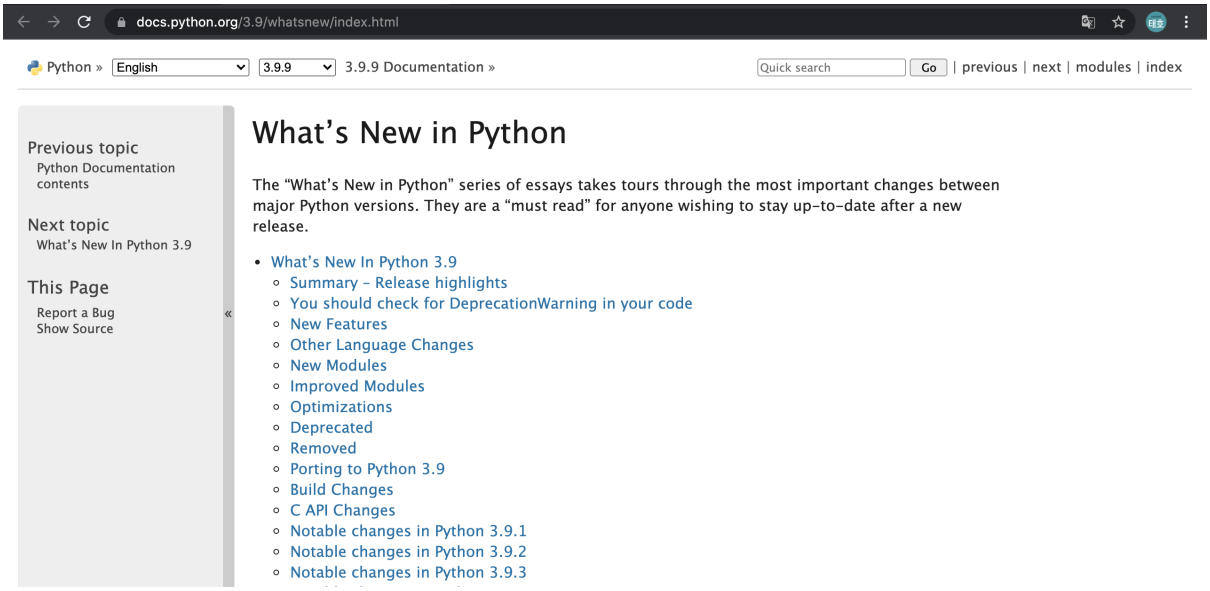
새로운 버전의 인터프리터를 다운받아서 사용해야 한다.

10. 파이썬 버전 새로운 내용 확인하기

영상에서 파이썬 major.minor.micro 버전들과 각 버전들 사이에 어떤 종류의 변화가 있는지에 대해서 알아봤는데요. 그럼 각 버전 사이에 정확히 어떤 변화가 있었는지는 어떻게 확인할 수 있을까요?
물론 가장 간단한 방법은 구글과 같은 검색 엔진에 검색해보는 건데요. 이번 노트에서는 파이썬 공식 홈페이지에서 이걸 확인하는 방법에 대해서 알아보겠습니다.

새로운 기능 (What's New) 페이지

파이썬 공식 웹사이트에는 새로운 기능 (What's New) 라는 페이지가 있는데요. 바로 이곳에서 파이썬 버전 별 바뀌는 내용들을 확인할 수 있습니다. (링크: https://docs.python.org/3.9/whatsnew/index.html)



가장 위를 보면 버전을 고를 수 있게 돼있는데요. 옵션 중 가장 최신 버전 마이너 고르면 됩니다. 가장 높은 버전을 고르게 되면 어짜피 그 아래 버전들에 대한 내용들도 같이 나오게 되니까요, 굳이 안 고를 이유가 없겠죠? 새로운 기능 출시는 마이너 버전에서 이뤄지기 때문에 섹션이 이걸 기준으로 나눠져있긴 한데요. 세부 항목을 확인해보면 마이크로 버전 중에서도 중요한 변화가 있으면, 이걸 확인할 수 있는 링크가 따로 주어지기 때문에 직접 확인하실 수 있습니다.
기본적으로 새로운 기능 페이지는 영어로 제공되긴 하는데요. 2021년 9월 기준 3.5 버전 이후에는 한글 번역본도 제공이 됩니다. 버전을 고르는 곳 바로 옆을 보면 언어를 선택하실 수도 있습니다. 여기서 Korean을 선택하시면 한글 번역본을 볼 수 있습니다.

11. 다양한 파이썬 버전 설치/사용하기

```
python3 --version
# 파이썬 버전확인
```

12. 컴퓨터가 파이썬 인터프리터를 찾는 법

PATH라는 문자열 변수를 사용해서 찾는다.

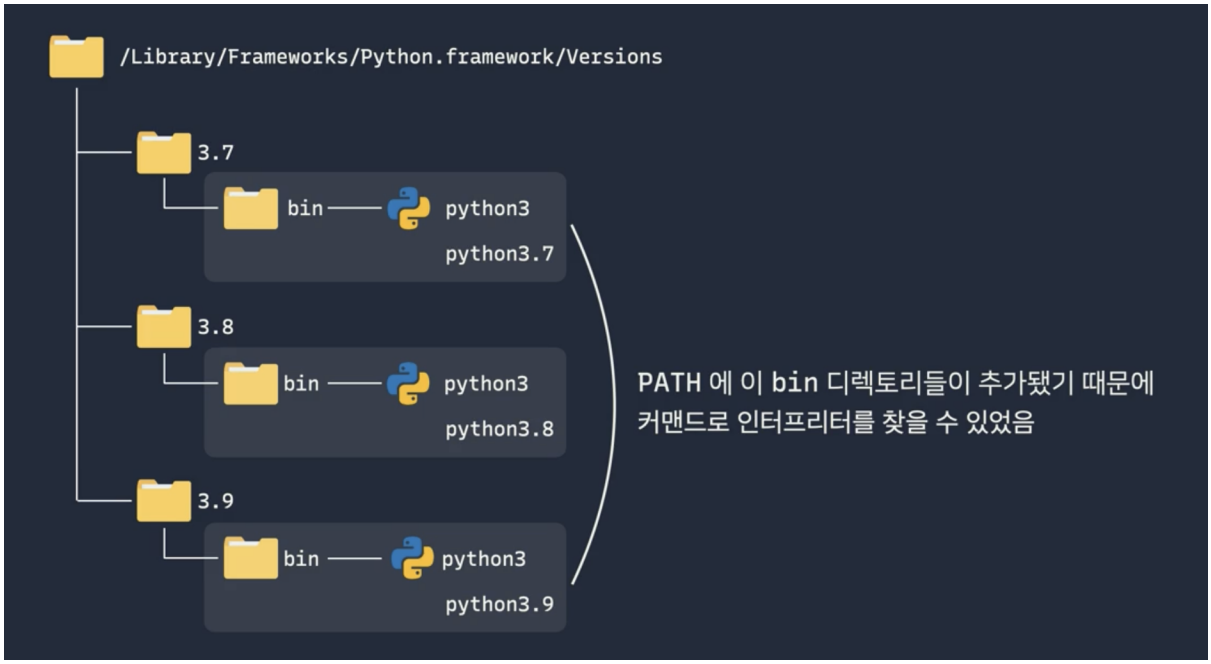
PATH

컴퓨터가 커맨드에 해당하는 프로그램들을 검색하는 경로들

path라는 문자열 변수는 각 경로를 :으로 구별한다.

가장 왼쪽부터 해당 프로그램을 찾는 것이다.

```
echo $PATH
# 이렇게 하면 PATH 목록 나온다.
/Users/akor1/opt/anaconda3/condabin:/Users/akor1/.sdkman/candidates/java/current/bin:/Users/akor1/.rbenv/shims:/Users/akor1/.rbenv/bin
```



13. 기본 파이썬 인터프리터 설정하기

path 변수를 사용하면 된다 컴퓨터가 원하는 파이썬 인터프리터를 먼저 찾도록 나머지 버전들 bin디렉토리보다 앞으로 보내면 된다.


```
subl ~/.zprofile
```

이렇게 실행시키면 새로운 터미널 창이 열릴 때마다 실행되는 파일이 열린다.

```
# Setting PATH for Python 3.9
# The original version is saved in .zprofile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.9/bin:${PATH}"
export PATH
eval "$(/opt/homebrew/bin/brew shellenv)"


# Added by Toolbox App
export PATH="$PATH:/Users/akor1/Library/Application Support/JetBrains/Toolbox/scripts"


# Setting PATH for Python 3.10
# The original version is saved in .zprofile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.10/bin:${PATH}"
export PATH

# Setting PATH for Python 3.11
# The original version is saved in .zprofile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.11/bin:${PATH}"
export PATH
```

PATH 변수는 터미널을 키고 끌때마다 초깃값으로 돌아간다. 그리고 이걸 코드를 사용해서 수정할 수 있다. 이 두가지 사실을 활용해서 터미널이 켜질 때마다 실행하는 파일에 PATH를 설정하는 코드를 추가해 놓은 것이다. 이러면 반 영구적으로 PATH를 관리할 수 있다.
아래 있을 수록 PATH의 앞쪽에 추가 된다. 그래서 이 파일 내용의 순서를 바꿔주면 순서를 바꿀 수 있다.

```
which python3
# 사용하고 싶은 커맨드가 정확히 어떤 경로에 있는 프로그램을 사용하는지 확인하고 싶을때 쓰는 코드
```

15. 파이썬 패키지와 pip

외부 패키지는 다운 받아야지만 사용할 수 있는데 가장 일반적으로 pip이라는 파이썬 패키지 관리 프로그램을 통해서 설치한다.

```
pip3 --version
# pip3 버전 확인
pip3 install --upgrade pip'
# 최신 버전으로 업그레이드
pip3 install numpy
# 이렇게 하면 numpy 패키지 설치할 수 있다.
pip3 install numpy==1.20.1
# 이렇게 버전 정보를 구체화 해서 실행시킬 수도 있다.
pip3 uninstall numpy
# 특정 패키지 삭제
pip3 list
# 현재 환경에서 설치한 모든 파이선 패키지들을 볼 수 있다.
```

16. pip 복습

이번 노트에서는 pip 커맨드들을 정리해보겠습니다. pip은 파이썬3.4 이상 버전에서는 파이썬을 설치할 때 같이 설치되기 때문에 일반적으로 따로 다운받을 필요는 없습니다.

최신 버전으로 업그레이드

`pip3 install --upgrade pip`

pip은 항상 최신 버전으로 업그레이드해서 사용되는 게 권장됩니다. 사용하고 있는 pip이 최신 버전이 아니라면, 경고 메시지가 출력되는데요. 위 커맨드를 실행시켜서 업데이트를 하면 되죠. 경고 메시지에서도 이 커맨드를 사용하라고 출력되기 때문에 굳이 모르셔도 문제는 없습니다.

패키지 다운로드

`pip3 install some_package`

패키지를 다운받을 때는 pip3 install, 그리고 그 뒤에는 패키지 이름을 아규먼트로 넘겨주시면 됩니다.

`pip3 install some_package==x.y.z`

패키지의 최신 버전이 아니라 x.y.z 버전을 다운받고 싶으시면 패키지 이름뒤에 ==를 붙이고 버전을 써주시면 됩니다.

패키지 삭제

`pip3 uninstall some_package`

패키지를 삭제할 때는 pip3 uninstall 그리고 삭제할 패키지 이름을 아규먼트로 넘겨주시면 됩니다.

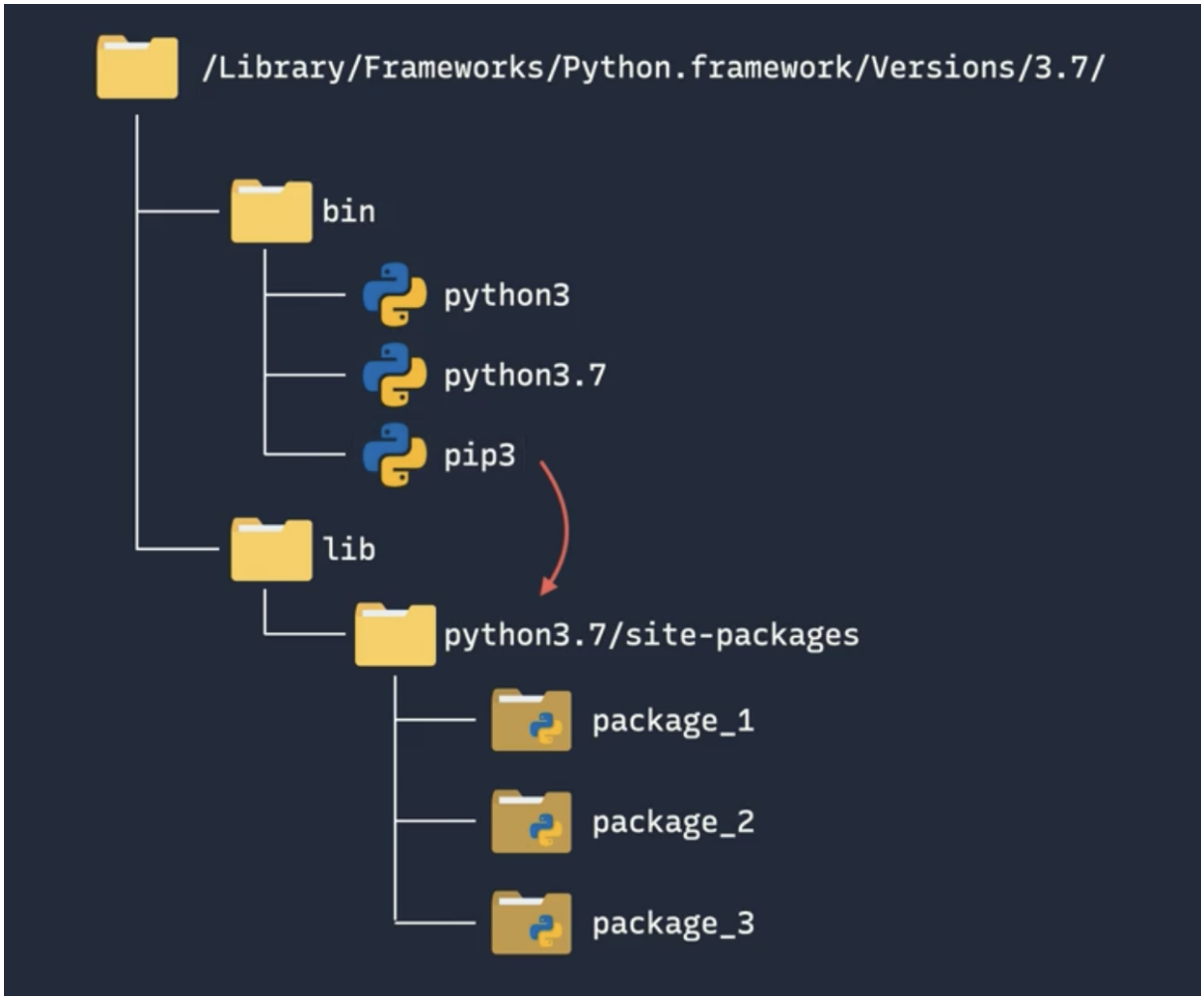
다운로드된 패키지들 확인

`pip3 list`

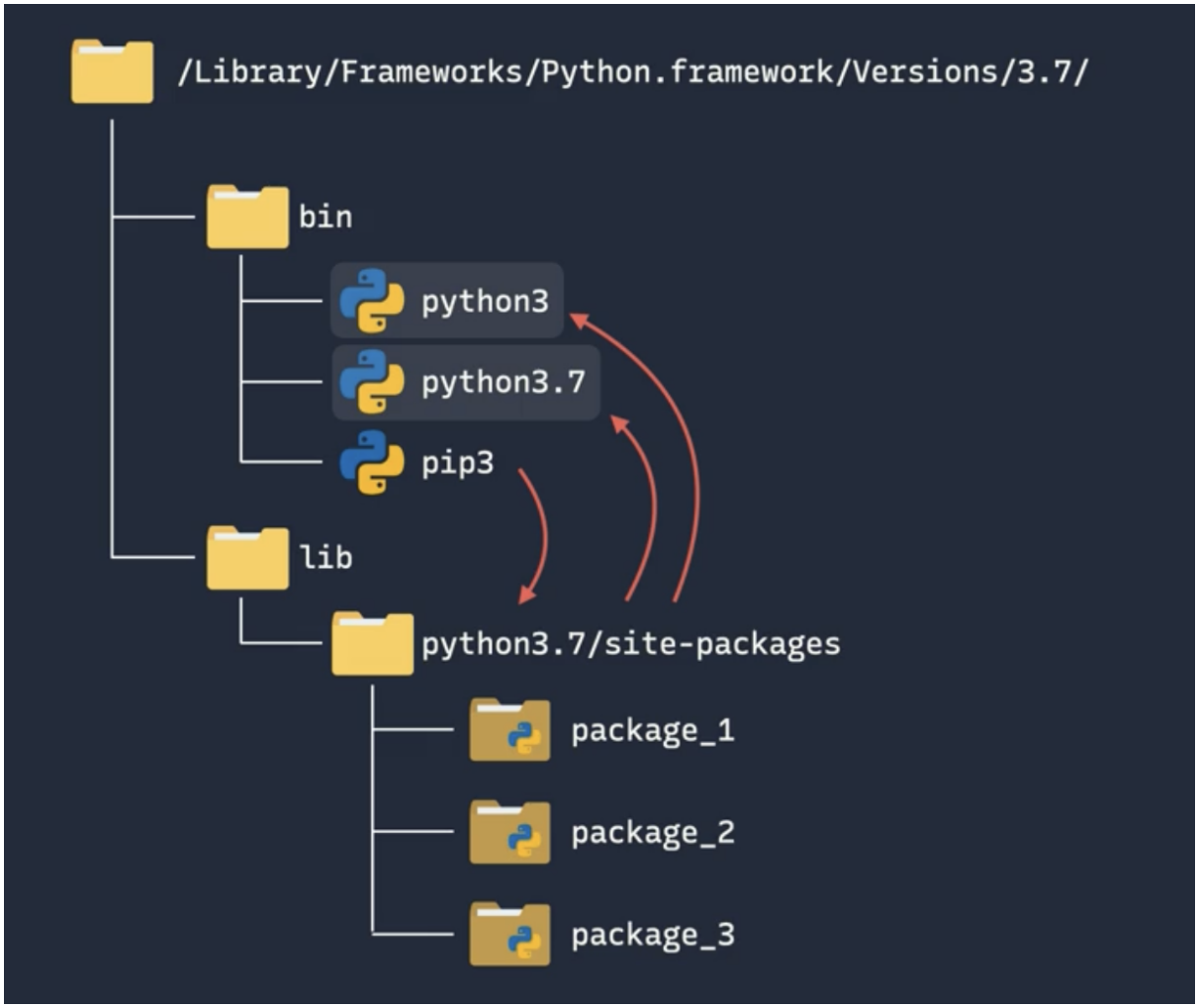
위 커맨드를 사용하면 현재 환경에서 설치한 모든 패키지와 그 버전을 확인할 수 있습니다.

17. 파이썬 인터프리터와 pip

3.7 bin 디렉토리에 저장된 pip3은



여기 site-packages에 다운 받고



같은 디렉토리에 있는 pip3도 외부 패키지를 찾을 때 여기 있는 site-package를 검색하도록 설정이 미리 되어 있기 때문에 python과 pip인 오류를 일으키지 않고 같이 작동할 수 있다.

20. 파이썬 minor 버전 중복 설치

파이썬 공식 홈페이지의 맥 인스톨러를 사용해서 **이미 설치한 파이썬 minor 버전**의 다른 micro 버전을 다운받으면 **새로운 minor 버전을 설치**할 때와는 다르게 작동합니다. 예를 들어 파이썬3.7.5 버전을 설치하고 다시 3.7.11 버전을 설치하는 경우를 얘기하는 거죠.
일단 맥 인스톨러를 사용해서 파이썬을 다운받으면 각 마이너 버전마다 **하나**의 인터프리터만 설치할 수 있습니다. 3.7.5 버전을 쓰고 있다가 3.7.11 버전을 다운받으면 인터프리터 파일만 덮어쓰기 됩니다.
또 다른 점은 PATH 변수에서 버전 순서가 변하지 않습니다. 맥 인스톨러는 새로운 파이썬 버전을 다운받은 후에 PATH에 해당 minor 버전 bin 디렉토리에 대한 경로가 **없으면** 그 경로를 가장 앞쪽에 추가해줍니다. 하지만 이미 있으면 따로 추가해주지 않죠.
예를 들어 파이썬 3.7.5를 처음 다운받았을 때 PATH에 파이썬3.7 인터프리터가 들어있는 경로가 추가됐을텐데요. 그 후에 파이썬 3.7.11을 다운받으면 이미 PATH에 해당 경로가 들어가있기 때문에 다시 추가되지 않습니다. 오히려 같은 경로가 PATH에 다른 순서로 여러번 추가되는 게 이상하겠죠? 새롭게 추가되지 않기 때문에 3.7 디렉토리에 있는 `python3` 커맨드가 제일 앞에 있지 않을 수 있는데요.
그렇기 때문에 이 버전을 기본 파이썬으로 사용하기 위해서는 ~/.zprofile 또는 ~/.bash_profile 파일에서 PATH 변수를 직접 수정해주셔야 합니다.
다행인 점은 인터프리터 자체는 덮어쓰워지더라도, site-packages 등 나머지 디렉토리나 파일들은 그대로 사용되기 때문에 사용하던 패지키들은 다시 설치할 필요 없이 그대로 사용할 수 있다는 점입니다.
전에 말했듯이 사용할 파이썬 버전을 고를 때는 가장 앞선 minor 버전에서 하나 낮은 버전 중 micro 버전이 가장 높은 걸 고릅니다. 최신 버전이 3.7.1 이면 3.6 버전 중에 가장 앞선 걸 고르면 되는 거죠. 그리고 이 minor 버전은 보통 잘 바꾸지 않습니다. 또, 같은 마이너 버전 안에서는 코드가 서로 호환되기 때문에 굳이 여러 버전을 사용할 필요가 없죠.
그렇기 때문에 보통 파이썬 마이너 버전은 하나씩만 다운받아서 사용하는데요. 여러분들도 파이썬 마이너 버전을 하나씩만 설치하고 관리하시면 이런 헷갈리는 상황을 방지할 수 있습니다.