2. 파이썬 가상 환경

☑ 복습	☐
≔ 개념	
☑ 블로그	☐
≡ 페이지	

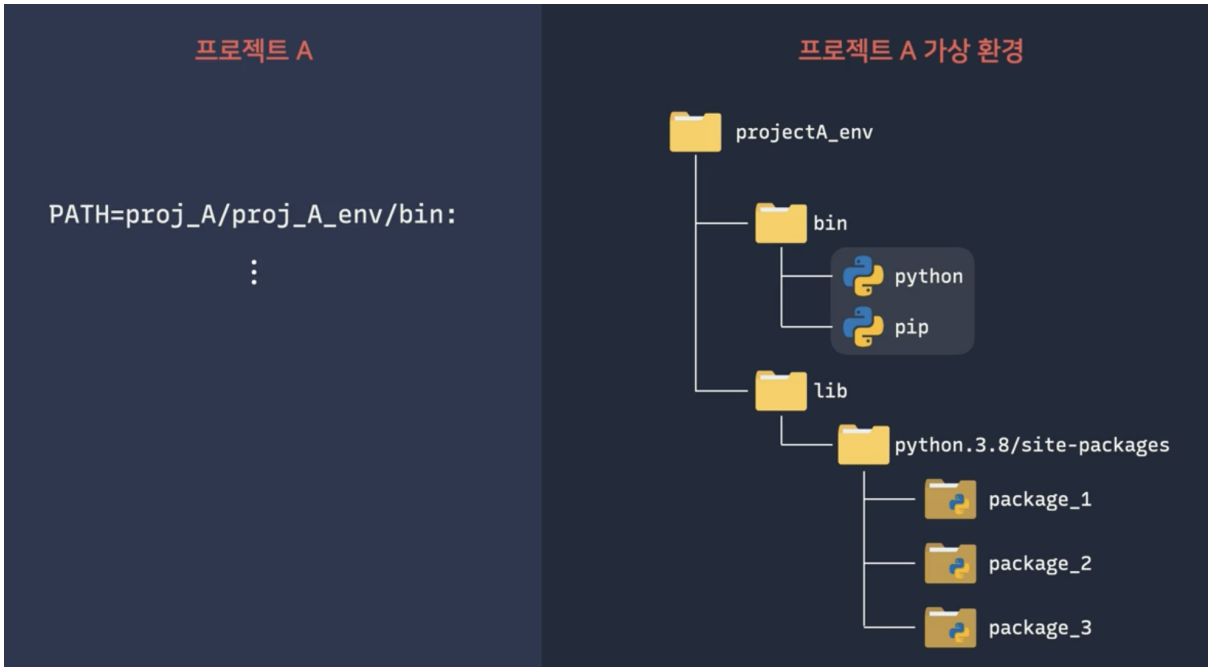
1. 파이썬 가상 환경

가상 환경

공용 또는 기본으로 사용하는 목적의 파이썬 환경이 아니라 프로젝트 단위로 고립시켜서 사용하는 환경을 가상 환경이라고 부른다.

가상 환경이 작동하는 원리는 글로벌 환경과 똑같다. 프로젝트마다 하나의 환경 폴더를 만들고 그 안에 파이썬 인터프리터, pip과 site-packages 디렉토리를 만든다.

대략적으로 이런 형태인데 그 다음에는 일시적으로 PATH 변수를 사용해서 새롭게 만든 환경의 bin 디렉토리를 가장 앞으로 추가시켜 준다.



virtualenv

이 가상 환경 도구를 사용하겠다.

```
pip3 install virtualenv
# 설치 코드
```

5. 가상 환경 커맨드 정리

virtualenv 다운받기

`pip3 install virtualenv`

virtualenv는 쉽게 pip을 통해서 다운받을 수 있습니다. 다운받고 후에는 `virtualenv` 커맨드를 사용할 수 있습니다.

virtualenv로 가상 환경 만들기

`virtualenv --python=python3.7 data_science_env`

virtualenv 커맨드뒤에, `--python` 옵셔널 아규먼트를 통해 사용하고 싶은 파이썬 버전의 인터프리터를, 그리고 아규먼트로 가상 환경의 이름을 넘겨주면 가상 환경을 만들 수 있죠.

조금 헷갈릴 수도 있는데요. 아규먼트로 파이썬 버전 인터프리터를 넘겨주면 그 인터프리터 자체를 사용하는 건 아니고요. 그 인터프리터의 버전을 사용합니다.

이 커맨드에 대해서 팁을 좀 드리면:

`--python` 부분을 그냥 줄여서 `-p` 라고 쓸 수 있고요. 이 옵션 뒤에 넘겨주는 파이썬 인터프리터는 `python3.7` 과 같은 커맨드뿐만 아니라 경로를 넘겨줘도 된다는 점입니다.

예를 들어 그냥 `python3.7`

```
virtualenv -p=/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 data_science_env
```

이렇게도 가상환경을 넘겨줄 수 있는 거죠.

파이썬 인터프리터는 커맨드로 넘겨주던 경로로 넘겨주던 컴퓨터에 이미 다운받아놓은 것들만 사용할 수 있기 때문에 주의해주세요.

가상 환경 활성화시키기

```
source data_science_env/bin/activate
```

`source` 커맨드는 현재 터미널 창에서 주어진 파일을 실행시켜줍니다. 그리고 가상 환경 디렉토리의 bin 폴더 안에 activate라는 파일이 있는데요. 이 파일에 가상 환경을 활성화시켜주는 코드가 저장돼있습니다. 이걸 실행시켜 주면 가상 환경이 활성화되는 거죠.

그냥 전 챕터에서 배운 거랑 똑같이 PATH 변수 가장 앞쪽에 가상 환경의 `bin` 디렉토리를 추가해줍니다.

가상 환경을 활성화시킨 후에는 하나의 파이썬과 pip만 쓴다는 걸 가정할 수 있기 때문에 `python3`, `pip3` 대신 `python` 과 `pip` 커맨드를 사용하시면 됩니다.

가상 환경에서 pip 사용하기

패키지 다운받기

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

가상 환경을 활성화하지 않았을 때와 동일하게 패키지를 설치할 수 있습니다.

패키지 삭제하기

```
pip uninstall some_package
```

가상 환경을 활성화하지 않았을 때와 동일하게 패키지를 삭제할 수 있습니다.

사용하는 패키지 목록 저장하기

```
pip freeze > requirements.txt
```

`pip freeze` 커맨드를 사용하면 `pip list` 와 같이 현재 환경에서 다운받은 모든 패키지와 패키지 버전을 출력할 수 있습니다. 그리고 `>` 를 사용하면 왼쪽 커맨드의 출력 내용을 오른쪽

파일에 저장시킬 수 있는데요. 이때 `requirements.txt` 라는 파일이 없으면 새롭게 만들어집니다.

정리하면 위 커맨드는 requirements.txt라는 파일에 현재 환경에 다운받은 패키지와 패키지 버전을 저장합니다.

패키지 목록에 있는 패키지 모두 다운받기

```
pip install -r requirements.txt
```

`pip install` 커맨드에 옵셔널 아규먼트 `-r` 을 사용하면 바로 뒤에 파일에 나와 있는 모든 패키지를 적혀있는 버전에 맞춰서 다운받을 수 있습니다.

가상 환경 비활성화시키기

```
deactivate
```

위 커맨드를 사용하면 활성화시킨 가상 환경에서 나와 글로벌 환경으로 돌아갈 수 있습니다.

7. direnv로 가상 환경 자동화하기

사실 하나의 컴퓨터에서 여러 파이썬 프로젝트에 동시에 진행하고 있으면 가상 환경을 매번 활성화하고 비활성화하면, 굉장히 귀찮을 뿐만 아니라 자잘한 실수로 이어질 수 있습니다.

이번 노트에서는 맥 운영체제에서 가상 환경을 직접 키고 끄지 않아도 되게 자동화하는 방법을 살펴보겠습니다. 파이썬 설치나 가상 환경 관리와 같이 가상 환경 자동화도 굉장히 다양한 도구들을 통해서 할 수 있는데요. 저희는 direnv라는 도구를 사용하는 법을 보겠습니다. 이번 노트 내용은 파이썬 가상 환경에서 필수로 알아야 되는 내용은 아니니까요. 관심 있으신 분들만 참고하시면 됩니다.

direnv를 사용하면 특정 디렉토리와 그 하위 디렉토리 안에서만 파이썬 환경이 활성화되고, 그 디렉토리들을 벗어나면 비활성화되게 설정 할 수 있습니다.

설치

먼저 direnv를 설치하는 방법에 대해서 알아보겠습니다.

맥의 패키지 관리자, homebrew를 이용하면 간단하게 설치 할 수 있습니다. (homebrew 설치 방법은 여기를 참고해주세요!)

```
brew install direnv
```

조금 기다리면 direnv가 설치가 되는 걸 확인할 수 있습니다.

그 다음에는 direnv를 시스템 쉘과 연결시켜줘야 하는데요.

ZSH를 사용하고 있으면

`subl ~/.zshrc`를 통해서 zshrc 파일을 열고

`eval "$(direnv hook zsh)"`를 파일 끝에 추가해주세요. 저장하고 닫는 걸 잊지 마세요!

BASH를 사용하고 있으면

`subl ~/.bashrc`를 통해서 bashrc 파일을 열고

`eval "$(direnv hook bash)"` 를 파일 끝에 추가해주세요. 저장하고 닫는 걸 잊지 마세요!

사용

자 다음에는 가상의 프로젝트 폴더로 이동합시다. 지금 파일이 이런 구조로 돼있습니다. `data_science_proj` 라는 프로젝트 파일이 있고, 그 안에 `data_science_env` 라는 가상 환경 디렉토리가 저장돼있죠. 지금처럼 `virtualenv` 커맨드를 사용해서 미리 사용하는 가상 환경 파일을 만들어놓으세요.

data_science_proj

|— src

|— data

|— data_science_env

`touch .envrc` 를 사용해서 이 디렉토리에 `.envrc` 라는 파일을 만들겠습니다.

그리고 이 파일을 열고 `subl .envrc` 안에 이렇게 써줄게요.

```
source ./data_science_env/bin/activate
echo "virtualenv changed to   `which python`"
unset PS1
```

제일 윗줄을 보면 그냥 이 프로젝트의 가상 환경을 활성화해주는 코드죠? 그 밑에는 어떤 파이썬 인터프리터를 쓰는지 출력해주는 코드고요. 가장 밑에 줄은 일단 이해하기는 어렵겠지만 direnv와 파이썬 가상 환경을 사용할 때는 추가해줘야 되는 코드니까 그냥 넘어갈게요.

direnv는 특정 디렉토리에 들어갔을 때, 그 안에, 또는 그 상위 디렉토리들에 있는 .envrc를 자동으로 실행시켜주는 프로그램입니다. 그러니까 프로젝트 디렉토리에 .envrc라는 파일을 만들고, 그 안에 가상 환경을 활성화시키는 코드를 쓰면, data_science_env 또는 그 하위 디렉토리로 들어가도 위에 있는 코드가 실행되는 거죠. 신기한 건 data_science_env 밖에 있는 디렉토리로 가면 위 코드 내용이 무효화되고, 가상 환경이 비활성화됩니다.

한 번 해볼까요?

상위 디렉토리로 간 다음에 다시 data_science_env 디렉토리로 돌아오면, 무슨 에러 메세지가 뜨는데요?

```
direnv: error /Users/taehosung/Desktop/codeit/python_tests/.envrc is blocked. Run `dir
env allow` to approve its content
```

.envrc 파일이 막혔다고 나옵니다. .envrc 파일은 새롭게 만들거나 수정할 때마다 해당 파일을 사용하겠다고 직접적으로 알려줘야지만 사용할 수 있습니다. 에러 메세지에 나온 것처럼 그냥 `direnv allow` 를 실행시키겠습니다.

그럼 .envrc 파일 안 내용이 실행되고 파이썬 가상 환경이 적용됩니다. 현재 사용하고 있는 파이썬 가상 환경이 출력되는 걸 확인할 수 있죠?

`which python`, `which pip` 을 해봐도 가상 환경 안에 있는 인터프리터와 pip을 사용하고 있는 걸 확인할 수 있습니다.

하지만 일반적으로 가상 환경을 사용할 때와는 다르게 프롬트 앞에 가상 환경 이름이 안 나올텐데요. 조금 이상하죠? direnv를 사용하면 자체적으로 프롬트를 바꿀 수 없습니다. 이걸 개선하고 싶으신 분들은 아래 내용대로 따라해보세요.

ZSH를 사용하고 있으면

`subl ~/.zshrc`를 통해서 zshrc 파일을 열고,

```
setopt PROMPT_SUBST

show_virtual_env() {
  if [[ -n "$VIRTUAL_ENV" && -n "$DIRENV_DIR" ]]; then
    echo "($(basename $VIRTUAL_ENV))"
  fi
}
PS1='$(show_virtual_env)'$PS1
```

이 내용을 추가해주세요.

BASH를 사용하고 있으면

`subl ~/.bashrc`를 통해서 bashrc 파일을 열고,

```
show_virtual_env() {
  if [[ -n "$VIRTUAL_ENV" && -n "$DIRENV_DIR" ]]; then
    echo "($(basename $VIRTUAL_ENV))"
  fi
}
export -f show_virtual_env
PS1='$(show_virtual_env)'$PS1
```

이 코드를 추가해주세요.

추가 이후

그리고 터미널을 끈 후 다시 키겠습니다.

다시 data_science_proj 파일로 들어가면, 가상 환경이 적용되고, 프롬트 앞에 이름도 앞에 잘 나오는 걸 확인할 수 있습니다.

실험해보시면 아시겠지만, data_science_proj의 아무 하위 폴더나 들어가도 자동으로 이 가상 환경이 활성화되고, 밖으로 나가면 가상 환경이 비활성화됩니다.

이제 더 이상 이 프로젝트 관련 일을 할 때 가상 환경을 직접 적용/해제하지 않으셔도 되는 거죠. 편리하죠?