2025-04-02(Wed.)

- The Python Tutorial (v.3.13)
- 파이썬 자습서 (v.3.13)
- yeonsookim-wt/lgtm
- 점프 투 파이썬
- ⭐ 파이썬 독학하기 좋은 위키독스 모음집

11-environment-and-packages

venv ── 가상 환경 생성 도구

파이썬 가상환경 만들기

- ⭐ ⭐ pyenv를 이용하여 여러 개의 Python 버전 관리하기(for. MacOS M1)
  1. 빌드 종속성 설치
     - $ brew install openssl readline sqlite3 xz zlib
  2. pyenv, pyenv-virtualenv 설치(Mac/M1)
     - $ brew install pyenv
     - $ brew install pyenv-virtualenv
  3. zshrc 설정

```
# pyenv/bin 경로를 PATH 에 등록하여 pyenv를 어디에서도 사용 가능하도록 하기 위해 추가한다.
$ echo 'export PATH="$HOME/.pyenv/bin:$PATH"' >> ~/.zshrc

# pyenv 를 정상적으로 사용할 수 있도록 zshrc 에 init 명령어를 추가한다.
$ echo 'eval "$(pyenv init -)"' >> ~/.zshrc

# pyenv virtualenv 를 정상적으로 사용할 수 있도록 zshrc 에 init 명령어를 추가한다.
$ echo 'eval "$(pyenv virtualenv-init -)"' >> ~/.zshrc

# 활성화
source ~/.zshrc
```

1. python -m venv {venv(가상환경명)}

   - ⭐ ⭐ 파이썬 venv 가상환경 구축(+파이썬 버전 지정)
   - ⭐ mac venv 가상환경
   - ⭐ [Python] 가상환경 만들기 on Mac (venv, virtualenv)
     - Case1. venv package 사용
       - 가상환경을 만들 파일 디렉토리로 이동
         - $ cd {your directory}
       - 가상환경 만들기
         - $ python3 -m venv ./{your venv name}

        - 가상환경 활성화
            - $ source {your venv name}/bin/activate

2. conda create -n venv
    - conda active venv
    - conda deactive
3. ⭐ uv - An extremely fast Python package and project manager, written in Rust.
    1. uv 설치
    2. uv init 프로젝트명
    3. cd 프로젝트명
    4. uv venv venv
    5. source venv/bin/activate <-- Activate
    6. code . <-- visual code로 열기
        1. Open Visual Studio Code
        2. Open the command pallette with Command + Shift + P (or F1)
        3. Type Shell in command palette
        4. Select Shell Command: Install code in PATH from suggested list ```
    - Install uv on macOS with MacPorts
    - ⭐ uv Commands
    - ⭐ python의 uv 사용법
        - 초기화 (example 프로젝트)
            - uv init exampleProj
        - 의존성 추가
            - uv add ruff
        - 특정 패키지를 설치
    - ⭐ Python environments
        - uv venv my-name
        - uv venv --python 3.11
        - uv add requests

환경 저장과 재현 ── requirements 파일 활용

requirements 파일에 현재 환경 정보 저장

- ⭐ [Python] pip 패키지 목록 requirements.txt 생성 및 설치 방법

12-unittest

- python unittest
- unittest — Unit testing framework
- https://www.daleseo.com/python-unittest-testcase/
- ⭐ [파이썬] 단위 테스트의 기본 (unittest)
- ⭐ 단위테스트(Unit test) 개념과 python에서 사용하기
    1. requirements.txt 생성 명령어
        1. pip freeze 명령어
            - pip freeze > requirements.txt

2. pip list 명령어
    - pip list --format=freeze > requirements.txt
2. requirements.txt 설치 명령어
    1. pip install 명령어
        - pip install -r requirements.txt

13-application

환경설정

> 1. cd workspace
> 2. cd python
> 3. mkdir lgtm
> 4. cd lgtm
> 5. python -m venv venv
> 6. quit vs-code
> 7. cd lgtm
> 8. code .
> 9. create main.py file
> 10. pip install requests (terminal)

```
── Workspace
│   ├── lgtm
│   └── python
│       └── lgtm
│           ├── main.py
│           └── venv
│               ├── bin
│               └── pyvenv.cfg
```

{'args': {'q': 'python'}, 'headers': {'Accept': '/', 'Accept-Encoding': 'gzip, deflate', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.32.3', 'X-Amzn-Trace-Id': 'Root=1-67ecb5b1-0c80c53627a150984043a5d6'}, 'origin': '112.221.198.147', 'url': 'https://httpbin.org/get?q=python'}

Click ── 커맨드 라인 도구 작성 라이브러리

```
# Click 설치
%pip install Click==7.0
```

greet.py

```
import click

@click.command()
```

```
@click.option('--words', default='Hello')
@click.argument('name')
def greet(name, words):
    click.echo(f'{words}, {name}!')

if __name__ == '__main__':
    greet()
```

```
!python3 greet.py user
```

Hello, user!

Pillow —— 이미지 처리 라이브러리

```
# Pillow 설치
%pip install Pillow==6.2.1
```

```
import os
from PIL import Image
def thumbnail(infile, size=(128, 128)):
    outfile = os.path.splitext(
        infile)[0] + ".thumbnail"
    try:
        im = Image.open(infile)
        im.thumbnail(size)
        im.save(outfile, "JPEG")
    except IOError:
        print("cannot create thumbnail for", infile)
```

프로젝트 작성

Git 이용

1. !mkdir workspace
2. cd workspace
3. git init

.gitignore 파일 작성

```
# 아래 URL의 내용을 .gitignore라는 이름의 파일에 저장
# 환경에 따라 키워드를 바꾸어 내용을 편집함
# https://www.gitignore.io/api/macos,python
!git add .
```

지속적인 통합 도입

CircleCI에서의 테스트 자동화

- 1. CI / CD 개념
- [개발방법] 애자일, CI/CD, TDD
- 애자일 ci cd

    1. CI(Continous Integration): 지속적인 통합이란 의미로 어플리케이션의 새로운 코드 변경 사항이 정기적으로 빌드 및 테스트 되어 통합하는 것
    2. CD(Continous Delivery, Continous Deployment)
        - Continuous Delivery: 공유 레포지토리로 자동으로 Release 하는 것
        - Continuous Deployment: Production 레벨까지 자동으로 deploy 하는 것을 의미

- CI/CD란 무엇일까? (Continuous Integration/Continuous Delivery)

    - Continuous Integration(CI)란 지속적인 통합이라는 의미

        - 어플리케이션의 새로운 코드 변경 사항이 정기적으로 빌드 및 테스팅 되어 공유 레포에 통합되는 것을 의미
            - CI의 간단한 순서 정리1
                1. 개발자가 구현한 코드를 기존 코드와 병합
                2. 병합된 코드가 올바르게 동작하고 빌드되는지 검증
                3. 테스트 결과 문제가 있다면 수정하고 다시 1로 돌아간다. 문제가 없다면 배포를 진행

    - CD는 CI의 연장선이며 소프트웨어가 항상 신뢰 가능한 수준에서 배포될 수 있도록 지속적으로 관리하자는 의미

        - Continuous Delivery (지속적 배포) : 지속적 제공은 CI를 통해서 새로운 소스코드의 빌드와 테스트 병합까지 성공적으로 진행되었다면, 빌드와 테스트를 거쳐 github과 같은 저장소에 업로드하는 것을 의미
        - Continuous Deplolyment(지속적 배포) : 지속적 배포는 이렇게 성공적으로 병합된 내역을 저장소뿐만 아니라 사용자가 사용할 수 있는 배포환경까지 릴리즈하는 것을 의미

git

- git-cheat-sheet-education