



EDDI

Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv1 과정

제 4기

2023. 02. 03

진동민

학습목표 & 14회차 날짜

학습목표

- flask를 사용하여 간단한 서버를 만들고 실행할 수 있다.
- pymysql 모듈을 사용하여 MySQL을 사용할 수 있다.
- 포트가 무엇인지 이해할 수 있다.

수업 날짜

2022-12-17 (토) 오후 6시~9시

목차

- 1) 웹 개발 환경 구축
- 2) vue 프론트엔드 서버 실행
- 3) flask 백엔드 서버 실행
- 4) MySQL 환경 구축
- 5) MySQL 실습
- 6) 참고 문서
- 7) 수업내용 사진

웹 개발 환경 구축

이번 수업의 실습에 필요한 flask와 pymysql 모듈을 사용하기 위해 강사님이 주피터 노트북으로 env_setting.ipynb 파일을 작성하셨다. 파일은 아래와 같다.

```
In [ ]: from flask import Flask

In [ ]: # 잘 나온다면 문제 없음
        # 잘 나오지 않는다면 Flask 모듈이 설치되어야함

In [ ]: from flask import redirect, url_for, request, jsonify

In [ ]: from flask_cors import CORS, cross_origin

In [ ]: # flask_cors 에러가 발생하는 경우
        !pip install flask_cors

In [ ]: # 웹/앱 개발시 활용할 MYSQL DB
        !pip install pymysql

In [ ]: import pymysql

In [ ]: import pickle

In [ ]: import numpy as np
```

가지고 있는 데스크탑의 경우에는 4, 7번째 셀을 실행하면 모듈이 설치되어 있지 않아 ModuleNotFoundError가 발생하였다.

해결 방법

5, 6번째 셀을 실행하여 모듈을 설치한다.

오류가 발생한 셀을 다시 실행해서 오류가 발생하지 않는 것을 확인한다.

오류가 발생하지 않으면 이제 vue 환경설정하러 가면 된다!
다음 장으로...

(참고)

주피터 노트북을 사용하여 개발 환경 설정을 해보니 환경 설정이 되어있는지 바로 확인할 수 있었다.
나중에 프로젝트 팀장을 맡게되고 팀원들의 개발 환경을 설정할 경우가 있다면 강사님이 사용하신 이 방법을 써야겠다.

웹 개발 환경 구축

이번에 설치할 것은 vue이다.

설치 방법은 README.md 파일에 적혀있다.
파일은 [EmbeddedMasterLv1/documents/vue](#) 에 있다.

How to install Vue on Ubuntu 20.04

```
sudo apt-get update  
sudo apt-get upgrade
```

```
curl -sL https://deb.nodesource.com/setup_16.x | sudo bash -  
sudo apt-get install nodejs
```

이후 아래 명령을 통해 nodejs가 잘 설치 되었는지 확인합니다.

```
node -v
```

그리고 최신 npm으로 업그레이드 합니다.

```
npm install npm@latest -g  
npm -v
```

vue-cli를 설치합니다.

```
sudo npm install -g @vue/cli  
vue --version
```

vue 프로젝트를 생성합니다.

```
vue create vue-project
```

오류가 발생할 수도 있는데 오류가 발생한 명령어 맨 앞에 `sudo`를 붙여서 다시 실행하면 해결할 수 있다.

```
(base) try@try-desktop:~$ npm install npm@latest -g
npm ERR! code EACCES
npm ERR! syscall rename
npm ERR! path /usr/lib/node_modules/npm
npm ERR! dest /usr/lib/node_modules/.npm-QUIFSSiV
npm ERR! errno -13
npm ERR! Error: EACCES: permission denied, rename '/usr/lib/node_modules/npm' -> '/usr/lib/node_modules/.npm-QUIFSSiV'
npm ERR! [Error: EACCES: permission denied, rename '/usr/lib/node_modules/npm' -> '/usr/lib/node_modules/.npm-QUIFSSiV'] {
npm ERR!   errno: -13,
npm ERR!   code: 'EACCES',
npm ERR!   syscall: 'rename',
npm ERR!   path: '/usr/lib/node_modules/npm',
npm ERR!   dest: '/usr/lib/node_modules/.npm-QUIFSSiV'
npm ERR! }
npm ERR! The operation was rejected by your operating system.
npm ERR! It is likely you do not have the permissions to access this file as the current user
npm ERR! If you believe this might be a permissions issue, please double-check the
npm ERR! permissions of the file and its containing directories, or try running
npm ERR! the command again as root/Administrator.

npm ERR! A complete log of this run can be found in:
npm ERR! /home/try/.npm/_logs/2022-12-29T15_24_38_117Z-debug-0.log
(base) try@try-desktop:~$ sudo npm install npm@latest -g

removed 14 packages, changed 75 packages, and audited 232 packages in 2s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) try@try-desktop:~$ npm -v
9.2.0
```

웹 개발 환경 구축

환경 설정이 끝나면 잘 설치가 되었는지 버전을 확인해 본다. 현재 설치된 버전은 다음과 같다.

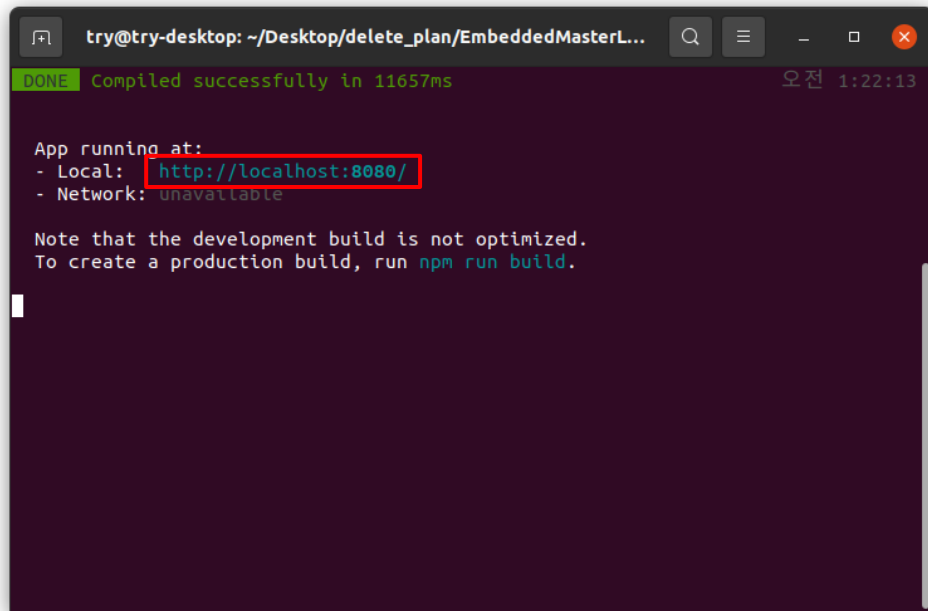
```
(base) try@try-desktop:~$ node -v  
v16.19.0  
(base) try@try-desktop:~$ npm -v  
9.2.0  
(base) try@try-desktop:~$ vue --version  
@vue/cli 5.0.8  
(base) try@try-desktop:~$
```

vue 프론트엔드 서버 실행

vue_frontend 디렉토리에 들어가서 터미널을 실행한다.

다음 명령어를 실행한다.

1. npm install (시간이 오래 걸릴 수 있음)
2. npm run serve (프론트엔드 서버 구동)



```
try@try-desktop: ~/Desktop/delete_plan/EmbeddedMasterL...
DONE Compiled successfully in 11657ms 오전 1:22:13

App running at:
- Local: http://localhost:8080/
- Network: unavailable

Note that the development build is not optimized.
To create a production build, run npm run build.
```

서버가 정상 구동되면 왼쪽과 같이 표시될 것이다.

빨간 박스안에 있는 링크를 클릭하여 열어보자

vue 프론트엔드 서버 실행

링크를 타고 들어가면 웹 브라우저에서 다음을 볼 수 있다.



flask 백엔드 서버 실행

주피터노트북에서 flask_backend 디렉토리에 들어가서, python_snippets.ipynb 파일을 클릭하여 노트북을 연다.

```
In [*]: from flask import Flask
        app = Flask(__name__)

        @app.route("/")
        def hello():
            return "Hello World!"

        if __name__ == "__main__":
            app.run()
```

- * Serving Flask app "__main__" (lazy loading)
- * Environment: production
- WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
- * Debug mode: off

* Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)

왼쪽의 코드는 파이썬으로
가장 간단한 웹 서버를
만드는 방법이다.

빨간 박스내에 있는 링크는
파이썬 서버가 돌고있는
주소이다.

Hello World!

첫 번째 예제

실행 결과

(용어 참고)

snippet: 스니펫은 재사용 가능한 소스 코드, 기계어, 텍스트의 작은 부분을 일컫는 프로그래밍 용어이다.

flask 백엔드 서버 실행

첫 번째 예제코드 분석

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

app = Flask(__name__)

- Flask가 파이썬 서버이다.

@app.route("/")

- route 데코레이터를 사용해서 Flask에게 어떤 URL이 우리가 작성한 함수를 실행시켜야 할 지 알려준다.
- 여기서는 /라는 URL로 들어오면 hello 함수가 동작한다.
- 반환하는 위치는 우리가 보고 있는 웹 브라우저 창으로 반환한다.

flask 백엔드 서버 실행

(의문) 함수 말고 람다로도 가능할까?

강사님의 답변

- 람다는 익명 객체의 느낌이 강한데, 한 번 사용하고 버릴것이라 어떤 요청이 들어왔을 때 그 쪽으로 고정시켜야 되는 거에는 문법적으로 된다고 하더라도 람다가 적합하지 않을 것 같다.

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route("/")
```

```
lambda: "Hello World"
```

```
if __name__ == "__main__":
```

```
    app.run()
```

```
File "/tmp/ipykernel_25767/3065661478.py", line 6
```

```
    (lambda: "Hello World")
```

```
    ^
```

```
SyntaxError: invalid syntax
```

← 궁금해서 람다로 바꿔서 실행해보니 오류가 난다.

flask 백엔드 서버 실행

보통 Web Server는 구성이 다음과 같다.

- Controller: 사용자 URL 요청을 처리한다

첫 번째 예제에서 Controller는 `@app.route("/")` 이다.

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

flask 백엔드 서버 실행

두 번째 예제코드 실행

```
from flask import Flask

app = Flask(__name__)

@app.route('/flask')
def hello_flask():
    return 'Hello Flask'

@app.route('/python/')
def hello_python():
    return 'Hello Python'

@app.route('/')
def flask_home():
    return '가즈아'

if __name__ == '__main__':
    app.run()
```

Hello Flask

/flask

Hello Python

/python 또는 /python/

가즈아

/

flask 백엔드 서버 실행

세 번째 예제코드 실행

```
from flask import Flask

app = Flask(__name__)

@app.route('/hello/<name>') ← 가변 인자를 사용하는 방법
def hello_name(name):
    return 'Hello %s!' % name

if __name__ == '__main__':
    app.run()
```

가변 인자가 필요한 예

- “~~~ 고객님, 상품이 정상 결제되었습니다.”의
메시지에서 고객 이름과 같은 가변적인 정보 처리가
필요한 경우

실행 결과

Hello test!

/hello/test

Hello hi!

/hello/hi

Hello dongmin!

/hello/dongmin

flask 백엔드 서버 실행

네 번째 예제코드 실행

```
from flask import Flask, redirect, url_for, request
app = Flask(__name__)

@app.route('/success')
def success():
    return 'welcome !!!'

@app.route('/login', methods = ['POST', 'GET'])
def login():
    if request.method == 'POST':
        return redirect(url_for('success'))
    else:
        return redirect(url_for('success'))

if __name__ == '__main__':
    app.run()
```

실행 결과

welcome !!!

/login으로 접속 시, /success로 이동

웹에서 데이터를 요청하는 가장 큰 두 가지 방식

- GET 방식: URL에 입력하는 정보
- POST: 정보를 json 형식으로 숨김

아이디와 비밀번호를 입력한 것을 GET 방식으로 처리하면, URL에 정보가 다 노출이된다.

옛날 사이트들, 관공서 사이트들 들어가보면 이 GET 방식을 쓰는데가 아직도 있다고 한다. 그래서 쓰면 바로 비밀번호를 바꾸라고 말하셨다 ㅋㅋ

우리는 URL을 통해서만 요청을 하고 있으니까 메서드는 무조건 GET으로만 들어온다.

네 번째 예제에서 사용된 api 설명

`url_for()`: 문자열을 URL 형식으로 생성한다.

`redirect()`: 인자로 받은 URL로 리다이렉트한다.

코드 분석

- `redirect(url_for("success"))`: succes라는 URL로 이동시킨다.

그래서 네 번째 예제의 login에 접속하면 success로 이동한 것이다.

flask 백엔드 서버 실행

다섯 번째 예제코드 실행

```
from flask import Flask, url_for, request, jsonify
from flask_cors import CORS, cross_origin

import json

app = Flask(__name__)
CORS(app)

@app.route('/vueMember/register', methods = ['POST'])
def register():
    # 낮은 버전의 파이썬
    params = json.loads(request.get_data(), encoding='utf-8') ①
    # 최신 버전의 파이썬 3.9
    #params = json.loads(request.get_data()) ②
    print("params: ", params)

    if len(params) == 0:
        return jsonify("No Parameter")

    for key in params.keys():
        print("params[key]: ", params[key], " key: ", key)

    send_string = params.get('id') + " 님 가입이 완료되었습니다!"
    print(send_string)

    return jsonify(send_string)

if __name__ == '__main__':
    app.run()
```

이 예제를 사용하려면 vue 예제가 실행 중이어야 한다.
(9번 슬라이드를 참고하라)

이 실습 환경의 경우엔 파이썬 버전이 3.9.13 인데, 왼쪽
예제를 실행하면 **오류가 발생**한다.

해결 방법은 ①번 줄을 주석처리하고, ②번 줄을 주석해제
한다.

flask 백엔드 서버 실행

다섯 번째 예제코드 실행

vue 프론트엔드 서버를 실행하고 들어가면 다음과 같은 화면이 나온다.

Home **Vuetify Member Register Test** Login Test Jpa Board List VuetifyMemberJoinColumnTest

1) 여기서 'Vuetify Member Register Test'를 클릭한다.

회원 가입 양식

아이디 test

비밀번호

등록 [최소](#)

2) 아이디와 비밀번호를 입력하고 등록 버튼을 클릭한다. (아이디와 비밀번호는 test로 했습니다)

flask 백엔드 서버 실행

다섯 번째 예제코드 실행

localhost:8080 내용:
등록 성공! - test 님 가입이 완료되었습니다!

확인

3) 웹 브라우저에서 위와 같은 메시지가 표시되면 정상적으로 실행된 것이다.

```
* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
```

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

```
127.0.0.1 - - [30/Dec/2022 19:38:03] "OPTIONS /vueMember/register HTTP/1.1" 200 -
127.0.0.1 - - [30/Dec/2022 19:38:03] "POST /vueMember/register HTTP/1.1" 200 -
```

← 서버 로그

```
params: {'id': 'test', 'pw': 'test'}
params[key]: test key: id
params[key]: test key: pw
test 님 가입이 완료되었습니다!
```

← print 함수를 이용한 출력문

4) 백엔드 서버에서는 로그와 출력문을 확인할 수 있다.

flask 백엔드 서버 실행

다섯 번째 예제코드 설명

버튼을 클릭하여 vue 프론트엔드 서버로 실행시킨 홈페이지에서 아이디와 비밀번호를 flask 백엔드 서버로 보내면서 요청했다.

vue 코드는 다음 파일에 있다.

vue_frontend/src/views/jpaMember/VuetifyMemberRegisterPage.vue

GitHub 링크:

https://github.com/TryTwoTop/EmbeddedMasterLv1/blob/main/47/LeeSangHoon/python/vue_frontend/src/views/jpaMember/VuetifyMemberRegisterPage.vue

```
22     methods: {  
23       onSubmit (payload) {  
24         const { id, pw } = payload  
25         axios.post('http://localhost:5000/vueMember/register', { id, pw })  
26           .then(res => {  
27             alert('등록 성공! - ' + res.data)  
28           })  
29           .catch(res => {  
30             alert(res.response.data.message)  
31           })  
22     }  
23   }  
24 }  
25 }  
26 }  
27 }  
28 }  
29 }  
30 }  
31 }
```

← POST 방식으로 요청

핵심 코드

다섯 번째 예제에서 사용된 api와 코드 분석

params = json.loads(request.get_data())

- 요청이 들어가면 데이터는 request에 들어간다.
- POST 방식은 json 형식으로 보내므로, 들어온 데이터를 json 방식으로 해석해야 한다.
- json 형식으로 읽기 위해 json.loads 메소드를 사용했다.

(참고)

- request.get_data()의 type을 확인해보니 bytes 객체이다.

return jsonify(send_string)

- jsonify 함수를 사용하여 json 형식으로 만들고 정보를 요청한 곳으로 보내고 있다. (여기서는 Vue 프론트엔드 서버로 날라간다)

다섯 번째 예제를 개인 프로젝트에 사용하기 위해서 개조가 필요하다면...

- route 데코레이터에 있는 URL을 변경하면 안된다. (만약 변경하면 Vue에 있는 코드 또한 변경해야한다)
- 그러므로 백엔드 서버의 함수 코드를 바꿔야 한다.

flask 백엔드 서버 실행

다섯 번째 예제 동작 분석

회원 가입 양식

아이디 test

비밀번호

등록 [취소](#)

①

```
22 methods: {
23   onSubmit (payload) {
24     const { id, pw } = payload
25     axios.post('http://localhost:5000/vueMember/register', { id, pw })
26     .then(res => {
27       alert('등록 성공! - ' + res.data)
28     })
29     .catch(res => {
30       alert(res.response.data.message)
31     })
  }
```

Vue 프론트엔드

②
요청

```
@app.route('/vueMember/register', methods = ['POST'])
def register():
    # 낮은 버전의 파이썬
    # params = json.loads(request.get_data(), encoding='utf-8')
    # 최신 버전의 파이썬 3.9
    params = json.loads(request.get_data())
    print("params: ", params)

    if len(params) == 0:
        return jsonify("No Parameter")

    for key in params.keys():
        print("params[key]: ", params[key], " key: ", key)

    send_string = params.get('id') + " 님 가입이 완료되었습니다!"
    print(send_string)

    return jsonify(send_string)
```

④
응답

⑤

localhost:8080 내용:
등록 성공! - test 님 가입이 완료되었습니다!

확인

Flask 백엔드 서버

MySQL 환경 구축

이번에 설치할 것은 MySQL이다.

```
## 이번에는 DB에 데이터를 넣어보자!!!
```

```
## 먼저 Ubuntu에 MySQL을 설치한다.
```

```
# 소프트웨어 패키지 업데이트
```

```
!sudo apt-get update
```

```
# 설치
```

```
!sudo apt-get install mysql-server
```

```
# 방화벽 해제
```

```
!sudo ufw allow mysql
```

```
# MySQL 서비스 재시작(스펙이 떨어지는 경우 권장하지 않음)
```

```
!sudo systemctl enable mysql
```

```
# 파이썬 패키지 설치
```

```
!pip install pymysql
```

pymysql: mysql을 파이썬에서 사용할 수 있는 라이브러리

터미널에서 사용하는 명령어를 주피터 노트북에서 사용하려면 명령어 앞에 !를 붙이고 실행하면 된다.

주피터 노트북의 셀에서 sudo가 붙은 명령어를 실행한다면 아래와 같이 암호를 입력하고 싶어도 입력할 수 없으므로 터미널에서 실행하도록 한다.

```
In [*]: # 설치
!sudo apt-get install mysql-server
[sudo] try 암호:
```


MySQL 환경 구축

```
import pymysql
```

```
# MySQL 접속에 사용할 아이디와 비밀번호를 파일로 저장 및 관리하기 위한 파일이름과 경로를 담은 변수  
MYSQL_USER_DATA_SAVED_FILE = "res/mysql/userinfo"
```

```
# Linux  
# p 옵션: 존재하지 않는 중간 디렉토리를 자동으로 생성 (여기서는 중간 디렉토리인 res 디렉토리가 존재하지 않을 경우에 생성한 후 mysql 디렉토리 생성)  
!mkdir -p res/mysql/  
  
# Windows  
#!mkdir res\mysql
```

```
# 디렉토리가 생성되었는지 확인  
!ls
```

```
python_snippets.ipynb  res
```



```
# mysql 계정을 만들어야함  
# sudo mysql -u root -p  
# create schema `pydb` default character set utf8;  
# create user eddi@localhost identified by 'eddi@123';  
# grant all privileges on pydb.* to eddi@localhost;  
# flush privileges;
```

MySQL 실습

```
import pickle

mysql_user_info = dict({
    'user_id': 'eddi',
    'password': 'eddi@123'
})

f = open(MYSQL_USER_DATA_SAVED_FILE, 'wb')
pickle.dump(mysql_user_info, f)
f.close() # 파일을 열었으면 닫아주어야 한다

# 삭제하기 전에 확인
print(mysql_user_info)

# 파일을 저장하였으므로 del 키워드를 사용하여 객체를 삭제
del mysql_user_info
```

{'user_id': 'eddi', 'password': 'eddi@123'} ← 객체를 삭제하기 전에 print 함수를 사용하여 출력한 결과

```
# 객체 삭제 후 확인
print(mysql_user_info)
```

```
-----
NameError                                Traceback (most recent call last)
/tmp/ipykernel_30628/2597053894.py in <module>
----> 1 print(mysql_user_info)

NameError: name 'mysql_user_info' is not defined
```

```
# 파일이 생성되었는지 확인
!ls res/mysql
```

userinfo

MySQL 실습

저장된 파일로부터 내용을 읽어오는 작업

```
import pickle
```

```
f = open(MYSQL_USER_DATA_SAVED_FILE, 'rb') # read binary  
mysql_user_info = pickle.load(f)  
f.close()
```

파일을 제대로 읽어왔는지 확인

```
print(mysql_user_info)
```

```
{'user_id': 'eddi', 'password': 'eddi@123'}
```

127.0.0.1

127.0.0.1은 localhost와 같은 의미이다. localhost는 자신의 컴퓨터이다.

포트

포트: 일명 서비스 번호라고 한다.

유명한 포트는 ssh: 22, http: 80, https: 443 이고 지금 사용하는 MySQL은 3306이다.

서비스 포트 번호를 나눈 이유는 컴퓨터는 각자의 아이피 주소를 가지고 있고 그 아이피 주소로 접근했는데 내가 사용할 서비스가 웹 서버인지 DB 서버인지 그거를 분간할 수 있는 방법이다.

한마디로 내가 어떤 서비스를 사용할 것인가 그 번호의 역할을 해주는 게 포트 번호이다.

스키마 (DB 용어)

데이터 저장소라고 생각하면 된다.

스키마는 여러 개가 있을 수 있다.

MySQL 실습

```
# DB 연결
# pymysql.connect() 메소드를 사용하여 MySQL에 연결
# 아래의 코드는 pydb라는 저장소를 사용하겠다는 코드
db = pymysql.connect(
    # localhost와 같은 아이피 주소임
    host = '127.0.0.1', ← '127.0.0.1' 대신 'localhost'로 해도 된다.
    # MySQL의 포트 번호
    port = 3306,
    # 아이디
    user = mysql_user_info['user_id'],
    # 비번
    passwd = mysql_user_info['password'],
    # 사용하려는 스키마
    db = 'pydb'
)

print(db)
```

<pymysql.connections.Connection object at 0x7f7fd4037a60>

cursor

1. MySQL 접속이 성공하면, Connection 객체로부터 cursor() 메서드를 호출하여 Cursor 객체를 가져옴
2. Cursor 객체의 execute() 메서드를 사용하여 SQL 문장을 DB 서버에 전송
3. SQL 쿼리의 경우 Cursor 객체의 fetchall(), fetchone(), fetchmany() 등의 메서드를 사용하여 서버로부터 가져온 데이터를 코드에서 활용

DB

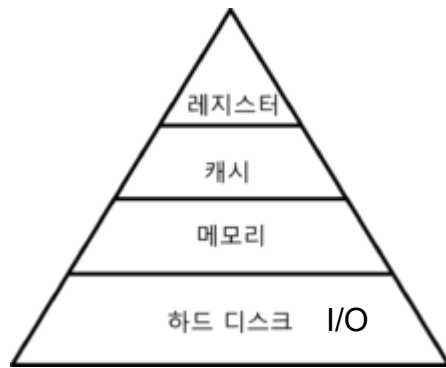
DB는 파일 시스템

파일 시스템하면 제일 먼저 떠오르는 것은 메모리 계층 구조의 I/O이다.

실제로 파일이라는 것은 디스크이다. 그러므로 제일 느린 쪽에 있다.

제일 느린 쪽에 있으니까 실제로 이걸 처리할 때 메모리에 쌓아놓고 있다가 일정량이 차면 한 번에 보내는게 좋다. 왜냐하면 I/O는 느리니까

그러다 보니까 db.commit() 명령어가 별도로 있는 것이다.



메모리 계층 구조

MySQL 실습

```
# 이제 DB에서 select 등을 수행했을때  
# 나오는 결과를 순회할 수 있도록 지원하는 객체를 가져온다.  
cursor = db.cursor()
```

```
print(cursor)
```

```
<pymysql.cursors.Cursor object at 0x7f7fd41bffd0>
```

```
# MySQL 수동 쿼리를 적용하여 table을 생성해보도록 한다.
```

```
sql = """  
    create table pydbtest(  
        id int unsigned not null auto_increment,  
        name varchar(20) not null,  
        price int not null,  
        primary key(id)  
    )  
    """
```

```
# 실제 SQL Query를 실행하는 코드
```

```
cursor.execute(sql)
```

```
# 작업한 내용을 실제 DB에 적용함
```

```
# I/O는 무겁기 때문에 별도로 commit이 존재함
```

```
db.commit()
```

```
# 작업 완료 이후 DB 연결을 끊는다.
```

```
db.close()
```

```
print("테이블 생성 성공!")
```

```
테이블 생성 성공!
```

MySQL 실습

테이블이 생성되었는지 터미널에서도 확인할 수 있다.

1. `mysql -u eddi -p`
 2. 비밀번호 입력 (여기서는 `eddi@123`)
 3. `show databases;`
 4. `use pydb;`
 5. `show tables;`
 6. `select * from pydbtest;`
- `desc` 테이블 이름;
 - 테이블의 구조를 볼 수 있다.

```
try@newtry-Lenovo: ~  
mysql> select * from pydbtest;  
Empty set (0.00 sec)  
  
mysql> desc pydbtest;  
+-----+  
| Field | Type          | Null | Key | Default | Extra          |  
+-----+  
| id    | int unsigned  | NO   | PRI | NULL    | auto_increment |  
| name  | varchar(20)   | NO   |     | NULL    |                |  
| price | int           | NO   |     | NULL    |                |  
+-----+  
3 rows in set (0.02 sec)  
  
mysql>
```

```
try@newtry-Lenovo: ~  
(base) try@newtry-Lenovo:~$ mysql -u eddi -p  
Enter password:  
welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 20  
Server version: 8.0.31-0ubuntu0.20.04.2 (Ubuntu)  
  
Copyright (c) 2000, 2022, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| performance_schema |  
| pydb        |  
+-----+  
3 rows in set (0.02 sec)  
  
mysql> use pydb;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_pydb |  
+-----+  
| pydbtest        |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```


MySQL 실습

```
# 생성한 테이블에 값을 넣는 코드이다.
# insert 처리
db = pymysql.connect(
    host = '127.0.0.1',
    port = 3306,
    user = mysql_user_info['user_id'],
    passwd = mysql_user_info['password'],
    db = 'pydb'
)

cursor = db.cursor()

# id 값이 없는 이유는 auto_increment인 자동 증가가 있어서, 우리가 입력하지 않아도 알아서 들어간다.
sql = """
insert into pydbtest(name, price) values (%s, %s)
"""

# %s에 해당하는 데이터들을 배치한다.
cursor.execute(sql, ('Xeon Server', 10000000))
cursor.execute(sql, ('FPGA Server', 34000000))
cursor.execute(sql, ('GPU Server', 20000000))
cursor.execute(sql, ('RTX 3090', 4000000))

db.commit()

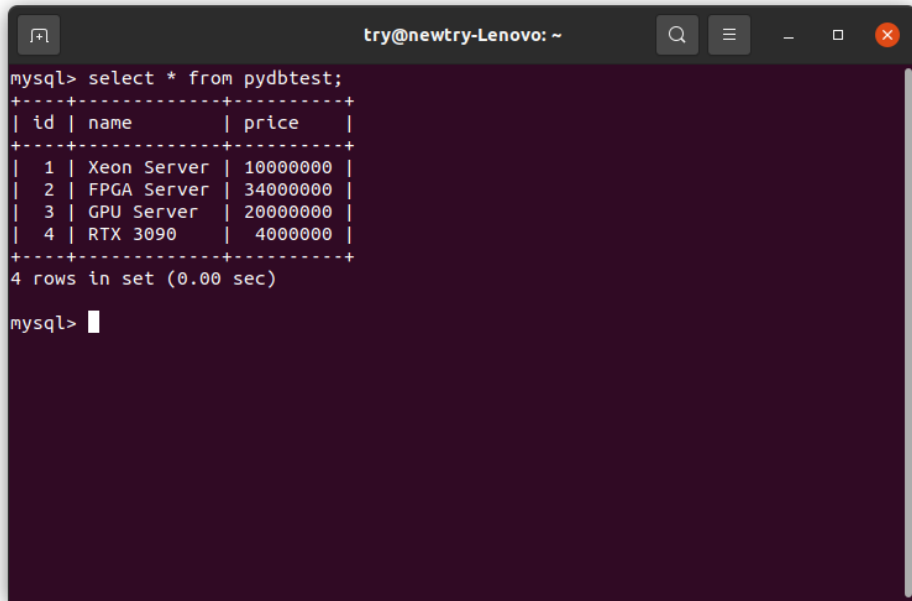
db.close()

print("테이블에 값 넣기 성공!")
```

테이블에 값 넣기 성공!

MySQL 실습

테이블에 값이 잘 들어갔는지 터미널로 확인



```
try@newtry-Lenovo: ~  
mysql> select * from pydbtest;  
+-----+  
| id | name      | price |  
+-----+  
| 1  | Xeon Server | 10000000 |  
| 2  | FPGA Server | 34000000 |  
| 3  | GPU Server  | 20000000 |  
| 4  | RTX 3090   | 4000000  |  
+-----+  
4 rows in set (0.00 sec)  
  
mysql> 
```

MySQL 실습

앞에서 했던 작업을 파이썬 코드로도 볼 수 있다.

```
# select(값 가져오기)
db = pymysql.connect(
    host = '127.0.0.1',
    port = 3306,
    user = mysql_user_info['user_id'],
    passwd = mysql_user_info['password'],
    db = 'pydb'
)
```

```
cursor = db.cursor()
```

```
sql = """
select * from pydbtest;
"""
```

```
cursor.execute(sql)
```

```
# cursor를 통해 가져온 모든 데이터를 rows에 배치한다.
# 객체에 저장!
```

```
rows = cursor.fetchall()
print(rows)
```

```
db.close()
```

```
print("데이터 불러오기 성공!")
```

그리고 select는 commit 이 없다.

이유

커밋이 있는 애들은 메모리가 저장해 놔다가 commit으로 내린다
작업이 write이다!!

근데 select는 저장되어 있는 정보를 메모리로 가져오는 거다.
그러므로 read이다.

그냥 불러올 뿐이라 commit이 필요없다.

← fetchall(): Fetch all the rows

```
((1, 'Xeon Server', 10000000), (2, 'FPGA Server', 34000000), (3, 'GPU Server', 20000000), (4, 'RTX 3090', 40000000))
데이터 불러오기 성공!
```

‘불러온다’의 의미

디스크 내용을 메모리에다가 저장한다.

옛날 게임 Save, Load 도 동일하다.

save는 메모리에 있던 내용을 디스크에 내리는 것이고, load는 디스크에 있던 파일을 메모리로 올리는 것이다.

옛날 게임에서 볼 수 있었던 시스템이다.

MySQL 실습

여기서 원하는 컬럼만 보고 싶을 경우에 사용하는 것은 cursor의 DictCursor 이다.

원래 cursor를 나누지 못하는데, 나눌 수 있게 서포트하는 것이 DictCursor이다.

```
# 날개 내에서 각각의 데이터를 얻기(딕셔너리 형식이라 키값으로 뽑으면 됨)
# select(값 가져오기) 날개로 가져오기
db = pymysql.connect(
    host = '127.0.0.1',
    port = 3306,
    user = mysql_user_info['user_id'],
    passwd = mysql_user_info['password'],
    db = 'pydb'
)

# 날개로 가져올 때 사용
cursor = db.cursor(pymysql.cursors.DictCursor)

sql = """
select * from pydbtest;
"""

cursor.execute(sql)

# cursor를 통해 가져온 모든 데이터를 rows에 배치한다.
rows = cursor.fetchall()

for row in rows:
    print('id: ', row['id'], ', name: ', row['name'], ', price: ', row['price'])

db.close()

print("데이터 날개로 불러와 특정값 추출하기 성공!")
```

```
id: 1 , name: Xeon Server , price: 10000000
id: 2 , name: FPGA Server , price: 34000000
id: 3 , name: GPU Server , price: 20000000
id: 4 , name: RTX 3090 , price: 4000000
데이터 날개로 불러와 특정값 추출하기 성공!
```

MySQL 실습

```
# 입력한 데이터값 정리하기(테이블 삭제)
db = pymysql.connect(
    host = '127.0.0.1',
    port = 3306,
    user = mysql_user_info['user_id'],
    passwd = mysql_user_info['password'],
    db = 'pydb'
)

cursor = db.cursor()

sql = """
drop table pydbtest;
"""

cursor.execute(sql)

db.commit()
db.close()

print("테이블 삭제 성공!")
```

테이블 삭제 성공!

실제로 데이터베이스를 어디에다가 써먹으면 좋냐면 RC 카를 만들면 사람이 수동으로 조종할 텐데 그 수동으로 조종한 정보를 데이터베이스화한다.

PWM 얼마 넣었고 어떤 프로토콜을 보냈는지 타이밍 값을 저장한 다음에 DB에 있는 정보를 읽으면서 반자율주행처럼 동작하게 만들 수 있다.

한국전력공사 회사라고 가정하고 테이블을 하나 만든다.

여기서 전압과 전류는 소수점을 가질 수 있어서 float가 들어간다.

원래는 없었던 기능이라 옛날에는 조금 복잡하게 소수점을 만들었어야 했다고 한다.

```
# 소수점 데이터 처리하기!
db = pymysql.connect(
    host = '127.0.0.1',
    port = 3306,
    user = mysql_user_info['user_id'],
    passwd = mysql_user_info['password'],
    db = 'pydb'
)

cursor = db.cursor()

sql = """
    create table power_data(
        no integer not null auto_increment primary key,
        voltage float not null,
        current float not null,
        vendor text null,
        name text not null,
        reg_date timestamp not null default now()
    )
"""

cursor.execute(sql)
db.commit()
db.close()

print("float 형식의 테이블 생성 성공!")

float 형식의 테이블 생성 성공!
```


딤러닝할 때도 많이 나오는 얘기 중 하나이다.

랜덤의 종류

1. 정규 분포 (가우시안)
 - 현실 세계는 대부분 정규 분포를 따르고 있다.
1. 균일 분포
2. 푸아송 분포
 - 보편적으로 제품의 수명을 계산할 때 많이 쓰는 확률 분포이다.
 - 보통 다음과 같은 경우에 많이 쓴다고 한다. 1년에 이 장비가 두 번 정도 오동작을 한다. 오늘 동작시킬 때 오동작할 확률은 얼마인가?

MySQL 실습

```
import time
import math
import calendar
import datetime
import numpy as np
```

```
# 0부터 10초까지의 시간을 0.001 단위로 자름
sample_time = np.arange(0, 10, 0.001)
```

```
print(sample_time)
```

```
[0.000e+00 1.000e-03 2.000e-03 ... 9.997e+00 9.998e+00 9.999e+00]
```

```
print(sample_time.size)
```

```
10000
```

```
# 가우시안 분포(정규 분포) 형식을 따르는 랜덤을 만듦
# 용어 필요 없고 데이터 개수에 따른 랜덤을 적절하게 만들어줌(자연스럽게)
noise = np.random.normal(size = len(sample_time))
```

```
print(noise)
```

```
[-0.09028079 -2.16141996 -0.60871834 ... -1.44169738 -1.12661199
 0.43732947]
```

노이즈 신호를 10,000개 만든다.
왜냐하면 실제 전력을 계측한다고 하더라도 순전력이 나오지
않고 노이즈가 낄 것이기 때문이다.

MySQL 실습

```
import time
import math
import calendar
import datetime
import numpy as np
```

```
# 0부터 10초까지의 시간을 0.001 단위로 자름
sample_time = np.arange(0, 10, 0.001)
```

```
print(sample_time)
```

```
[0.000e+00 1.000e-03 2.000e-03 ... 9.997e+00 9.998e+00 9.999e+00]
```

```
print(sample_time.size)
```

```
10000
```

```
# 가우시안 분포(정규 분포) 형식을 따르는 랜덤을 만들  
# 용어 필요 없고 데이터 개수에 따른 랜덤을 적절하게 만들어줌(자연스럽게)  
noise = np.random.normal(size = len(sample_time))
```

```
print(noise)
```

```
[ 0.10608    1.2149614 -0.11481048 ... 0.02284635 -1.13848181  
 -0.7816518 ]
```

```
# 전압 · 전류 신호를 간단하게 만드는 코드이다.  
#  $2 * \pi * f * t = w = 5 \implies f = 5 / (2 * \pi)$   
voltage = 3 * np.cos(5 * sample_time) + noise  
current = 3 * np.sin(3 * sample_time + (math.pi / 3)) + noise
```

```
print(voltage)  
print(current)
```

```
[3.10608    4.2149239 2.88503953 ... 2.90561234 1.74840042 2.10927449]  
[ 2.70415621 3.81752591 2.49221892 ... -1.08358516 -2.23654281  
 -1.87133242]
```

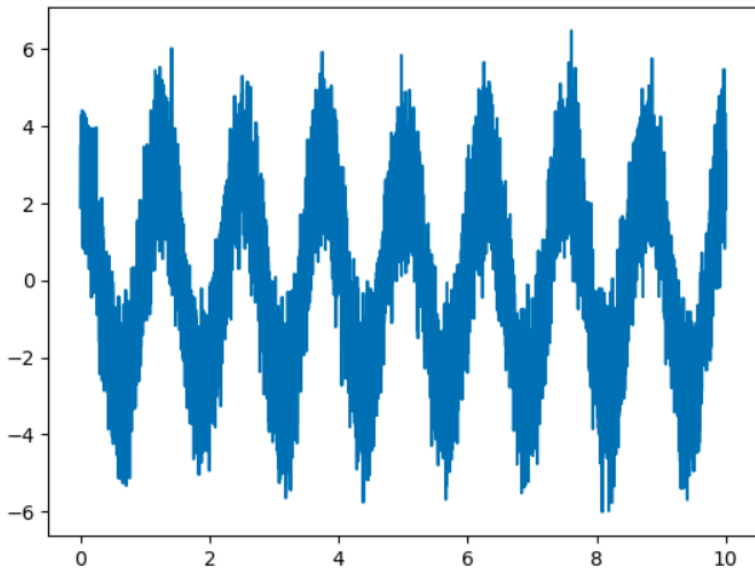
노이즈 신호를 10,000개 만든다.
왜냐하면 실제 전력을 계측한다고 하더라도 순전력이 나오지 않고
노이즈가 낄 것이기 때문이다.

MySQL 실습

```
# 시각화 라이브러리 설치  
!pip install matplotlib
```

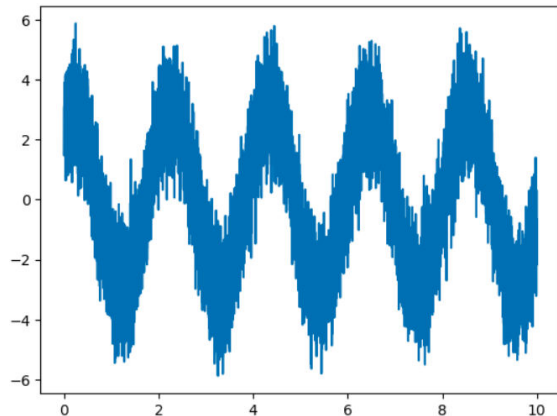
```
import matplotlib.pyplot as plt
```

```
# 아래 plot 함수의 인자의 첫 번째와 두 번째는 각각 x 좌표와 y 좌표이다.  
plt.plot(sample_time, voltage)  
plt.show()
```



MySQL 실습

```
plt.plot(sample_time, current)
plt.show()
```



```
# 현재 시간값을 컴퓨터가 계산하고 있는 수치로 가져옴
# timestamp 값을 가져온다
ts = calendar.timegm(time.gmtime())

print(ts)

# sample_time이 이미 1000개짜리 배열이므로
# 곱셈을 수행할 경우 알아서 1000개 배열에 1000씩 곱해짐(자동으로)
# 샘플 타임이 0.001초(1 / 1000 초)를 대상으로 하고 있음
# 전체 샘플 타임의 개수는 10000개(만개)이므로 1 / 1000 초 단위 샘플링을 하면
# 10000 / 1000 = 10초간의 데이터를 수집함을 의미함
ts = (sample_time * 1000) + ts # 테이블에 값을 저장하기 위해 시간을 만들었다고 생각하면 된다

print(ts)

1673601363
[1.67360136e+09 1.67360136e+09 1.67360136e+09 ... 1.67361136e+09
 1.67361136e+09 1.67361136e+09]
```

MySQL 실습

```
In [21]: # 실제 db에 넣기 위해 상호간의 데이터 타입을 np.float32로 맞춰주어야 함(이것을 해줘야 DB에 입력 가능함)
sample_time = np.array(sample_time, dtype=np.float32)
voltage = np.array(voltage, dtype=np.float32)
current = np.array(current, dtype=np.float32)
```

```
In [22]: curr_time = []

# datetime.datetime을 통해 시간값을 제어할 수 있으며
# datetime.datetime.fromtimestamp(컴퓨터시간).strftime(형식)을 통해
# 원하는 형태로 시간을 형 변환할 수 있음
for i in range(len(sample_time)):
    curr_time.append(
        datetime.
            datetime.
                fromtimestamp(ts[i]).
                    strftime('%Y-%m-%d %H:%M:%S')
    )

print(curr_time)
```

```
['2023-01-13 18:16:03', '2023-01-13 18:16:04', '2023-01-13 18:16:05', '2023-01-13 18:16:06', '2023-01-13 18:16:07', '2023-01-13 18:16:08', '2023-01-13 18:16:09', '2023-01-13 18:16:10', '2023-01-13 18:16:11', '2023-01-13 18:16:12', '2023-01-13 18:16:13', '2023-01-13 18:16:14', '2023-01-13 18:16:15', '2023-01-13 18:16:16', '2023-01-13 18:16:17', '2023-01-13 18:16:18', '2023-01-13 18:16:19', '2023-01-13 18:16:20', '2023-01-13 18:16:21', '2023-01-13 18:16:22', '2023-01-13 18:16:23', '2023-01-13 18:16:24', '2023-01-13 18:16:25', '2023-01-13 18:16:26', '2023-01-13 18:16:27', '2023-01-13 18:16:28', '2023-01-13 18:16:29', '2023-01-13 18:16:30', '2023-01-13 18:16:31', '2023-01-13 18:16:32', '2023-01-13 18:16:33', '2023-01-13 18:16:34', '2023-01-13 18:16:35', '2023-01-13 18:16:36', '2023-01-13 18:16:37', '2023-01-13 18:16:38', '2023-01-13 18:16:39', '2023-01-13 18:16:40', '2023-01-13 18:16:41', '2023-01-13 18:16:42', '2023-01-13 18:16:43', '2023-01-13 18:16:44', '2023-01-13 18:16:45', '2023-01-13 18:16:46', '2023-01-13 18:16:47', '2023-01-13 18:16:48', '2023-01-13 18:16:49', '2023-01-13 18:16:50', '2023-01-13 18:16:51', '2023-01-13 18:16:52', '2023-01-13 18:16:53', '2023-01-13 18:16:54', '2023-01-13 18:16:55', '2023-01-13 18:16:56', '2023-01-13 18:16:57', '2023-01-13 18:16:58', '2023-01-13 18:16:59', '2023-01-13 18:17:00', '2023-01-13 18:17:01', '2023-01-13 18:17:02', '2023-01-13 18:17:03', '2023-01-13 18:17:04', '2023-01-13 18:17:05', '2023-01-13 18:17:06', '2023-01-13 18:17:07', '2023-01-13 18:17:08', '2023-01-13 18:17:09', '2023-01-13 18:17:10', '2023-01-13 18:17:11', '2023-01-13 18:17:12', '2023-01-13 18:17:13', '2023-01-13 18:17:14', '2023-01-13 18:17:15', '2023-01-13 18:17:16', '2023-01-13 18:17:17', '2023-01-13 18:17:18', '2023-01-13 18:17:19', '2023-01-13 18:17:20', '2023-01-13 18:17:21', '2023-01-13 18:17:22', '2023-01-13 18:17:23', '2023-01-13 18:17:24', '2023-01-13 18:17:25', '2023-01-13 18:17:26', '2023-01-13 18:17:27', '2023-01-13 18:17:28', '2023-01-13 18:17:29', '2023-01-13 18:17:30', '2023-01-13 18:17:31', '2023-01-13 18:17:32', '2023-01-13 18:17:33', '2023-01-13 18:17:34', '2023-01-13 18:17:35', '2023-01-13 18:17:36', '2023-01-13 18:17:37', '2023-01-13 18:17:38', '2023-01-13 18:17:39', '2023-01-13 18:17:40', '2023-01-13 18:17:41', '2023-01-13 18:17:42', '2023-01-13 18:17:43', '2023-01-13 18:17:44', '2023-01-13 18:17:45', '2023-01-13 18:17:46', '2023-01-13 18:17:47', '2023-01-13 18:17:48', '2023-01-13 18:17:49', '2023-01-13 18:17:50', '2023-01-13 18:17:51', '2023-01-13 18:17:52', '2023-01-13 18:17:53', '2023-01-13 18:17:54', '2023-01-13 18:17:55', '2023-01-13 18:17:56', '2023-01-13 18:17:57', '2023-01-13 18:17:58', '2023-01-13 18:17:59', '2023-01-13 18:18:00']
```

MySQL 실습

```
db = pymysql.connect(
    host = '127.0.0.1',
    port = 3306,
    user = mysql_user_info['user_id'],
    passwd = mysql_user_info['password'],
    db = 'pydb'
)

cursor = db.cursor()

# 소수점 형식의 데이터와 현재 시간값을 기록하도록 한다.
sql = """
    insert into power_data(
        voltage, current, vendor, name, reg_date
    ) values(
        %s, %s, %s, %s, %s
    )
"""

# volt, curr이 np.float32
# DB에 넣는 과정은 %s(문자열) 형식이므로 반드시 str()로 한 번 감싸줘야함
try:
    with db.cursor() as cursor:
        for i in range(len(sample_time)):
            cursor.execute(sql,
                (
                    str(voltage[i]),
                    str(current[i]),
                    'EDDI',
                    'P32N21BTN77PW',
                    curr_time[i]
                )
            )

        db.commit()
finally:
    db.close()

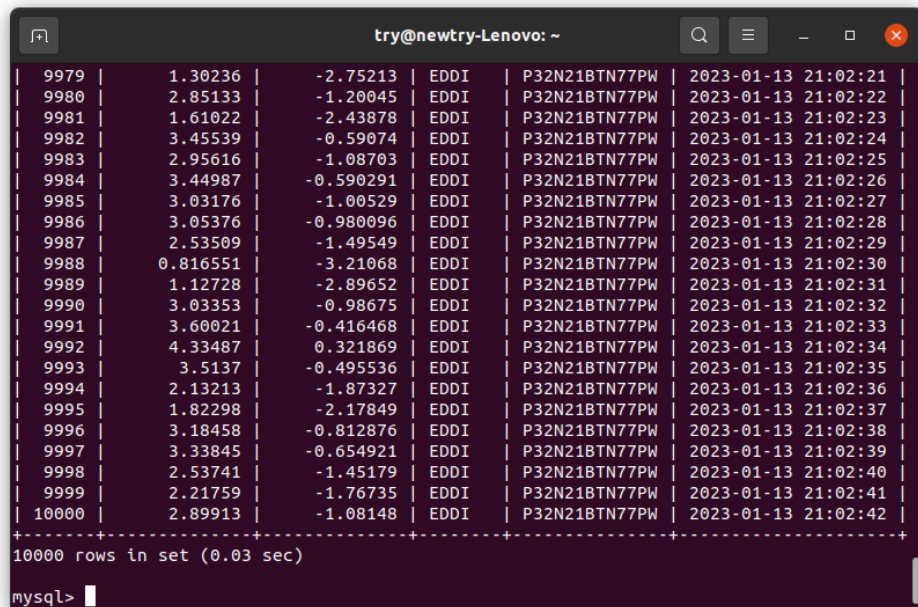
print('소수점 형식 데이터 및 날짜 형식 데이터 입력 완료!')
```

소수점 형식 데이터 및 날짜 형식 데이터 입력 완료!

MySQL 실습

저장한 값을 터미널로 확인해 보자

- `select * from power_data;`



```
try@newtry-Lenovo: ~  
+-----+  
| 9979 | 1.30236 | -2.75213 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:21 |  
| 9980 | 2.85133 | -1.20045 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:22 |  
| 9981 | 1.61022 | -2.43878 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:23 |  
| 9982 | 3.45539 | -0.59074 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:24 |  
| 9983 | 2.95616 | -1.08703 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:25 |  
| 9984 | 3.44987 | -0.590291 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:26 |  
| 9985 | 3.03176 | -1.00529 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:27 |  
| 9986 | 3.05376 | -0.980096 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:28 |  
| 9987 | 2.53509 | -1.49549 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:29 |  
| 9988 | 0.816551 | -3.21068 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:30 |  
| 9989 | 1.12728 | -2.89652 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:31 |  
| 9990 | 3.03353 | -0.98675 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:32 |  
| 9991 | 3.60021 | -0.416468 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:33 |  
| 9992 | 4.33487 | 0.321869 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:34 |  
| 9993 | 3.5137 | -0.495536 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:35 |  
| 9994 | 2.13213 | -1.87327 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:36 |  
| 9995 | 1.82298 | -2.17849 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:37 |  
| 9996 | 3.18458 | -0.812876 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:38 |  
| 9997 | 3.33845 | -0.654921 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:39 |  
| 9998 | 2.53741 | -1.45179 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:40 |  
| 9999 | 2.21759 | -1.76735 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:41 |  
| 10000 | 2.89913 | -1.08148 | EDDI | P32N21BTN77PW | 2023-01-13 21:02:42 |  
+-----+  
10000 rows in set (0.03 sec)  
mysql>
```


구글에 pymysql을 검색하던 도중 좋은 사이트를 찾았다.

- pymysql: https://www.fun-coding.org/mysql_basic6.html
- flask: https://www.fun-coding.org/flask_basic-2.html

Flask 객체가 무엇인지 검색하던 도중 좋은 사이트를 찾았다.

- flask: <https://flask-docs-kr.readthedocs.io/ko/latest/quickstart.html>

pickle

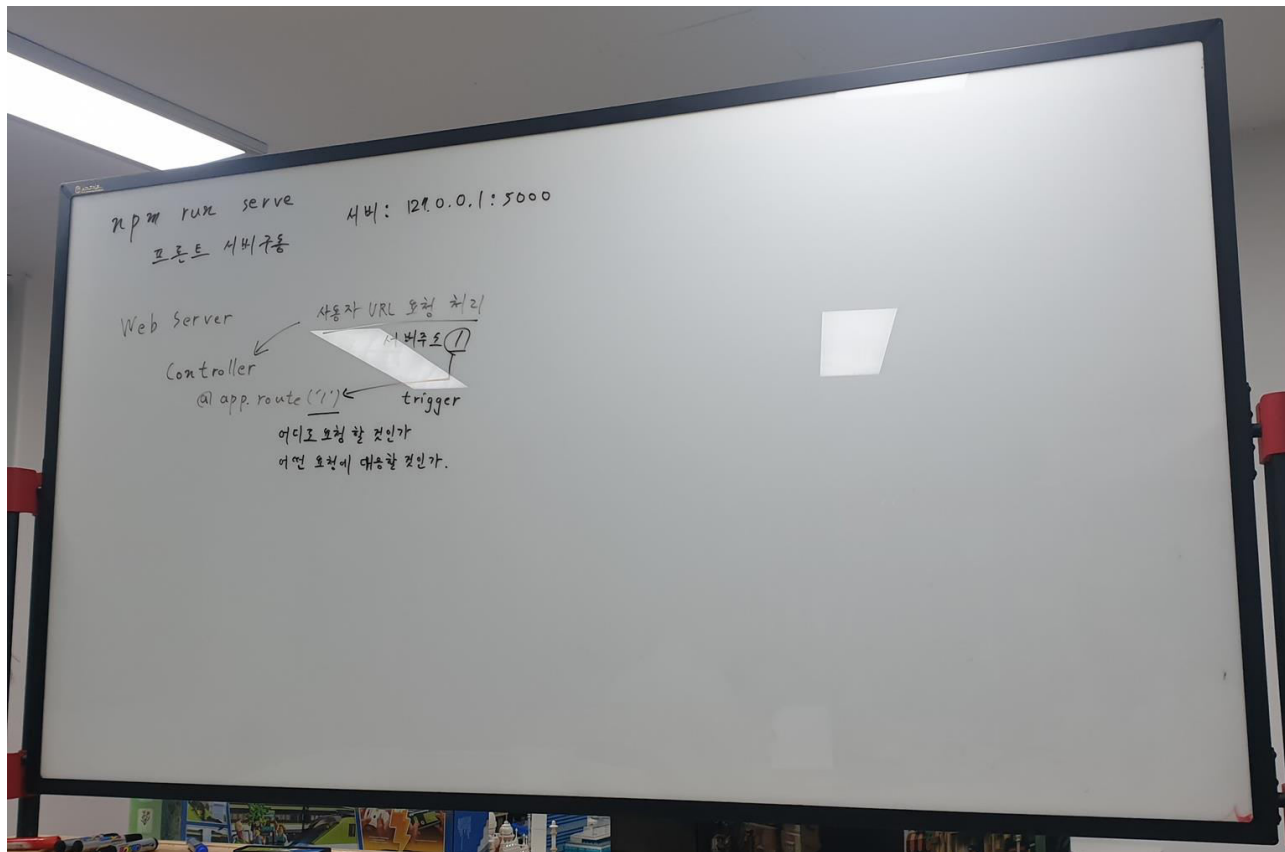
- <https://wikidocs.net/110788>

matplotlib

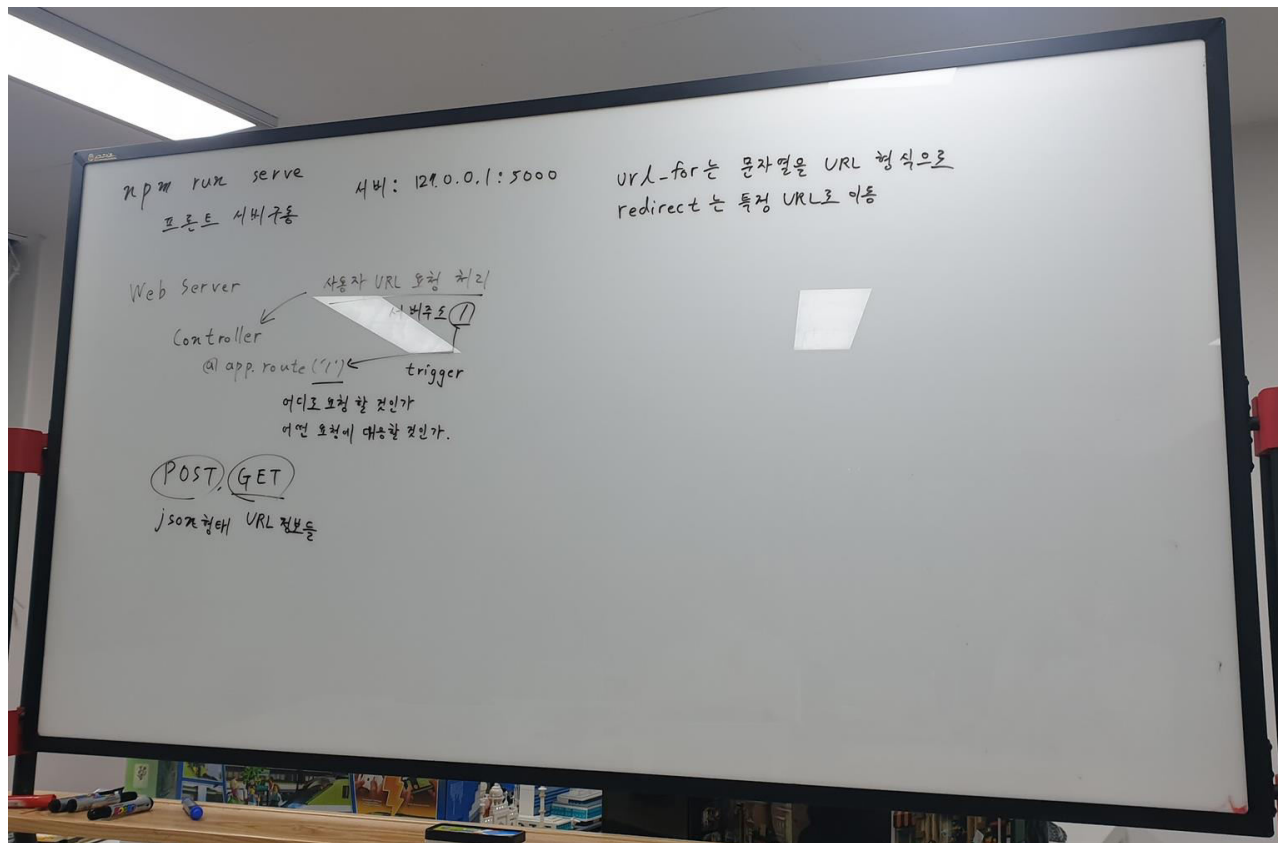
- <https://wikidocs.net/92071>

<https://moonhy7.tistory.com/entry/Database-Database-π π 연습>

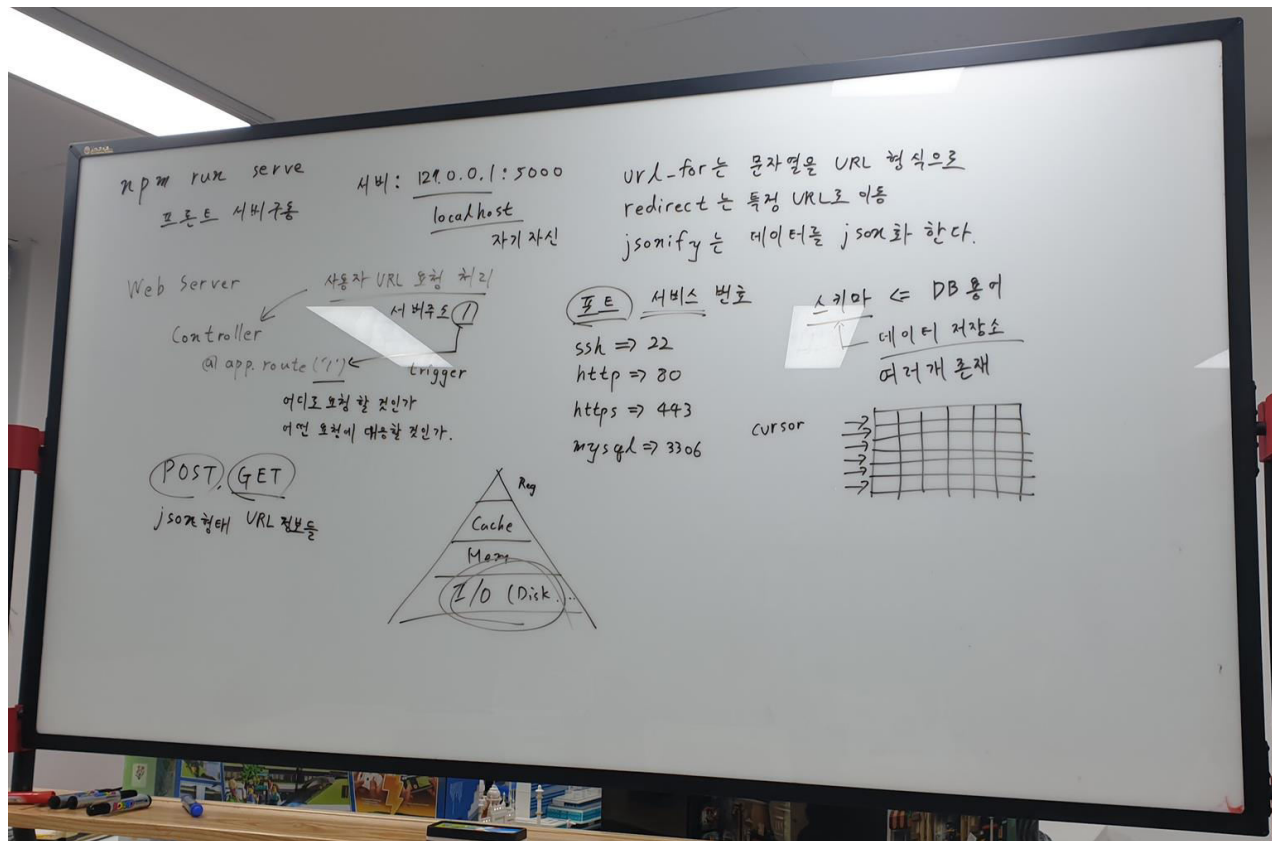
수업내용 사진



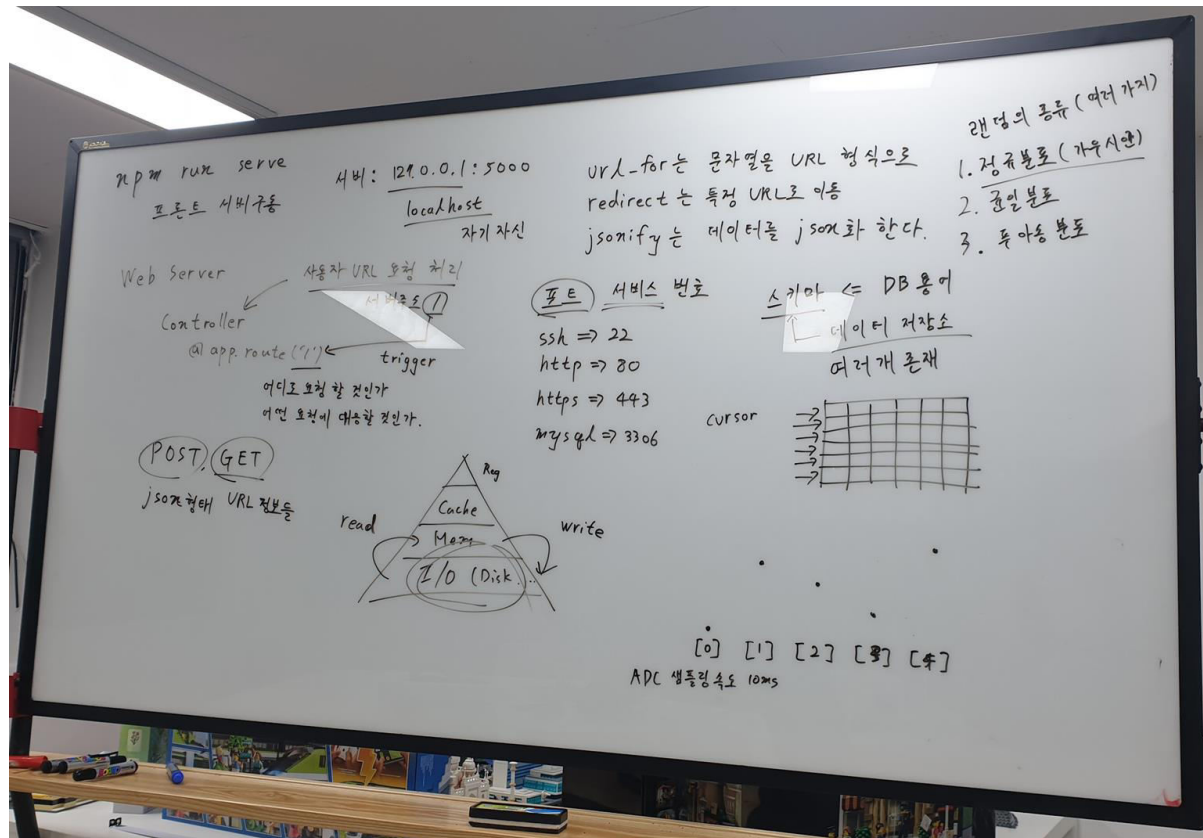
수업내용 사진



수업내용 사진



수업내용 사진



CONTENTS

1) 형식은 자유롭게~~~

<공부 내용을 적어주세요.>