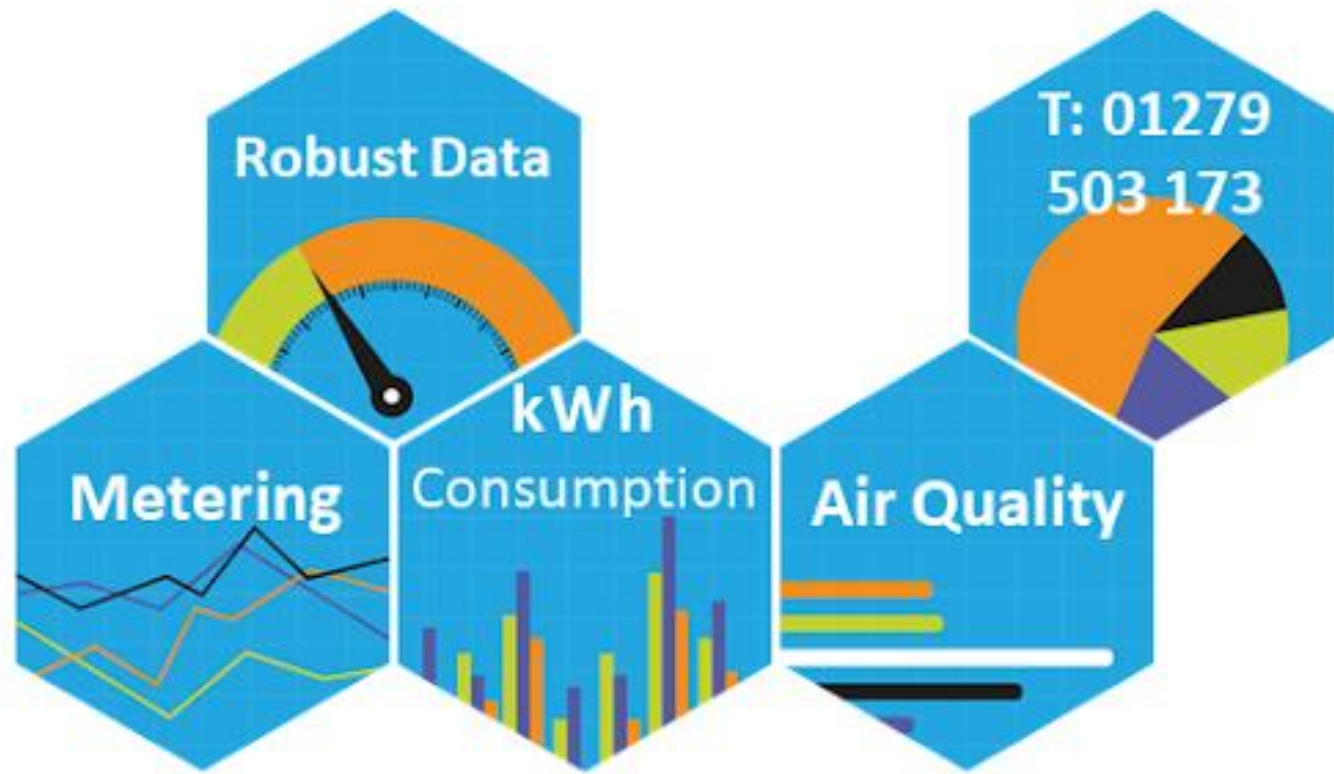
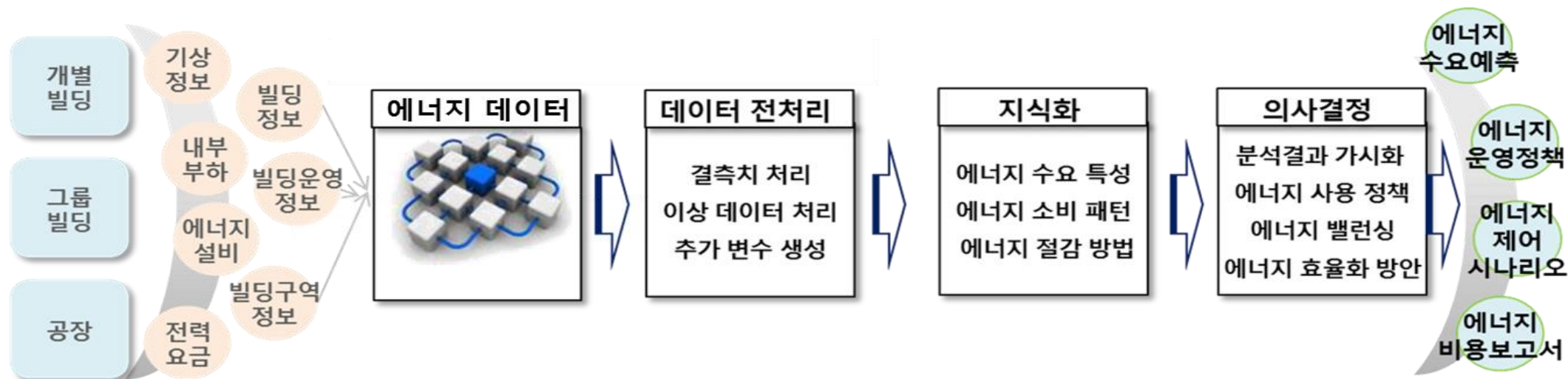


### 3. 에너지데이터 수집 실습



# 에너지 데이터 프로세싱

## 에너지 데이터 프로세싱



## 데이터 프로세싱 라이브러리/프로그램

데이터 수집

PyModbus

BACpypes

SOCKET

HTTP

REST API

데이터베이스

PostgreSQL

MariaDB  
FOUNDATION

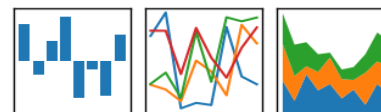
ORACLE

Microsoft SQL Server

데이터 분석

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



머신러닝, 딥러닝

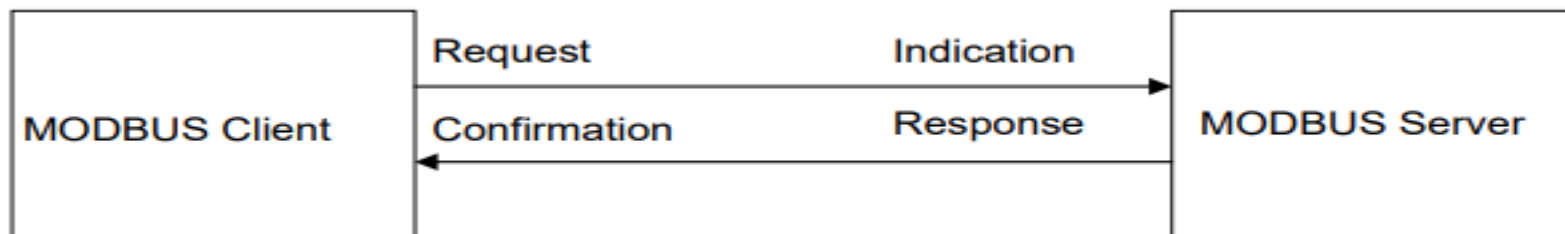
scikit  
learn

TensorFlow™

# 데이터 수집 프로토콜 - MODBUS

1979년 Modicon사에서 개발한 간단한 산업용 프로토콜로 전력시스템에 많이 사용되고 있습니다.

## MODBUS 구조



- 기본 포트번호 : TCP 502번
- 클라이언트가 요청(Request)을 보내면 서버가 그에 대한 응답(Response)을 보내는 구조

## MODBUS 데이터 타입

Name	Type	Access	Visual
Discrete Input	single bit	read-only	
Discrete Output (Coils)	single bit	read-write	
Input Registers	16-bit word	read-only	
Holding Registers (Registers)	16-bit word	read-write	


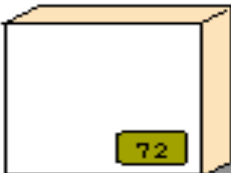
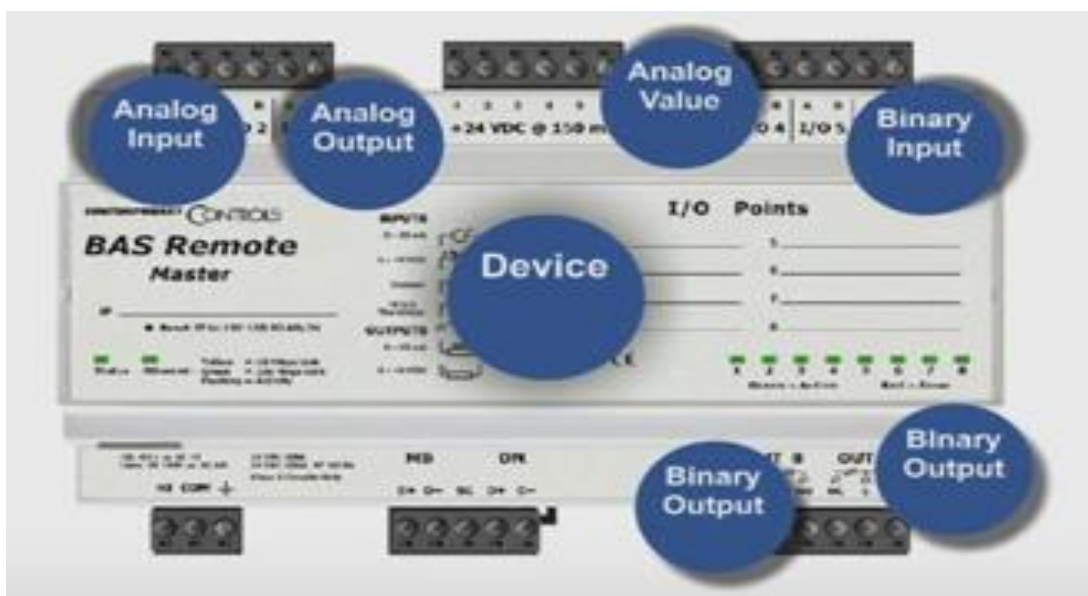


<https://system-monitoring.readthedocs.io/en/latest/modbus.html>

# 데이터 수집 프로토콜 - BACnet

미국표준협회(ANSI)와 냉동공조기술협회(ASHRAE)가 공동으로 채용/지지하는 표준 프로토콜

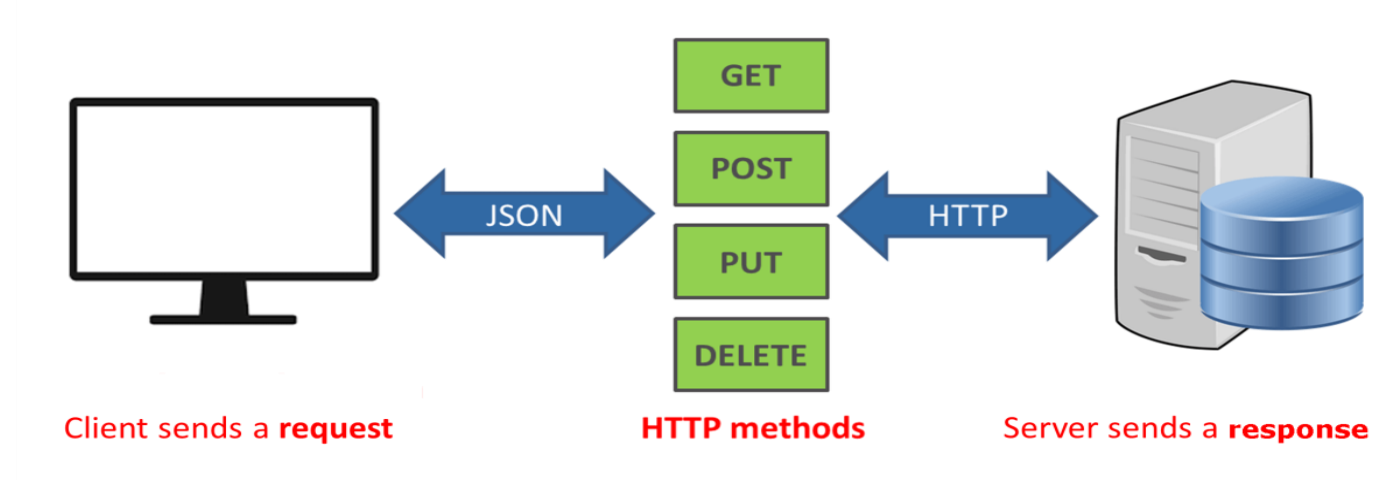
- What is BACnet? : <https://www.youtube.com/watch?v=oevGXrkxEos>



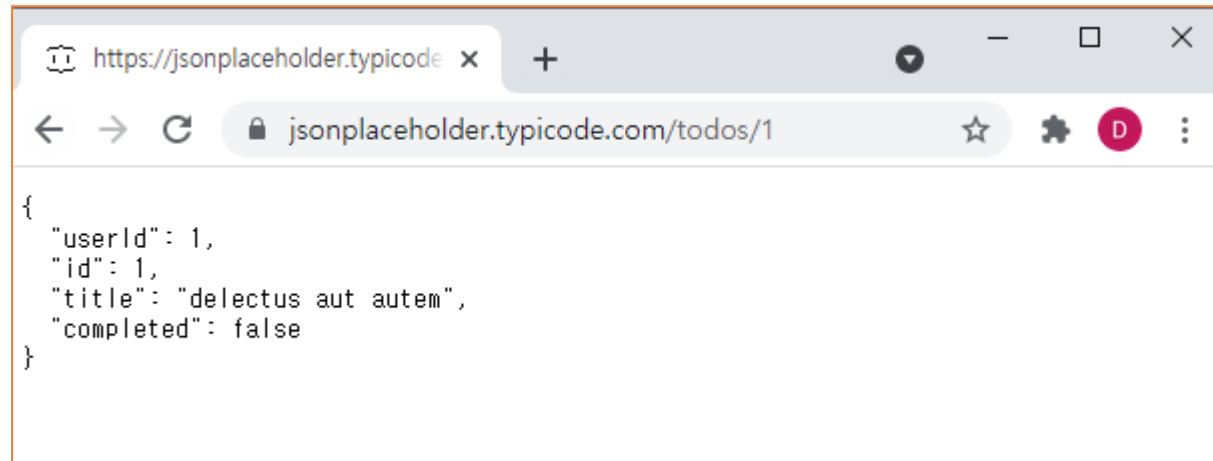
Object_Name	SPACE TEMP
Object_Type	ANALOG INPUT
Present_Value	72.3
Status_Flags	Normal, Out-of-Service
High_Limit	78.0
Low_Limit	68.0

# 데이터 수집 프로토콜 - REST

HTTP를 통해 CRUD를 실행하는 API로 GET, POST, PUT, DELETE 메서드와 JSON(JavaScript Object Notation) 데이터 포맷을 사용하며 대부분의 OPEN API에서 활용하고 있습니다.



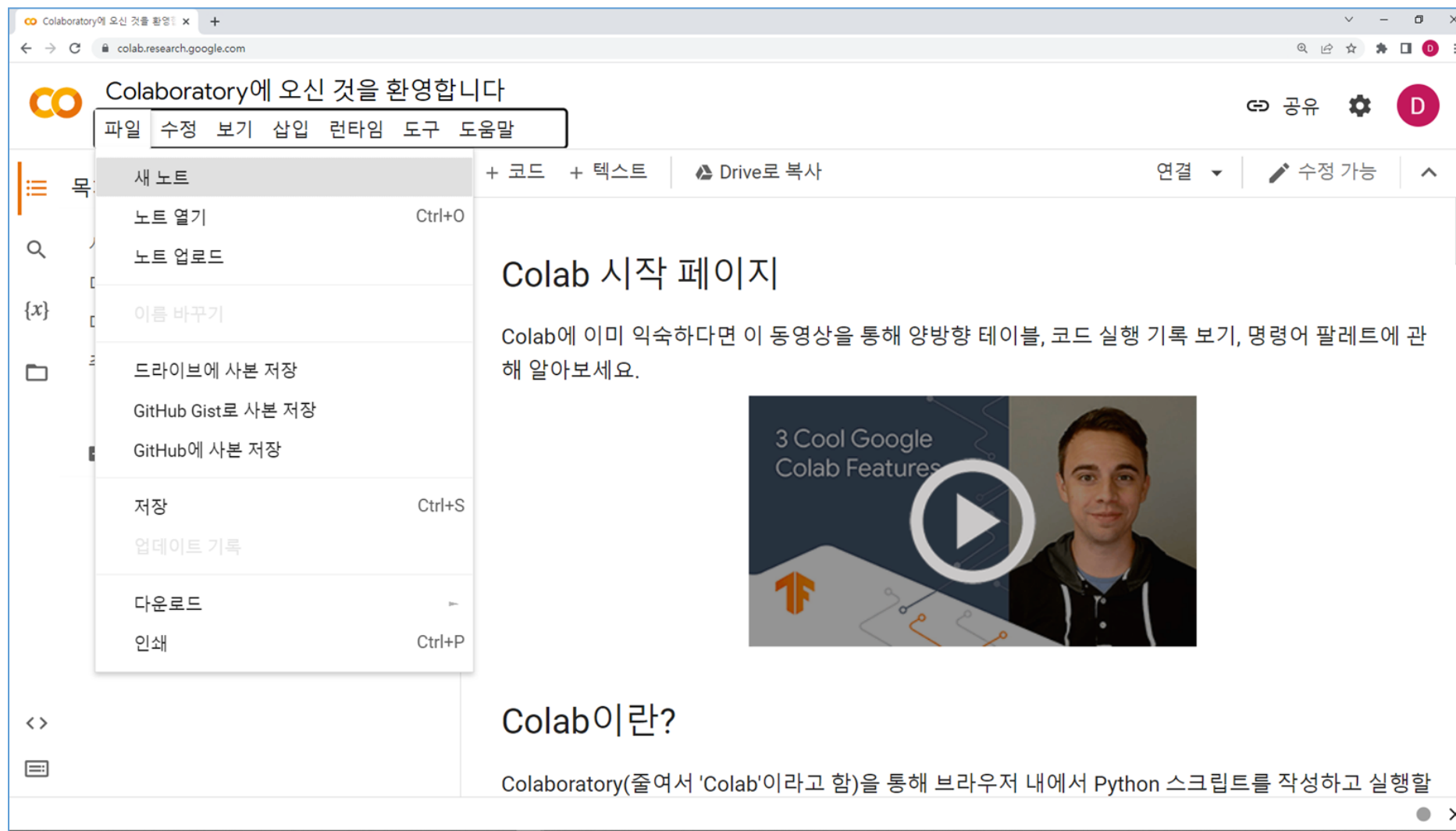
<https://jsonplaceholder.typicode.com/todos/1>



# 실습환경 - 구글 코랩(Colab)

<https://colab.research.google.com>

구글 계정 필요





# DB 테이블 생성 및 데이터 저장

```
[1] import sqlite3

# 데이터베이스 생성 및 접속
con = sqlite3.connect("bems.db")
cursor = con.cursor()

# energy 테이블 생성
cursor.executescript("""
    DROP TABLE IF EXISTS energy;
    CREATE TABLE energy( date_time text, b_code text, usage real,
        peak real, unit_price real, temp real, rh real ); """)

# energy 테이블에 데이터 추가(insert)
cursor.execute("INSERT INTO energy VALUES('202205131000', 'SGBD', 424.8, 1699.2, 100, 7, 50)")
cursor.execute("INSERT INTO energy VALUES('202205131015', 'SGBD', 434.8, 1799.2, 100, 8, 52)")
cursor.execute("INSERT INTO energy VALUES('202205131030', 'SGBD', 444.8, 1899.2, 100, 9, 55)")
cursor.execute("INSERT INTO energy VALUES('202205131045', 'SGBD', 454.8, 1799.2, 100, 8, 59)")
con.commit()
con.close()
```

# 데이터 조회

```
[2] con = sqlite3.connect("bems.db")
    with con:
        cursor = con.cursor()
        cursor.execute("SELECT * FROM energy")
        rows = cursor.fetchall()
        for row in rows:
            print(row)
    con.close()
```

```
('202205131000', 'SGBD', 424.8, 1699.2, 100.0, 7.0, 50.0)
('202205131015', 'SGBD', 434.8, 1799.2, 100.0, 8.0, 52.0)
('202205131030', 'SGBD', 444.8, 1899.2, 100.0, 9.0, 55.0)
('202205131045', 'SGBD', 454.8, 1799.2, 100.0, 8.0, 59.0)
```



# 파일 데이터를 DB에 저장

```
[3] import sqlite3
import csv

con = sqlite3.connect("bems.db")
cursor = con.cursor()

reader = csv.reader(open('data.csv', 'r'), delimiter=',')

# csv 헤더 스킵
next(reader)

for row in reader:
    to_db = [ row[0], row[1], row[2], row[3] ]
    cursor.execute("INSERT INTO energy (date_time, b_code, usage, peak ) VALUES (?, ?, ?,?);", to_db)

con.commit()
con.close()
```

# 데이터 선택 조회

```
[4] import sqlite3

con = sqlite3.connect("bems.db")
with con:
    cursor = con.cursor()
    # 2022년 01월 07일 15시 데이터만 조회
    cursor.execute(" SELECT date_time, usage, peak FROM energy #
                    WHERE b_code = 'SGBD' AND date_time LIKE '2022010715%' ")
    rows = cursor.fetchall()
    for row in rows:
        print(row)
con.close()
```

```
('202201071500', 398.88, 1595.52)
('202201071515', 400.5, 1602.0)
('202201071530', 397.26, 1589.04)
('202201071545', 391.86, 1567.44)
```

# 웹스크래핑으로 데이터 수집 - 동네예보 데이터

```
[5] from urllib.request import urlopen
    from bs4 import BeautifulSoup

    # 동네예보 : 서울특별시 송파구 방이1동
    data_name = "서울특별시 송파구 방이1동"
    data_code = "1171056100"
    japi = "http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=" + data_code
    response = urlopen(japi)
    weather = BeautifulSoup(response, "html.parser")
    print(data_name, " 동네 예보")
    print("데이터 소스 :", japi)

    for data in weather.findAll('data'):
        hour = data.hour.string
        temp = data.temp.string
        reh = data.reh.string
        print("시간: %02s, 온도: %04s, 습도: %02s" % (hour, temp, reh ))
```

# 웹스크래핑으로 데이터 수집 - 동네예보 데이터

서울특별시 송파구 방이1동 동네 예보

데이터 소스 : <http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1171056100>

시간: 9, 온도: 17.0, 습도: 75

시간: 12, 온도: 20.0, 습도: 60

시간: 15, 온도: 23.0, 습도: 50

시간: 18, 온도: 20.0, 습도: 70

시간: 21, 온도: 18.0, 습도: 85

시간: 24, 온도: 17.0, 습도: 85

시간: 3, 온도: 16.0, 습도: 90

시간: 6, 온도: 16.0, 습도: 90

시간: 9, 온도: 18.0, 습도: 80

시간: 12, 온도: 22.0, 습도: 65

시간: 15, 온도: 25.0, 습도: 55

시간: 18, 온도: 24.0, 습도: 55

시간: 21, 온도: 22.0, 습도: 70

시간: 24, 온도: 20.0, 습도: 70

# 오픈 API로 데이터 수집 - 동네예보 데이터

```
[6] import requests
    from datetime import date

    sky_code = ['맑음', '구름조금', '구름많음', '흐림']
    today = date.today()
    today_str = today.strftime('%Y%m%d')

    ServiceKey = '발급 받은 인증키(Decoding) 입력'

    url = 'http://apis.data.go.kr/1360000/VilageFcstMsgService/getLandFcst'
    params = {'serviceKey' : ServiceKey , 'pageNo' : '1', 'numOfRows' : '10',
              'dataType' : 'JSON', 'regId' : '11B10101' }
    response = requests.get(url, params=params)
    print(f'응답 내용 : {response.content}\n')
```

# 오픈 API로 데이터 수집 - 동네예보 데이터

## ■ 공공데이터 포털

<https://www.data.go.kr/>

- API 신청방법 : [동네예보] 검색 -> [기상청\_동네예보 통보문 조회서비스] -> [육상예보조회] -> [활용신청]

```
[6] import requests
    from datetime import date

    sky_code = ['맑음', '구름조금', '구름많음', '흐림']
    today = date.today()
    today_str = today.strftime('%Y%m%d')

    ServiceKey = '발급 받은 인증키(Decoding) 입력'

    url = 'http://apis.data.go.kr/1360000/VilageFcstMsgService/getLandFcst'
    params = {'serviceKey' : ServiceKey , 'pageNo' : '1', 'numOfRows' : '10',
              'dataType' : 'JSON', 'regId' : '11B10101' }
    response = requests.get(url, params=params)
    print(f'응답 내용 : {response.content}\n')
```

# 오픈 API로 데이터 수집 - 동네예보 데이터

## ■ 발효번호(기상청 오픈API 활용가이드 참고)

- 0 - 오늘오전, 1 - 오늘오후, 2 - 내일오전, 3 - 내일오후

```
if response.status_code == 200:
    json_data = response.json()
    response_data = json_data['response']
    result_msg = response_data['header']['resultMsg']

    if result_msg == 'NORMAL_SERVICE':
        body_data = response_data['body']
        for d in body_data['items']['item']:
            announce_time = d['announceTime']
            print(f"발표시간 : {d['announceTime']}, 발효번호 : {d['numEf']}, 예상기온 : {d['ta']}")
    else:
        print(result_msg)
```



# 오픈 API로 데이터 수집 - 동네예보 데이터

응답 내용 : b'{"response":{"header":{"resultCode":"00","resultMsg":"NORMAL\_SERVICE"}},

발표시간 : 202204240500, 발효번호 : 0, 예상기온 : °C, 날씨 : 흐림

발표시간 : 202204240500, 발효번호 : 1, 예상기온 : 23°C, 날씨 : 구름많음

발표시간 : 202204240500, 발효번호 : 2, 예상기온 : 15°C, 날씨 : 흐림

발표시간 : 202204240500, 발효번호 : 3, 예상기온 : 27°C, 날씨 : 흐림

발표시간 : 202204240500, 발효번호 : 4, 예상기온 : 17°C, 날씨 : 흐리고 비

발표시간 : 202204240500, 발효번호 : 5, 예상기온 : 28°C, 날씨 : 맑음

발표시간 : 202204240500, 발효번호 : 6, 예상기온 : 10°C, 날씨 : 맑음

발표시간 : 202204240500, 발효번호 : 7, 예상기온 : 22°C, 날씨 : 맑음

# 에너지 데이터 수집 실습



energy\_data\_collection.ipynb

# Thank you