Ask Company

크롤링 차근차근 시작하기

데이터를 저장하는 다양한 방법

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

먼저 데이터부터 ~ → dict 목록을 list화

```
import requests
from bs4 import BeautifulSoup

headers = {
    'User-Agent': ('Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) '
                   'AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36'),
}
res = requests.get("http://www.melon.com/chart/index.htm", headers=headers)
html = res.text
soup = BeautifulSoup(html, 'html.parser')
tr_tag_list = soup.select('.d_song_list tbody tr')

song_list = []

for rank, tr_tag in enumerate(tr_tag_list, 1):
    song_tag = tr_tag.select_one('a[href*=playSong]')
    album_tag = tr_tag.select_one('.wrap_song_info a[href*=goAlbumDetail]')
    artist_tag = tr_tag.select_one('a[href*=goArtistDetail]')

    song = {
        'title': song_tag.text,
        'album': album_tag.text,
        'artist': artist_tag.text,
        'rank': rank,
    }
    song_list.append(song)

song_list
```

https://gist.github.com/allieus/9e43bc279051ac3eeeb2ba06e7fcb3de

데이터를 저장하는 다양한 방법

- 파일 : json 포맷, json lines 포맷, pickle 포맷 등
- CSV 포맷 엑셀
- 데이터베이스

파일
json, json lines, pickle

json 포맷

XML과 함께 주요 데이터 포맷

언어 독립적이며, Number/String/Boolean/Array/Object/null 타입을 지원

웹에서 Ajax 통신 시에 주로 활용

데이터 객체를 한 번에 변환 → 데이터 append 시에도 전체 변환이 필요

```
import json

# song_list 객체를 한 번에 JSON포맷 문자열로 변환
json_string = json.dumps(song_list,
                         indent=4, ensure_ascii=False)

with open("melon_chart.json", "wt", encoding="utf8") as f:
    f.write(json_string)
```

```
# JSON 파일 읽어서, 객체로 변환
with open("melon_chart.json", "rt", encoding="utf8") as f:
    json_string = f.read()
    song_list = json.loads(json_string)
```

https://www.json.org/json-ko.html

```
[
    {
        "title": "술이 달다 (Feat. 크러쉬)",
        "album": "sleepless in __________",
        "artist": "에픽하이 (EPIK HIGH)",
        "rank": 1
    },
    {
        "title": "고고베베 (gogobebe)",
        "album": "White Wind",
        "artist": "마마무(Mamamoo)",
        "rank": 2
    },
    중략
    {
        "title": "우리 그만하자",
        "album": "우리 그만하자",
        "artist": "로이킴",
        "rank": 100
    }
]
```

json lines 포맷

list형 데이터를 개행("\n")을 구분자로 하여, JSON포맷으로 표현

CSV (Comma Seperated Values) 포맷과 비교하여, json lines 포맷은 중첩된 데이터를 표현 가능

데이터 append 시에 단순히 파일 끝에 추가하면 OK → 로그성 데이터에 편리

```python
import json

with open("melon_chart.jsonl", "wt", encoding="utf8") as f:
    for song in song_list:
        json_string_for_song = json.dumps(song, ensure_ascii=False)
        f.write(json_string_for_song + '\n')
```

혹은 .jl 확장자 사용

```
{"title": "술이 달다 (Feat. 크러쉬)", "album": "sleepless in _________", "artist": "에픽하이 (EPIK HIGH)", "rank": 1}
{"title": "고고베베 (gogobebe)", "album": "White Wind", "artist": "마마무(Mamamoo)", "rank": 2}
중략
{"title": "우리 그만하자", "album": "우리 그만하자", "artist": "로이킴", "rank": 100}
```

```python
# JSON LINES 파일 읽어서, 객체로 변환
with open("melon_chart.jsonl", "rt", encoding="utf8") as f:
    song_list = []
    for line in f:
        song = json.loads(line)
        song_list.append(song)
```

http://jsonlines.org/

pickle 포맷

파이썬 고유 직렬화 포맷 → 거의 모든 파이썬 데이터 타입을 지원

파이썬 버전 별로 pickle 데이터가 상이할 수 있음에 유의

데이터 객체를 한 번에 변환 → 데이터 append 시에도 전체 변환이 필요

```
import pickle

# song_list 객체를 한 번에 pickle포맷 데이터로 변환
pickle_data = pickle.dumps(song_list)

with open("melon_chart.pickle", "wb") as f:
    f.write(pickle_data)
```

```
import pickle

# song_list 객체를 한 번에 pickle포맷 데이터로 변환
pickle_data = pickle.dumps(song_list)

with open("melon_chart.pickle", "rb") as f:
    pickle_data = f.read()
    song_list = pickle.loads(pickle_data)
```

```
b'\x80\x03]q\x00(}q\x01(X\x05\x00\
x00\x00titleq\x02X\x1f\x00\x00\x00
\xec\x88\xa0\xec\x9d\xb4
\xeb\x8b\xac\xeb\x8b\xa4 (Feat.
\xed\x81\xac\xeb\x9f\xac\xec\x89\x
ac)q\x03X\x05\x00\x00\x00albumq\x0
4X\x17\x00\x00\x00sleepless in
__________q\x05X\x06\x00\x00\x00ar
tistq\x06X\x18\x00\x00\x00\xec\x97
\x90\xed\x94\xbd\xed\x95\x98\xec\x
9d\xb4 (EPIK
HIGH)q\x07X\x04\x00\x00\x00rankq\x
08K\x01u}q\t(X\x05\x00\x00\x00titl
eq\nX\x17\x00\x00\x00\xea\xb3\xa0\
xea\xb3\xa0\xeb\xb2\xa0\xeb\xb2\xa
0
(gogobebe)q\x0bX\x05\x00\x00\x00al
bumq\x0cX\n\x00\x00\x00White
Windq\rX\x06\x00\x00
```

CSV와 엑셀

CSV/엑셀포맷을 지원하는 다양한 라이브러리

- pyopenxl
- xlrd / xlwt
- pandas : DataFrame객체에 대해 csv/excel/pickle/json/sql/dict 포맷으로의 저장을 지원. excel 포맷의 경우 내부적으로 xlrd/xlwt 활용
- pyexcel
- xlwings

https://github.com/pyexcel/pyexcel

pyexcel 활용하여 엑셀로 저장

설치 : pip install pyexcel

- 별도 팩키지를 통한 다양한 포맷 지원
  - pyexcel-io (csv, csvz, tsv, tsvz), pyexel-xls (xls, xlsx, xlsm), pyexcel-xlsx, pyexcel-ods3, pyexcel-ods

```
import pyexcel

pyexcel.save_as(records=song_list, dest_file_name="melon_by_pyexcel.xls")
```

pandas

설치 : pip install pandas

```
import pandas as pd

df = pd.DataFrame(song_list)
df.to_excel('melon_chart.xlsx')
df.to_csv('melon_chart.csv')
df.to_pickle('melon_chart.pickle')
```

```
df = pd.read_excel('melon_chart.xlsx')
df = pd.read_csv('melon_chart.csv')
df = pd.read_pickle('melon_chart.pickle')
```

데이터베이스

다양한 데이터베이스

- RDB (관계형 데이터베이스) : SQLite, PostgreSQL, MySQL, Oracle, MSSQL 등
- NoSQL : MongoDB, CouchDB, HBase, Redis

SQLite

- 파일 데이터베이스
- Single User 전용
  - 멀티 프로세스/쓰레드에 의해 다중 처리 시에 Database Lock 등의 오류 발생의 여지가 있습니다.

파이썬3 기본에서의 sqlite3 지원

```
import sqlite3

conn = sqlite3.connect('example.db')

c = conn.cursor()

# 초기에 테이블 생성
c.execute('''CREATE TABLE stocks
             (date text, trans text, symbol text, qty real, price real)''')

# 데이터 1 Row 추가
c.execute("INSERT INTO stocks VALUES ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)")

# 변화를 저장
conn.commit()

# 연결 닫기
conn.close()
```

https://docs.python.org/ko/3/library/sqlite3.html

기본 Context Manager를 통한 접근

```
import sqlite3

with sqlite3.connect('example1.db') as conn:
    cursor = conn.cursor()

    cursor.execute('''CREATE TABLE stocks
                      (date text, trans text, symbol text, qty real, price real)''')

    cursor.execute('''INSERT INTO stocks VALUES
                      ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)''')
```

예외가 발생하면 conn.rollback() 수행
그렇지 않으면 conn.commit() 수행

커스텀 context manager를 통한 구현

```
import sqlite3
from contextlib import contextmanager

@contextmanager
def sqlite_cursor(db_path):
    conn = sqlite3.connect(db_path)
    conn.set_trace_callback(print) # 실행된 SQL 내역 출력
    cursor = conn.cursor()
    try:
        yield cursor
    except Exception as e:
        conn.rollback()
        raise
    else:
        conn.commit()
    conn.close()
```

활용 예

```
with sqlite_cursor('example.db') as cursor:
    cursor.execute('CREATE TABLE stocks (date text, trans text, symbol text, qty real, price real)')
    cursor.execute("INSERT INTO stocks VALUES ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)")
```

executemany

```
with sqlite_cursor('example.db') as cursor:
    purchases = [
        ('2006-03-28', 'BUY', 'IBM', 1000, 45.00),
        ('2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
        ('2006-04-06', 'SELL', 'IBM', 500, 53.00),
    ]
    # ? Placeholder
    cursor.executemany('INSERT INTO stocks VALUES (?, ?, ?, ?, ?)', purchases)
```

조건 조회 – Parameterized Query (로그인을 처리하는 SQL)

```
with sqlite_cursor('example.db') as cursor:
    username = "' OR 1=1 --" # 위와 동일
    password = ''

    sql = "SELECT * FROM users WHERE username='%s' AND password='%s'"
    cursor.execute(sql % (username, password))
    print(cursor.fetchone())

    # ? Placeholder
    sql = "SELECT * FROM users WHERE username=? AND password=?"
    cursor.execute(sql, (username, password))
    print(cursor.fetchone())

    # Named Placeholder
    sql = "SELECT * FROM users WHERE username=:username AND password=:password"
    cursor.execute(sql, {'username': username, 'password': password})
    print(cursor.fetchone())
```

SQL Injection 공격에 취약

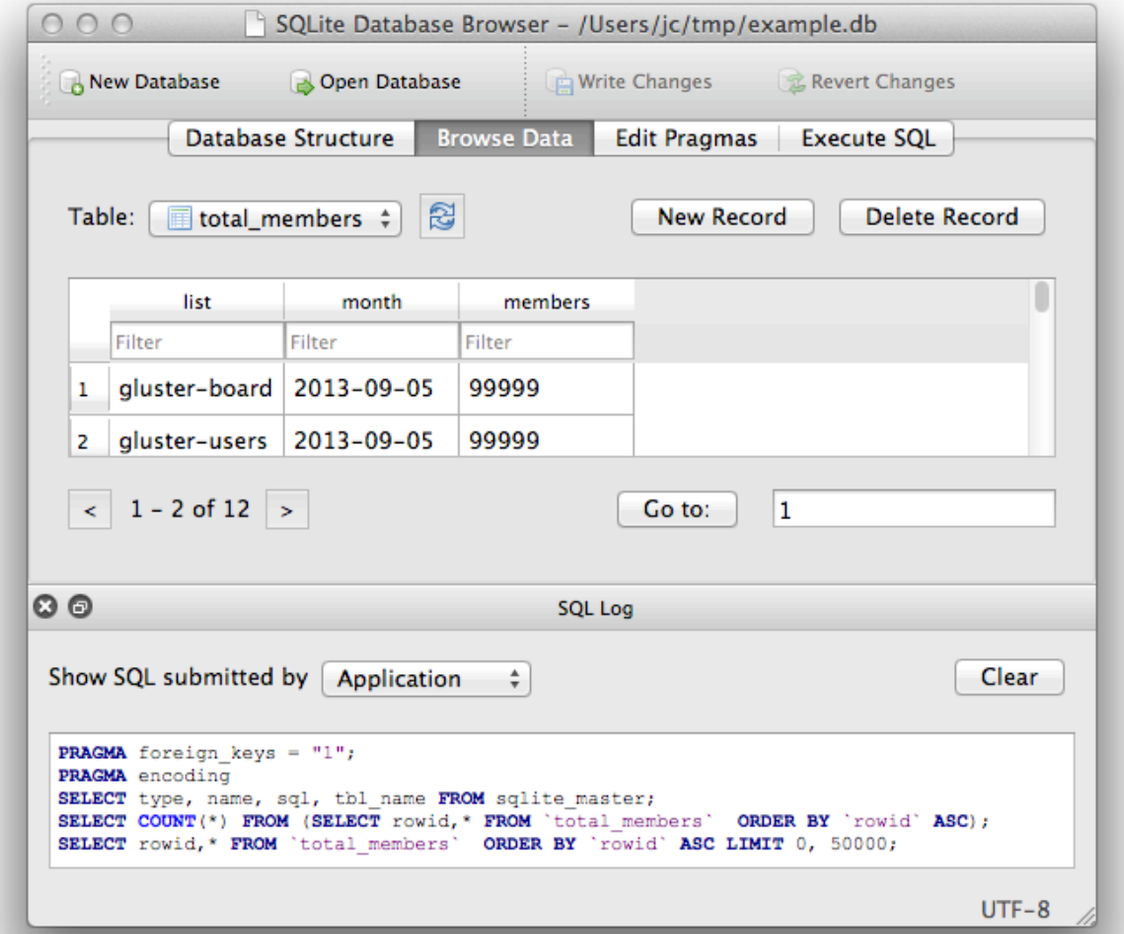
```
SELECT * FROM users WHERE username='' OR 1=1 --' AND password=''
('admin', 'adminpw')
```

SQL Injection 공격에 대응

```
SELECT * FROM users WHERE username=''' OR 1=1 --' AND password=''
None

SELECT * FROM users WHERE username=''' OR 1=1 --' AND password=''
None
```

DB Browser for SQLite 프로그램



https://sqlitebrowser.org/

다양한 ORM

- Django ORM : 풀스택 웹프레임워크에 포함된 ORM
- SQLAlchemy : 데이터베이스 툴킷
- dataset : databases for lazy people

https://github.com/vinta/awesome-python#orm

dataset

- 설치 : pip install dataset
- 특징
  - Automatic schema 지원 : 테이블/컬럼이 없을 때, 자동으로 생성
  - Upserts 지원
  - 쉬운 Query 헬퍼 지원
  - 호환성 : SQLAlchemy 기반으로 구현되었기에, 주요 DB들을 지원

https://dataset.readthedocs.io

dataset 예시 (1)

```
import dataset

db = dataset.connect('sqlite:///mydatabase.db')

table = db['sometable']
table.insert(dict(name='John Doe', age=37))
table.insert(dict(name='Jane Doe', age=34, gender='female'))


john = table.find_one(name='John Doe')

for row in table.all():
print(row)


# SQLite 데이터베이스와 연결 시도
db = dataset.connect('sqlite:///mydatabase.db')

# MySQL 데이터베이스와 연결 시도
db = dataset.connect('mysql://user:password@localhost/mydatabase')

# PostgreSQL 데이터베이스와 연결 시도
db =
dataset.connect('postgresql://scott:tiger@localhost:5432/mydatabase')
```

dataset을 쓰지 않았을 경우

```
import sqlite3
conn = sqlite3.connect(':memory:')
c = conn.cursor()

c.execute('CREATE TABLE IF NOT EXISTS sometable (name, age INTEGER)')
conn.commit()

c.execute('INSERT INTO sometable values (?, ?) ', ('John Doe', 37))
conn.commit()

c.execute('ALTER TABLE sometable ADD COLUMN gender TEXT')
conn.commit()

c.execute('INSERT INTO sometable values (?, ?, ?) ', ('Jane Doe', 34, 'female'))
conn.commit()

c.execute('SELECT name, age FROM sometable WHERE name = ?', ('John Doe', ))
row = list(c)[0]
john = dict(name=row[0], age=row[1])
```

https://dataset.readthedocs.io/en/latest/quickstart.html

dataset 예시 (2)

```
import dataset

db = dataset.connect('sqlite:///melon.db')

table = db['chart']

for song in song_list:
    table.insert(song)
```

jsonl로 먼저 정리 후에 모아서 RDB로 보내기

크롤링 후에 저장해야할 양이 방대할 때, 바로 RDB로 저장하기보다

로컬 파일(jsonl 등)에 저장한 후에, 모아서 RDB(로컬 or 외부)로 보낼 수 있습니다.

```
# 크롤링한 결과를 지정 파일에 계속 누적
with open("melon_chart_20190301.jl", "at", encoding="utf8") as f:
    for song in get_melon_chart():
        json_string = json.dumps(song, ensure_ascii=False)
        f.write(json_string + '\n')
```

```
# 혹은 MySQL, PostgreSQL 데이터베이스로의 INSERT
db = dataset.connect('sqlite:///melon.db')

# 일정 주기 단위로 모아서, 한 방에 DB로 저장
with open("melon_chart_20190301.jl", "rt", encoding="utf8") as f:
    # 지정 파일내 모든 내역을 한 번에 리스트화 (실행시간/메모리 이슈 등)
    song_list = [json.loads(line) for line in f if line]

    table = db['chart']
    table.insert_many(song_list, chunk_size=1000)
```

디폴트 : 1000

https://dataset.readthedocs.io/en/latest/api.html#dataset.Table.insert_many

개선) 조금씩 읽어가면서 DB에 저장하기

```
def chunk_list(iterable, chunk_size):
    chunk = []
    for el in iterable:
        chunk.append(el)
        if len(chunk) == chunk_size:
            yield chunk
            chunk = []
    if chunk:
        yield chunk

# 일정 주기 단위로 모아서, 한 방에 DB로 저장
with open("melon_chart_20190301.jl", "rt", encoding="utf8") as f:
    table = db['chart']

    for line_list in chunk_list(f, 1000):
        song_list = [json.loads(line) for line in line_list]
        table.insert_many(song_list, chunk_size=1000)
```

1000줄 이상의 데이터일 때, 유의미

각 클라우드 벤더 서비스 활용을 해볼 수도 있습니다.

- 벤더 별 NoSQL 데이터베이스와 정적 스토리지
  - AWS : DynamoDB, Simple Storage Service ← AWS Lambda
  - Microsoft Azure : Azure Table, Azure Storage ← Azure Function
  - Google Cloud : Datastore, Cloud Storage ← Google Cloud Function

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

- Ask Company