

LESSON

csv, json 파일 다루기

CSV(comma separated values) 파일

- 데이터 요소를 콤마(,)로 구분하여 저장한 데이터.
- 글꼴 같은 서식 정보가 없는 텍스트 파일.
- 메모장으로 열어서 확인할 수 있고 가공하기 좋은 형태.

CSV

이름,직업,주소
김사랑,작가,서울
조우리,변리사,울산
한아름,기자,광주

DB

이름	직업	주소
김사랑	작가	서울
조우리	변리사	울산
한아름	기자	광주

- 모듈 import
import csv
- csv 모듈로 csv 파일 읽기 함수
csv.reader() : 읽은 자료를 리스트로 저장.
csv.DictReader() : 읽은 자료를 딕셔너리로 저장

```
import csv
```

```
with open('차량관리.csv','r',encoding='utf-8') as file:  
    csv_reader=csv.reader(file)  
    for line in csv_reader:  
        print(line)
```

```
['1', '00가1234', '2020-10-01, 14:00']  
['2', '09다5666', '2020-10-01, 14:10']  
['3', '03가9999', '2020-10-01, 14:20']
```

차량관리.csv 파일 읽기(DictReader)

➤ csv 모듈로 csv 파일 읽기 함수

csv.reader() : 읽은 자료를 리스트로 읽기.

csv.DictReader() : 읽은 자료를 딕셔너리로 읽기.

```
import csv

with open('차량관리.csv', 'r', encoding='utf-8') as file:
    csv_reader = csv.DictReader(file)
    for line in csv_reader:
        print(line)
```

```
{ '순번': '1', '차량번호': '00가1234', '입차일시': '2020-10-01, 14:00' }
{ '순번': '2', '차량번호': '09마5666', '입차일시': '2020-10-01, 14:10' }
{ '순번': '3', '차량번호': '03가9999', '입차일시': '2020-10-01, 14:20' }
```

차량관리.csv 파일 생성하기

➤ csv 모듈로 csv파일 생성하는 순서

- ① import csv
- ② csv.writer(파일 객체)
- ③ csv파일 객체에 csv.writerow(저장할 내용) or csv.writerows(저장할 내용)



writerow() : 한 행씩 파일에 쓰기
writerows() : 여러 행을 한 번에 쓰기

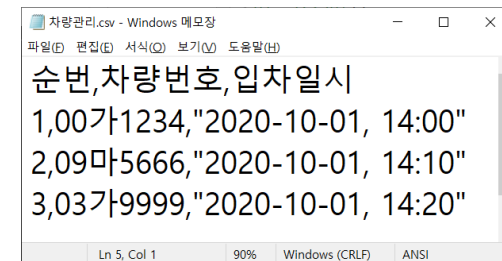
```
import csv

with open('차량관리.csv', 'w', newline='', encoding='utf-8') as file:
    csv_maker = csv.writer(file)
    csv_maker.writerow(['순번', '차량번호', '입차일시'])
    csv_maker.writerow([1, '00가1234', '2020-10-01, 14:00'])
    csv_maker.writerow([2, '09마5666', '2020-10-01, 14:10'])
    csv_maker.writerow([3, '03가9999', '2020-10-01, 14:20'])

print('파일이 생성되었습니다.')
```



writerow, writerows는 빈 줄이 자동
추가된다. 새로운 줄이 추가되지 않도록
하려면 newline 옵션을 설정해야 한다.



차량관리.csv 파일 생성하기(DictWriter)

```
import csv
field_names=['순번','차량번호','입차일시']
data=[
    {'순번': '1', '차량번호': '00가1234', '입차일시': '2020-10-01, 14:00'},
    {'순번': '2', '차량번호': '09마5666', '입차일시': '2020-10-01, 14:10'},
    {'순번': '3', '차량번호': '03가9999', '입차일시': '2020-10-01, 14:20'}
]
f=open('차량관리2.csv','w',newline='',encoding='utf-8')
csv_writer=csv.DictWriter(f,fieldnames=field_names)
csv_writer.writeheader()
csv_writer.writerows(data)
# for row in data:
#     csv_writer.writerow(row)
f.close()
print('차량관리2.csv 생성 완료!')
```

- 공공데이터 취득

- 공공데이터 포털

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

- 기상청 기상자료개방포털

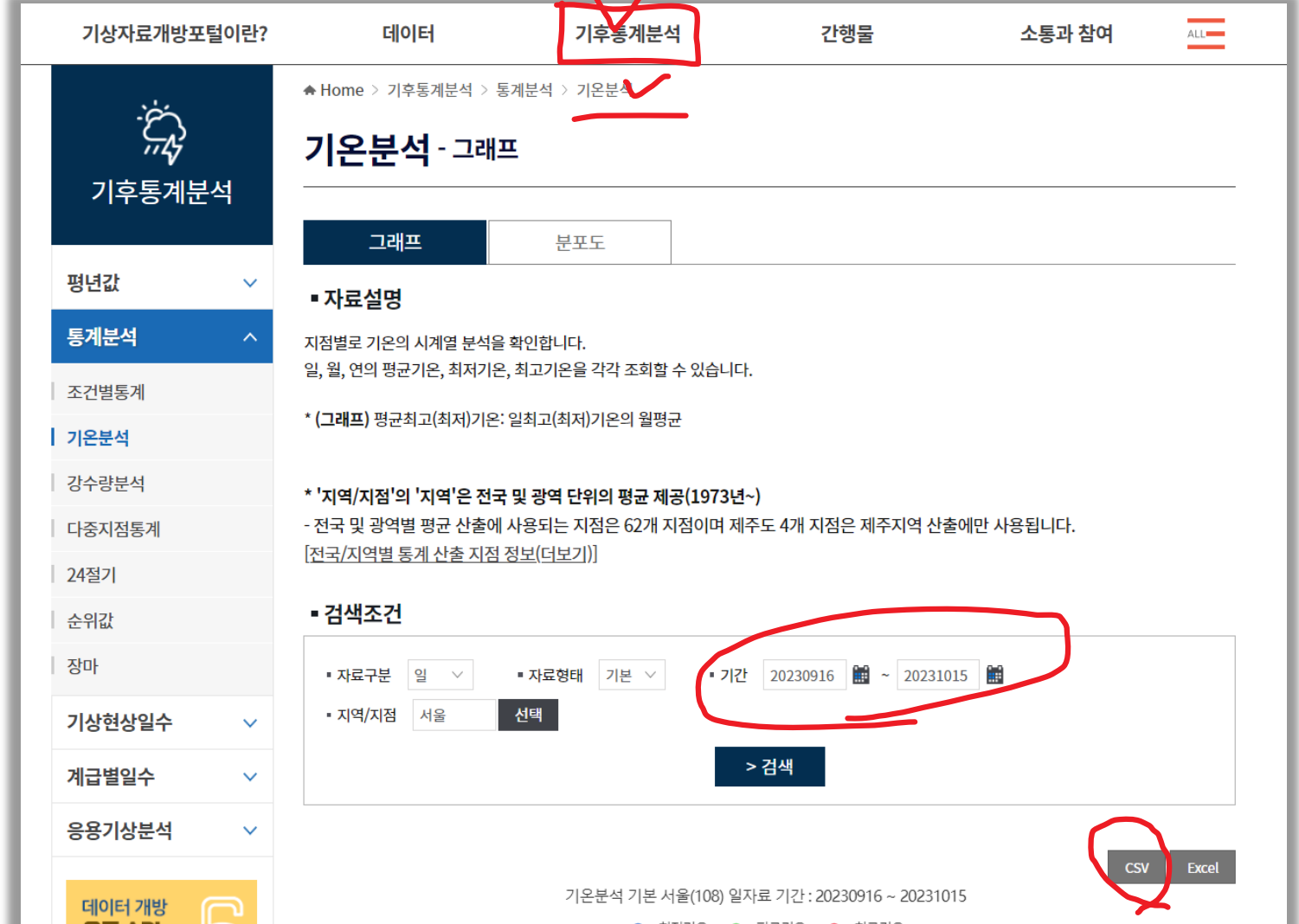
- <https://data.kma.go.kr/cmmn/main.do>

- 행정안전부 주민등록 인구통계

- <https://jumin.mois.go.kr/>

<https://data.kma.go.kr/stcs/grnd/grndTaList.do?pgmNo=70>

1. 기후통계분석에서 "기온분석"을 선택한다.
2. 지역/지점에서 "서울"을 선택한다.
3. 기간에서 시작일과 종료일을 8자리로 입력하고 "검색"을 클릭한다.
4. 검색조건 아래에서 "CSV"를 클릭하면 파일이 다운로드된다.



The screenshot shows the KMA Climate Data Portal interface. The top navigation bar includes '기상자료개방포털이란?', '데이터', '기후통계분석' (highlighted with a red box and checkmark), '간행물', and '소통과 참여'. The left sidebar contains a '기후통계분석' section with a dropdown menu where '통계분석' is selected. The main content area is titled '기온분석 - 그래프' and includes a '자료설명' section with instructions on how to use the data. The '검색조건' (Search Conditions) section is highlighted with a red box and contains the following fields: '자료구분' (Data Category) set to '일' (Daily), '자료형태' (Data Format) set to '기본' (Basic), '기간' (Period) set to '20230916 ~ 20231015', and '지역/지점' (Region/Point) set to '서울' (Seoul). A red box also highlights the '기간' field. At the bottom right, the 'CSV' button is highlighted with a red box, indicating the download option.


```
import csv
data=csv.reader(open('seoultemp.csv'))
# 가공한 데이터를 저장할 리스트 변수
data_list=[]
for row in data:
    # 데이터가 입력되어 있으며 \t만 입력된 게 아닌 경우
    if row and row[0]!='\t':
        data_list.append(row)

# 데이터 위쪽 불필요한 행을 제외한 데이터만 새로운 리스트에 저장
data_list=data_list[7:]

# 가공한 데이터를 csv파일로 저장
import csv
with open("after_seoultemp.csv","w",newline='',encoding="utf-8") as f:
    csv_writer=csv.writer(f)
    csv_writer.writerows(data_list)
```

파일명 : seoultemp.csv

	A	B	C	D	E
1	기온분석				
2	[검색조건]				
3	자료구분 : 일				
4	자료형태 : 기본				
5	지역/지점 : 서울				
6	기간 : 19070701~20231015				
7					
8	날짜	지점	평균기온(°C)	최저기온(°C)	최고기온(°C)
9	1907-10-01	108	13.5	7.9	20.7
10	1907-10-02	108	16.2	7.9	22
11	1907-10-03	108	16.2	13.1	21.3
12	1907-10-04	108	16.5	11.2	22
13	1907-10-05	108	17.6	10.9	25.4

```
import csv
data=csv.reader(open('after_seoultemp.csv'))
```



다운 받은 파일을 열어서 제목행 위까지 삭제하고 작업해야 문제가 없다.

```
for row in data:
    print(row[0])    # 날짜 데이터 선택
```

비어있는 데이터만 확인

```
for row in data:
    if row[-1]=='':
        print(row)
```

특정 날짜의 데이터만 확인

```
for row in data:
    if '02-28' in row[0]:
        print(row)
```

파일명 : seoultemp.csv

	A	B	C	D	E
1	기온분석				
2	[검색조건]				
3	자료구분 : 일				
4	자료형태 : 기본				
5	지역/지점 : 서울				
6	기간 : 19070701~20231015				
7					
8	날짜	지점	평균기온(°C)	최저기온(°C)	최고기온(°C)
9	1907-10-01	108	13.5	7.9	20.7
10	1907-10-02	108	16.2	7.9	22
11	1907-10-03	108	16.2	13.1	21.3
12	1907-10-04	108	16.5	11.2	22
13	1907-10-05	108	17.6	10.9	25.4

```
import csv

data=csv.reader(open('after_seoultemp.csv'))
# next(data)
max=-1000
date=''
for row in data:
    if row[-1]!='':
        if max < float(row[-1]): # csv파일을 읽어오면 요소들은 문자열이므로 숫자로 변환
            max=float(row[-1])
            date=row[0]
print(f'가장 더웠던 날짜:{date}, 최고 기온:{max}')
```

가장 더웠던 날짜:2018-08-01, 최고 기온:39.6

```
import csv
data=csv.reader(open('after_seoultmp.csv'))
max=-1000
min=1000
max_date=''
min_date=''

for row in data:
    if row[-1]!='':
        if float(row[-1])>max:
            max=float(row[-1])
            max_date=row[0].strip('\t')
    if row[-2] != '':
        if float(row[-2])<min:
            min=float(row[-2])
            min_date=row[0].strip('\t')

print(f"가장 더웠던 날짜:{max_date}, 최고 기온:{max}")
print(f"가장 추웠던 날짜:{min_date}, 최저 기온:{min}")
```

가장 더웠던 날짜:2018-08-01, 최고 기온:39.6
가장 추웠던 날짜:1927-12-31, 최저 기온:-23.1

sorted() 함수로 리스트 특정 요소 기준으로 정렬하기

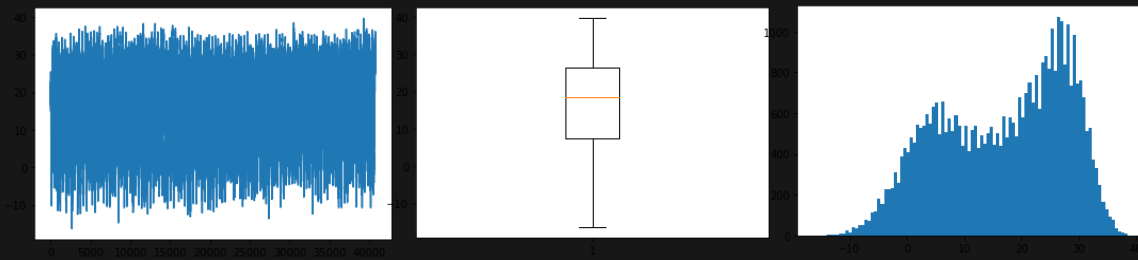
```
# 역대 더위순 3개 추출
import csv

temps=[]
with open("after_seoul.csv","r",encoding="utf-8") as f:
    data=csv.reader(f)
    for row in data:
        if row[-1]!='':
            temps.append((row[0].strip(),float(row[-1])))

# sorted()함수로 최고기온으로 내림차순 정렬시키기
sorted_data=sorted(temps,key=lambda x:x[1],reverse=True)
for i in range(3):
    print(f"{i+1}위: {sorted_data[i][1]}, {sorted_data[i][0]}")
```

1위: 39.6, 2018-08-01
2위: 38.4, 1994-07-24
3위: 38.3, 2018-07-31

```
import csv
data=csv.reader(open('seoultemp_가공후.csv'))
next(data)
temp=[]
for row in data:
    if row[-1]!='':
        temp.append(float(row[-1]))
# temp=[float(row[-1]) for row in data if row[-1]!='']
# 최고 기온 데이터 시각화하기
import matplotlib.pyplot as plt
plt.plot(temp)
plt.show()
plt.boxplot(temp)
plt.show()
plt.hist(temp,bins=100)
plt.show()
```



- 네트워크 상에서 주고받는 데이터 표준 텍스트 파일.
- json데이터의 형식
 - 속성-값(attribute-value) 쌍으로 구성된 데이터.
 - {속성:값,속성:값,...} 로 딕셔너리와 같은 형태.



JSON은 키나 값에 문자열, 숫자, 불리언, 배열(리스트), 객체(딕셔너리)만 허용된다

```
{  
    '회원명':'조나단',  
    '연락처':'1234',  
    '주소':'seoul'  
}
```

- 파이썬의 내장 모듈인 json의 기능을 향상시킨 확장 버전이다.
- json 보다 유연하고 속도가 빠르다. 대량의 데이터인 경우 simplejson을 사용하는 것이 좋다.

```
pip install simplejson
```

```
import simplejson as json
```

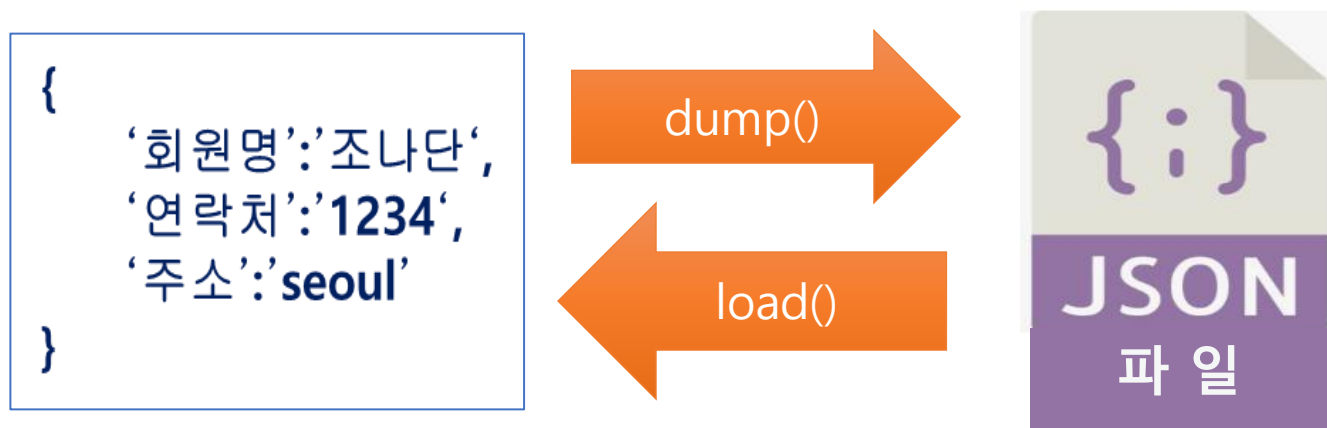
json: w3schools.com/js/js_json_syntax.asp

simplejson: simplejson.readthedocs.io/en/latest

➤ json 파일로 내보내기 및 가져오기

json.dump() : 파이썬 객체를 json 파일로 내보내기.

json.load() : json 파일을 파이썬 객체로 가져오기.



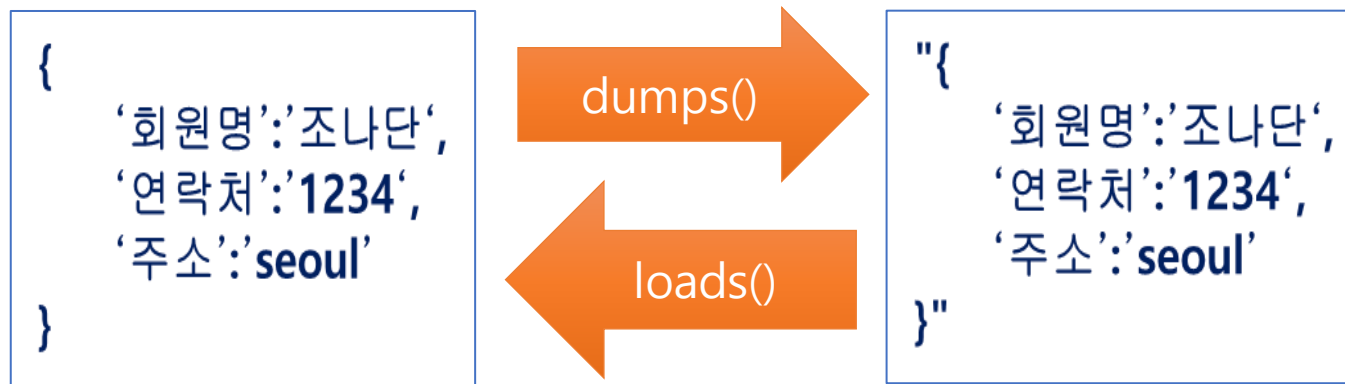
➤ json -> str, str -> json

json.dumps() : 파이썬 객체를 json 문자열로 변경.

json(dict) -----> str

json.loads() : json 문자열을 파이썬 객체로 변경

str -----> json(dict)



json.dump(내용, 파일)

```
import simplejson as json

d={'id':1,'name':'hans','hobby':['game','watch
tv','travel'],'test':True}
# print(d)

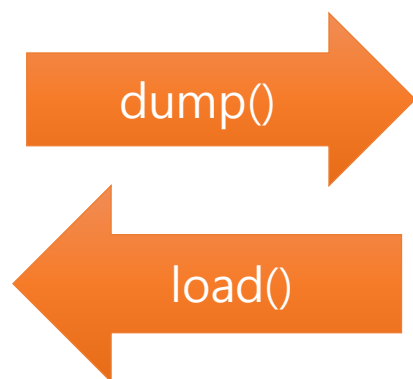
with open('people.json','w',encoding='utf8') as f:
    json.dump(d,f)
```

json.load(파일)

```
import simplejson as json

with open('people.json','r',encoding='utf8') as f:
    data=json.load(f)
    print(data)
```

```
{
    '회원명':'조나단',
    '연락처':'1234',
    '주소':'seoul'
}
```



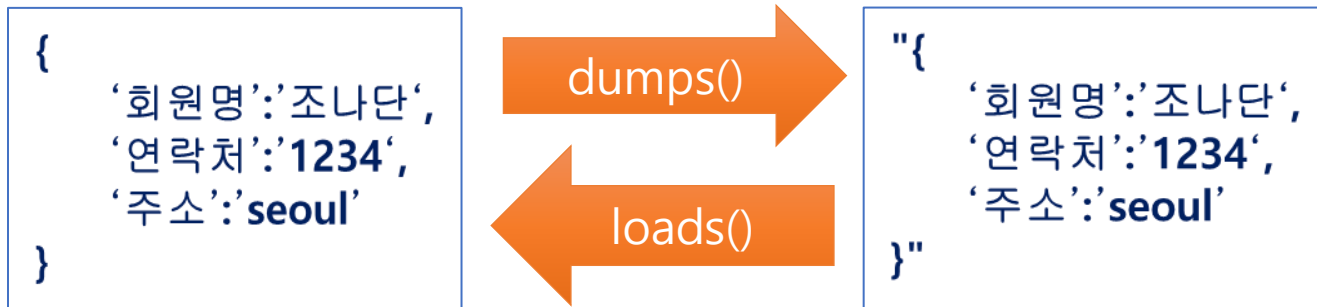
예제) json.dumps(내용)

```
import simplejson as json

dict_list=[{'name':'elsa','age':22,'spec':[170,65]},
            {'name':'anna','age':21,'spec':[172,63]]

string=json.dumps(dict_list)
print(type(string))

string=json.dumps(dict_list,indent=4)
with open('frozenStaff.json','w') as f:
    f.write(string)
```



[결과]

<class 'str'>

```
{ } frozenStaff.json > ...
1  [
2      {
3          "name": "elsa",
4          "age": 22,
5          "spec": [
6              170,
7              65
8          ]
9      },
10     {
11         "name": "anna",
12         "age": 21,
13         "spec": [
14             172,
15             63
16         ]
17     }
18 ]
```