



## ch.8 흘러가는 데이터

8.1~8.2 부분만

### 파일 입출력

- 파일로부터 데이터를 읽어 메모리에 적재, 메모리에 파일로 데이터를 쓴다 (read, write)
- c언어 파일입출력과 매우 똑같다 (file open할 때 r,w,x,a mode을 명시해주는 거)

file open할때 mode

mode의 첫번째 글자	mode의 두번째 글자(파일 타입)	Files	Column	Column 1
<u>r</u> : 파일 읽기	t(아무것도 안써도 이거) :텍스트 타입			
<u>w</u> : 파일 쓰기(존재하지 않으면 파일 생성)	b : 이진타입			
<u>x</u> : 파일 쓰기(존재하지 않을 경우에만 해당)				
<u>a</u> :파일 추가(존재하면 파일의 끝에서부터 씀)				

### write(): 텍스트 파일에 쓰기

```
f = open("../textfile.txt", 'w') #절대경로, 상대경로 다 가능 f = open("/Users/shinhaeran/Desktop/myProject/code/text.txt", 'w')
for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```

▼ print도 가능

```
f = open("/Users/shinhaeran/Desktop/myProject/code/text.txt", 'w')
for i in range(1, 11):
    data = "뽀양"
    print(data,file=f) #대신 줄바꿈 자동 추가 write처럼 하고싶으면 seq='',end='' 추가
f.close()
```

### read(), readline(), readlines(): 텍스트 파일 읽기

```
f = open("../textfile.txt", 'w')
while True:
    line = f.readline() #한줄씩 읽어옴
    if not line: break
    print(line)
f.close()

-----아니면
for line in f:
    print(line) #도 똑같다
```

```
f = open("../textfile.txt", 'w')
lines = f.readlines() #한번에 모든 줄을 읽어옴 -> 반환값은 리스트
for line in lines:
```

```
print(line)
f.close()
```

```
f = open("./textfile.txt", 'w')
data = f.read() #한번에 모든 문자열을 가져옴 -> 반환값은 문자열, 안에 숫자 넣으면 읽어들이는 글자수 제한, 없으면 '' 반환
print(data)
f.close()
```

## binary file 0~255byte

- write(): 모드에 b 포함 : 문자열 대신 바이트를 읽고 쓸 수 있다
- read(): 마찬가지로 모드에 b 포함

## 기타 파일입출력 메서드

- 자동으로 파일 닫기: with : 컨텍스트 매니저 코드 블록의 코드 한줄이 실행되고 나서 자동으로 파일을 닫아줌

```
with open("./textfile.txt", 'w') as fout:
    fout.write('얏')
```

- 파일 위치 찾기/변경: seek() & 작업위치 확인: tell()

파일 객체의 위치를 바꾸려면, `f.seek(offset, from_what)` 를 사용합니다. 위치는 기준점에 *offset* 을 더해서 계산됩니다; 기준점은 *from\_what* 인자로 선택합니다. *from\_what* 값이 0이면 파일의 처음부터 측정하고, 1이면 현재 파일 위치를 사용하고, 2 는 파일의 끝을 기준으로 사용합니다. *from\_what* 은 생략될 수 있고, 기본값은 0이라서 파일의 처음을 기준으로 사용합니다.

```
>>> f = open('workfile', 'rb+')
>>> f.write(b'0123456789abcdef')
16
>>> f.seek(5)          # Go to the 6th byte in the file
5
>>> f.read(1)
b'5'
>>> f.seek(-3, 2)      # Go to the 3rd byte before the end
13
>>> f.read(1)
b'd'
```

```
fname = input("파일명:")
try:
    fs = open(fname, "rb")
    data = fs.read()
    print("파일 데이터:", data)
    fs.seek(0, 2) #파일 끝에서 0바이트 이동(파일 끝으로 이동)
    fsize = fs.tell()
    print("파일 크기:", fsize, "bytes")
    pflag = True
    for i in range(0, fsize):
        fs.seek(i) #파일 시작에서 i바이트 이동
        bd = fs.read(1)
        fs.seek(-(i+1), 2) #파일 끝에서 (i+1)바이트 이전으로 이동
        ad = fs.read(1)
        if(bd != ad):
            pflag = False
            break
    fs.close()

    if pflag:
        print("회문입니다.")
    else:
        print("회문이 아닙니다.")
```

```
except:
    print("예외가 발생하였습니다.")
```

▼ 이진파일에서 위치를 이동할때 유용하다는건 알겠고 텍스트파일에서도 똑같이 적용?

↳ ↳ 아스키코드가 아니라면 2바이트니까 오프셋 계산하기 힘들.

## csv file serving

- 수동으로 csv파일을 한번에 한 라인씩 읽어 콤마로 구분된 필드를 분리할 수 있다.
- 형식은이렇게 ,로 컬럼을 구분. row는 \n로 구분. 어떤 것은 |나 \t으로 구분



```
#views.py
def status_information(request):

    return render(request, 'view_table/status_information.html')

def upload_csv(request):
    data = {}
    count = 0
    if "GET" == request.method:
        return render(request, "view_table/data_collection.html", data)
    # if not GET, then proceed
    try:
        csv_file = request.FILES["csv_file"]
        if not csv_file.name.endswith('.csv'):
            error(request, 'File is not CSV type')
            return HttpResponseRedirect(reverse("view_table:home"))
        #if file is too large, return
        if csv_file.multiple_chunks():
            error(request, "Uploaded file is too big (%.2f MB)." % (csv_file.size/(1000*1000)))
            return HttpResponseRedirect(reverse("view_table:query_input"))

        file_data = csv_file.read().decode("utf-8")

        lines = file_data.split("\n")
        print(lines)
        lines.remove('')

        #loop over the lines and save them in db. If error , store as string and then display
        for line in lines:
            fields = line.split(",")
            data_dict = {}
            data_dict["name"] = fields[0]
            data_dict["start_date_time"] = fields[1]
            data_dict["end_date_time"] = fields[2]
            print(fields[0])# data_dict["notes"] = fields[3]
            count+=1
        return render(request, 'view_table/status_information.html', {'count':count-1})
        # try:
        #     form = EventsForm(data_dict)
        #     if form.is_valid():
        #         form.save()
        #     else:
        #         logging.getLogger("error_logger").error(form.errors.as_json())
        # except Exception as e:
        #     logging.getLogger("error_logger").error(repr(e))
        #     pass

    except Exception as e:
        logging.getLogger("error_logger").error("Unable to upload file. "+repr(e))
        error(request, "Unable to upload file. "+repr(e))

    return HttpResponseRedirect(reverse("view_table:query_input"))
```

```
#data_collection
<form action="{% url 'view_table:upload_csv' %}" method="POST" enctype="multipart/form-data" class="form-horizontal">
    {% csrf_token %}
    <div class="form-group">
        <label for="name" class="col-md-3 col-sm-3 col-xs-12 control-label">File: </label>
        <div class="col-md-8">
            <input type="file" name="csv_file" id="csv_file" required="True" class="form-control">
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-3 col-sm-3 col-xs-12 col-md-offset-3" style="margin-bottom:10px;">
            <button class="btn btn-primary"> <span class="glyphicon glyphicon-upload" style="margin-right:5px;">
        </div>
    </div>
</form>
```

### 03 진행 사항-UI

Data collect : csv file serving

**Data Collection**

사용자는 자율기계학습 수행을 위해 직간접적으로 데이터를 수집하여 데이터베이스에 입력한다. 사용자는 직접 데이터를 입력할 수도 있고, 소셜 미디어에서 실시간 데이터를 크롤링하여 수집할 수도 있다.

[View details »](#)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	id	address	addressAge	birthday	email	gender	job	loginid	nickname	password	mgDate	idType	
2	Objectid	SakU1501	대전광역시 동서대로	2008-07-10	07715.000	user1@bnu.ac.kr	남자	교수	09707.05	44.4442	2018-03-12	1234	01707.05
3	Objectid	SakU1501	대전광역시 서구	1923	000002	ec.com					44.4442		
4	Objectid	SakU1504	대전광역시 갈매로	2004-07-14	18715.000	user1@bnu.ac.kr	남자	의사	09707.05	48.1122	2018-03-12	1234	01707.05
5	Objectid	SakU1501	대전광역시 서구	160	000002	ec.com					48.1122		
6	Objectid	SakU1504	대전광역시 갈매로	2003-03-11	18715.000	user1@bnu.ac.kr	남자	의사	09707.05	48.1122	2018-03-12	1234	01707.05
7	Objectid	SakU1501	대전광역시 서구	230	000002	ec.com					50.7762		
8	Objectid	SakU1504	대전광역시 과당로14	2009-05-18	27715.000	user1@bnu.ac.kr	남자	의사	09707.05	49.2802	2018-03-12	1234	01707.05
9	Objectid	SakU1501	대전광역시 서구	연립 14	000002	ec.com					50.5362		
10	Objectid	SakU1504	대전광역시 동북로	2002-07-17	20715.000	user1@bnu.ac.kr	남자	의사	09707.05	49.2802	2018-03-12	1234	01707.05
11	Objectid	SakU1501	대전광역시 서구	219연립	000002	ec.com					49.2802		
12	Objectid	SakU1504	대전광역시 신당로	2003-11-16	24715.000	user1@bnu.ac.kr	남자	의사	09707.05	53.0202	2018-03-12	1234	01707.05
13	Objectid	SakU1501	대전광역시 서구	141연립	000002	ec.com					53.0202		
14	Objectid	SakU1504	대전광역시 신당로	2002-11-19	24715.000	admin@bnu.ac.kr	남자	의사	09707.05	53.0202	2018-03-12	1234	01707.05
15	Objectid	SakU1501	대전광역시 서구	102	000002	ec.com					51.7862		
16	Objectid	SakU1504	대전광역시 신당로	2008-07-10	11115.000	user12@bnu.ac.kr	남자	의사	09707.05	55.4352	2018-03-12	1234	01707.05
17	Objectid	SakU1501	대전광역시 서구	358	000002	user.com					55.4352		

### 03 진행 사항-UI

Data collect : csv file serving

**Manually Data Feeding**

Input Data

File:  [파일 선택](#) [test.csv](#)

[Upload](#)

**LOG status**

more success: 12 data to MariaDB

Status MariaDB

Running [SEARCH](#)

	address	addressAge	age	birthday	email	gender	job	loginid	nickname	password	mgDate	idType
SakU1501	대전광역시 동서대로	2008-07-10	07715.000	user1@bnu.ac.kr	남자	교수	09707.05	44.4442	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	1923	000002	ec.com								
SakU1504	대전광역시 갈매로	2004-07-14	18715.000	user1@bnu.ac.kr	남자	의사	09707.05	48.1122	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	160	000002	ec.com								
SakU1504	대전광역시 갈매로	2003-03-11	18715.000	user1@bnu.ac.kr	남자	의사	09707.05	48.1122	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	230	000002	ec.com								
SakU1504	대전광역시 과당로14	2009-05-18	27715.000	user1@bnu.ac.kr	남자	의사	09707.05	49.2802	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	연립 14	000002	ec.com								
SakU1504	대전광역시 동북로	2002-07-17	20715.000	user1@bnu.ac.kr	남자	의사	09707.05	49.2802	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	219연립	000002	ec.com								
SakU1504	대전광역시 신당로	2003-11-16	24715.000	user1@bnu.ac.kr	남자	의사	09707.05	53.0202	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	141연립	000002	ec.com								
SakU1504	대전광역시 신당로	2002-11-19	24715.000	admin@bnu.ac.kr	남자	의사	09707.05	53.0202	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	102	000002	ec.com								
SakU1504	대전광역시 신당로	2008-07-10	11115.000	user12@bnu.ac.kr	남자	의사	09707.05	55.4352	2018-03-12	1234	01707.05	1234
SakU1501	대전광역시 서구	358	000002	user.com								

## xml

- 특정 목적에 따라 데이터를 태그를 감싸서 마크업하는 범용적인 포맷
- xml은 데이터 피드/메시지 전송(파싱) dp aksgdl tkdyd
- xml에서 파생된 데이터를 검색하고 수정할 수있는 다양한 방법을 제공.

```
<rss version="2.0">
<channel>
<title>기상청 육상 중기예보</title>
<link>
http://www.kma.go.kr/weather/forecast/mid-term_02.jsp
</link>
<description>기상청 날씨 웹서비스</description>
<language>ko</language>
<generator>기상청</generator>
<pubDate>2019년 06월 27일 (목)요일 18:00</pubDate>
<item>
<author>기상청</author>
<category>육상중기예보</category>
<title>서울, 경기도 육상 중기예보 - 2019년 06월 27일 (목)요일 18:00 발표</title>
<link>
http://www.kma.go.kr/weather/forecast/mid-term_02.jsp
</link>
<guid>
http://www.kma.go.kr/weather/forecast/mid-term_02.jsp
</guid>
<description>
<header>
<title>서울, 경기도 육상중기예보</title>
<tm>201906271800</tm>
<wf>
<![CDATA[
7월 1일은 기압골의 영향으로 비가 오겠으며, 그 밖의 날은 고기압의 가장자리에 들어 구름이 많겠습니다.<br />
/>
기온은 평년(최저기온: 19~21℃, 최고기온: 26~28℃)과 비:
]]>
</wf>
</header>
<body>
<location wl_ver="3">
<province>서울·인천·경기도</province>
<city>서울</city>
<data>
<mode>A02</mode>
<tmEf>2019-06-30 00:00</tmEf>
<wf>구름많음</wf>
<tmn>20</tmn>
<tmx>28</tmx>
<reliability>보통</reliability>
</data>
<data>
<mode>A02</mode>
<tmEf>2019-06-30 12:00</tmEf>
<wf>구름많음</wf>
<tmn>20</tmn>
<tmx>28</tmx>
<reliability>보통</reliability>
</data>
</data>
</body>
</channel>
</rss>
```

이런 xml 데이터가 있으면

```
from bs4 import BeautifulSoup
import requests
import datetime
import json

info_url = 'http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=109'
response = requests.get(info_url)
soup = BeautifulSoup(response.content, 'html.parser')

locations = soup.find_all('location')

for location in locations:
    print(location.find('city').text, ":", location.find('wf').text)

#output
```

서울 : 구름조금  
인천 : 구름조금  
수원 : 구름조금  
파주 : 구름조금  
이천 : 구름조금  
평택 : 구름조금  
백령도 : 구름조금  
과천 : 구름조금  
광명 : 구름조금  
강화 : 구름조금

## JSON (javaScript Object Notation)

- 데이터 교환하는 아주 인기있는 형식 → 웹 환경에서 서버와 클라이언트 사이에 데이터를 주고받을 때 많이 사용 → Rest API에서 많이 쓴다고 한다
- django rest framework에서 serializer : queryset과 모델 인스턴스와 같은 복잡한 데이터를 json,xml 같은 다른 콘텐츠 유형으로 쉽게 변환해준다. 유효성 검사 후 형변환할 수 있도록 serialization 제공

```
# movie_api/movies/serializers.py

from rest_framework import serializers
from .models import Movie
class MovieSerializer(serializers.ModelSerializer):
    class Meta:
        model = Movie # 모델 설정
        fields = ('id','title','genre','year') # 필드 설정
```

```
# movie_api/movies/views.py
from rest_framework import viewsets
from .serializers import MovieSerializer
from .models import Movie
class MovieViewSet(viewsets.ModelViewSet):
    queryset = Movie.objects.all()
    serializer_class = MovieSerializer
```

해주면

The default basic root view for DefaultRouter

GET /

HTTP 200 OK  
Allow: GET, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "movies": "http://localhost:8000/movies/"
}
```

Raw data HTML form

Title 어벤져스: 엔피니티 워

Genre 액션,모험,판타지

Year 2018

POST

### Movie List

OPTIONS GET

POST /movies/

HTTP 201 Created  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "id": 1,
  "title": "어벤져스: 엔피니티 워",
  "genre": "액션,모험,판타지",
  "year": 2018
}
```

이렇게 내가 적은 링크대로 get/post요청가 json형태로 보여진다~~~

참고:<https://jamanbbo.tistory.com/43>