학습 내용

2부. 프로그래밍 언어 활용

- 6장. 모듈과 패키지
- 7장. 객체지향 프로그래밍
- 8장. 예외 처리
- **9장. 파일 입/출력 프로그래밍**
  - 1. 파일에 데이터 저장하고 불러오기
  - 2. 피클을 이용한 객체 저장하고 불러오기
  - 3. CSV 형식 파일 읽기/쓰기
  - 4. JSON 데이터 저장하고 불러오기
  - 5. HDF5 파일 읽기/쓰기
- 10장. 데이터베이스 연동

파일 입출력

1절. 파일에 데이터 저장하고 불러오기

- open() : 파일 객체를 반환

- *file_pointer* = open(*file_name*, *mode*, *encoding='ASCII'*)
- 구문에서…
  - *file_pointer* : 열린 파일 객체,
    파일 객체의 readline() 또는 readlines() 함수를 이용하여 파일로부터 데이터를 읽음
    write() 함수는 데이터를 씀
  - *mode* : 파일 열기 모드를 의미
    r : 읽기 모드 - 파일을 읽기만 할 때 사용
    w : 쓰기 모드 - 파일의 내용을 쓸 때 사용
    a : 추가 모드 - 파일의 마지막에 추가할 때 사용
    b : 바이너리 모드 - 피클 등을 사용하여 저장하거나 불러올 때는 바이너리 모드로 지정해야 함
  - *encoding* : 파일의 인코딩을 지정
    UTF-8 인코딩으로 저장되어 있는 파일이라면 encoding='UTF8'을 사용

파일 입출력

1절. 파일에 데이터 저장하고 불러오기

```
1  f = open("sample.txt", "w")
```

파일을 쓰기 모드로 연다.

```
1  print(f.writable())
```

```
True
```

```
1  f.write("HelleWnWorldWn")
```

파일에 텍스트를 쓴다.

```
12
```

```
1  f.close()
```

파일을 닫아준다.

파일 입출력

1절. 파일에 데이터 저장하고 불러오기

```
1  f = open("sample.txt", "a")
```

파일을 추가 모드로 연다.

```
1  print("프린트 함수로 쓸 수 있습니다.", file=f)
```

print() 함수의 file 인수에 파일 객체를 지정하면 파일로 저장이 가능함

```
1  f.close()
```

```
1  f = open("sample.txt", "r")
```

파일을 읽기 모드로 연다.

```
1  lines = f.readlines()
2  for line in lines:
3      print(line.strip())
```

- ❖ readlines()는 파일의 모든 라인을 한 번에 읽는다.
- ❖ strip()는 앞/뒤의 공백을 없애준다.

```
Helle
World
프린트 함수로 쓸 수 있습니다.
```

```
1  f.close()
```

형식이 있는 텍스트 데이터

- 형식이 있는 텍스트 데이터(member.txt)
  - 홍길동,20,kildong@hong.com,서울시 강동구
  - 홍길서,25,kilseo@hong.com,서울시 강서구

- 이러한 형식으로 저장된 데이터를 읽어 사용하려면…
  - 한 라인씩 읽기
  - 한 라인을 콤마(,)등의 구분자(delimiter)로 분리(split)하기
  - 자료형 변환하기
  - NA또는 없는 필드에 대한 예외 처리하기

형식이 있는 텍스트 데이터

2절. 피클을 이용한 객체 저장하고 불러오기

```
f = open("member.txt", "r", encoding='UTF8')
lines = f.readlines()
for line in lines:
    data = line.strip().split(',')
    name = data[0]
    age = int(data[1])
    email = data[2]
    address = data[3]
    print("이름 : {}, 나이 : {}, 이메일 : {}, 주소 : {}"
          .format(name, age, email, address))
f.close()
```

```
이름 : 홍길동, 나이 : 20, 이메일 : kildong@hong.com, 주소 : 서울시 강동구
이름 : 홍길서, 나이 : 25, 이메일 : kilseo@hong.com, 주소 : 서울시 강서구
```

피클링

2절. 피클을 이용한 객체 저장하고 불러오기

- 파이썬 객체 계층 구조가 바이트 스트림으로 변환되는 것

```
pickle.dump(obj, file, protocol=None, *,
            fix_imports=True)
```

```
Pickler(file, protocol).dump(obj)
```

- 구문에서…
  - *obj* : 열려있는 파일에 저장할 객체
  - *file* : 피클링하기 위한 파일 객체
    단일 바이트 인수를 받아들이는 write() 메서드가 있어야 함
    open() 함수를 이용해 파일을 열 때 피클링하기 위한 모드는 'wb'
  - *protocol* : 프로토콜을 사용하기 위한 정수(0 ~ HIGHEST_PROTOCOL) 값
    기본값은 DEFAULT_PROTOCOL, 음수이면 HIGHEST_PROTOCOL이 선택
  - *fix_imports* : 이 인수가 True이고 프로토콜은 3보다 작으면 파이썬 2에서 피클 데이터 스트림을 읽을 수 있음

피클링

2절. 피클을 이용한 객체 저장하고 불러오기

```
class Member:
    def __init__(self, name, age, email, address):
        self.name = name
        self.age = age
        self.email = email
        self.address = address

    def __str__(self):
        return "Name:{}, Age:{}, Email:{}, Address:{}"₩
               .format(self.name, self.age, self.email, self.address)
```

고객 정보를 저장할 클래스

```
user1 = Member("홍길동", 20, "kildong@hong.com", "서울시 강동구")
user2 = Member("홍길서", 25, "kilseo@hong.com", "서울시 강서구")
user_list = [user1, user2]
```

파일에 저장할 데이터(고객 정보를 갖는 리스트 객체)

```
f = open("member.data", "wb")
import pickle
pickle.dump(user_list, f)
f.close()
```

피클링 하려면 바이너리 쓰기 모드로 열어야 함

언피클링

2절. 피클을 이용한 객체 저장하고 불러오기

- 피클링한 데이터를 다시 불러오는 것

```
data = pickle.load(file, *, fix_imports=True,
                   encoding="ASCII")
```

```
Unpickler(file).load()
```

- 구문에서…
  - *file* : 불러올 파일 객체
    이 객체는 정수 인수를 사용하는 read() 메서드와 인수가 필요 없는 readline() 메서드의 두 가지 메서드가 있어야 함
    open() 함수를 이용해 파일을 열 때 'rb' 모드로 열려야 함
  - *fix_imports* : 이 인수가 True 이면 피클은 파이썬 2 이름을 파이썬 3 이름에서 사용된 재 이름에 매핑하려고 시도
  - encoding : 기본값은 'ASCII
    피클된 8비트 문자열 인스턴스를 어떻게 디코딩 하는지를 알려줌
    8비트 문자열 인스턴스를 바이트 객체로 읽으려면 인코딩을 'byte'로 지정

언피클링

2절. 피클을 이용한 객체 저장하고 불러오기

```
del user_list
import pickle
f = open("member.data", "rb")
user_list = pickle.load(f)
type(user_list)

list
```

언피클링 하려면 바이너리 읽기 모드로 열어야 함

불러온 데이터의 타입은 리스트

```
for user in user_list:
    print(user)
f.close()
```

```
Name:홍길동, Age:20, Email:kildong@hong.com, Address:서울시 강동구
Name:홍길서, Age:25, Email:kilseo@hong.com, Address:서울시 강서구
```

CSV(Comma Separated Values)

3절. CSV 형식 파일 읽기/쓰기

- 스프레드시트 또는 데이터베이스를 가져오거나 내보내기 할 때 가장 많이 사용하는 일반적인 형식
- 프로그래머는 엑셀(Excel)에서 사용하는 CSV 형식의 세부적인 내용을 알지 못해도 엑셀에서 선호하는 형식으로 데이터를 쓰거나 엑셀에서 생성된 CSV 파일의 데이터를 읽을 수 있음
- csv 모듈
  - 파이썬 기본 라이브러리에 포함
  - 일반적으로 csv 모듈의 reader()와 writer()를 이용
  - 데이터의 메타정보를 저장하고 싶다면 csv 모듈의 DictReader와 DictWriter 클래스를 사용하여 딕셔너리 형식으로 데이터를 읽고 쓸 수 있음

reader

3절. CSV 형식 파일 읽기/쓰기

- reader() 함수는 지정된 파일에서 분리된 문자열로 데이터를 읽기 위한 객체를 반환

```
csv.reader(csvfile, dialect='excel', **fmtparams)
```

- 구문에서…
  - *csvfile* : 데이터가 저장되어 있는 CSV 파일의 객체
  - *dialect='excel'* : 엑셀에서 생성한 CSV 파일의 일반적인 속성일 경우 기본값 'excel'을 사용
    엑셀에서 생성된 탭(TAB) 구분 파일의 속성을 정의하려면 'excel-tab'을 사용
    UNIX 시스템에서 생성되는 CSV 파일(라인 구분자가 ₩n) 속성을 사용하려면 'unix'를 사용
  - ***fmtparams* : 추가적인 속성을 부여할 때 사용
    delimiter 속성(기본값은 ',')은 필드 구분자를 지정
    quotechar 속성(기본값은 "")은 인용할 문자를 지정
    CSV 파일에서 읽은 각 행은 문자열 목록을 반환
    quoting=csv.QUOTE_NONNUMERIC 속성을 지정하지 않으면 quotechar로 둘러싸인 문자가 아니어도 자동으로 데이터 형 변환이 수행되지 않음

CSV 파일 읽기

3절. CSV 형식 파일 읽기/쓰기

member1.csv

```
"홍길동",20,"kildong@hong.com","서울시 강동구"
"홍길서",25,"kilseo@hong.com","서울시 강서구"
```

```
import csv
with open('member1.csv', 'r', encoding='UTF8') as csvfile:
    r = csv.reader(csvfile)
    for row in r:
        print(row)
```

```
['홍길동', '20', 'kildong@hong.com', '서울시 강동구']
['홍길서', '25', 'kilseo@hong.com', '서울시 강서구']
```

quotechar 속성과 quoting 속성

- 자동 형변환을 위한 딕셔너리 인수
  - quotechar='"'
  - quoting=csv.QUOTE_NONNUMERIC
- 숫자 데이터가 아닌 문자 데이터가 문자열임을 알리는 따옴표 등으로 묶여있지 않을 경우 형변환 에러가 발생
  - 홍길동,20,kildong@hong.com,서울시 강동구

```
import csv
with open('member1.csv', 'r', encoding='UTF8') as csvfile:
    r = csv.reader(csvfile, quotechar='"', quoting=csv.QUOTE_NONNUMERIC)
    for row in r:
        print(row)
```

```
['홍길동', 20.0, 'kildong@hong.com', '서울시 강동구']
['홍길서', 25.0, 'kilseo@hong.com', '서울시 강서구']
```

writer

3절. CSV 형식 파일 읽기/쓰기

- 파일에 구분자로 연결된 문자열 데이터를 저장하는 객체를 반환

```
csv.writer(csvfile, dialect='excel', **fmtparams)
```

- 구문에서…
  - *csvfile* : 데이터를 저장하기 위한 CSV 파일의 객체.
    write() 메서드가 있는 모든 객체
  - *dialect='excel'* : 엑셀에서 생성한 CSV 파일의 일반적인 속성일 경우 기본값 'excel'을 사용
    엑셀에서 생성된 탭(TAB) 구분 파일의 속성을 정의하려면 'excel-tab'을 사용
    UNIX 시스템에서 생성되는 CSV 파일(라인 구분자가 ₩n) 속성을 사용하려면 'unix'를 사용
  - ***fmtparams* : 추가적인 속성을 부여할 때 사용
    delimiter 속성(기본값은 ',')은 필드 구분자를 지정
    quotechar 속성(기본값은 "")은 인용할 문자를 지정
    quoting 속성은 문자열을 quotechar로 인용할지 여부를 결정
    csv.QUOTE_NONNUMERIC을 사용하면 숫자가 아닌 데이터만 quotechar로 인용

writer()

3절. CSV 형식 파일 읽기/쓰기

```
user_list = [['홍길동', 20.0, 'kildong@hong.com', '서울시 강동구'],
             ['홍길서', 25.0, 'kilseo@hong.com', '서울시 강서구']]
```

```
import csv
with open('member2.csv', 'w', newline='', encoding='UTF8') as csvfile:
    w = csv.writer(csvfile)
    w.writerows(user_list)
```

```
홍길동,20.0,kildong@hong.com,서울시 강동구
홍길서,25.0,kilseo@hong.com,서울시 강서구
```

```
import csv
with open('member2.csv', 'w', newline='', encoding='UTF8') as csvfile:
    w = csv.writer(csvfile, quoting=csv.QUOTE_NONNUMERIC)
    w.writerows(user_list)
```

```
"홍길동",20.0,"kildong@hong.com","서울시 강동구"
"홍길서",25.0,"kilseo@hong.com","서울시 강서구"
```

DictReader

3절. CSV 형식 파일 읽기/쓰기

- DictReader 클래스는 reader와 비슷하게 동작하지만 각 행의 정보를 OrderedDict 객체로 반환
- CSV 파일에 각 열들의 이름(메타정보)을 지정해서 데이터를 **딕셔너리 형식**으로 불러옴

```
csv.DictReader(f, fieldnames=None, restkey=None,
               restval=None, dialect='excel',
               *args, **kwds)
```

- 구문에서...
  - *fieldnames* : 파일객체 f가 헤더 정보를 포함하고 있지 않을 경우 필드의 이름들을 설정하기 위해 사용
  - *restkey* : 행에 필드 이름보다 많은 필드가 있으면 나머지 데이터가 restkey 속성에 지정된 필드 이름과 함께 저장
  - *restval* : 필드 이름보다 필드 수가 적으면 restval에 지정된 값으로 채워 짐

딕셔너리 형식으로 읽기

3절. CSV 형식 파일 읽기/쓰기

```
1  import csv
2  with open('member3.csv', encoding='UTF8') as csvfile:
3      dict_reader = csv.DictReader(csvfile)
4      for row in dict_reader:
5          print(row["Name"], row["Age"], row["Email"], row["Address"])
```

```
홍길동 20 kildong@hong.com 서울시 강동구
홍길서 25 kilseo@hong.com 서울시 강서구
```

```
1  import csv
2  with open('member3.csv', encoding='UTF8') as csvfile:
3      dict_reader = csv.DictReader(csvfile)
4      for row in dict_reader:
5          print(row)
```

```
OrderedDict([('Name', '홍길동'), ('Age', '20'), ('Email', 'kildong@hong.com'), ('Ad
dress', '서울시 강동구')])
OrderedDict([('Name', '홍길서'), ('Age', '25'), ('Email', 'kilseo@hong.com'), ('Add
ress', '서울시 강서구')])
```

fieldnames 속성

- csv 파일이 헤더 정보를 포함하고 있지 않을 경우 fieldnames 속성을 이용해서 각 필드들의 이름을 지정

```
1  import csv
2  with open('member1.csv', encoding='UTF8') as csvfile:
3      dict_reader = csv.DictReader(csvfile,
4                                   fieldnames=["Name", "Age", "Email", "Address"])
5      for row in dict_reader:
6          print(row["Name"], row["Age"], row["Email"], row["Address"])
```

```
홍길동 20 kildong@hong.com 서울시 강동구
홍길서 25 kilseo@hong.com 서울시 강서구
```

restkey 속성

- 지정한 필드의 이름보다 데이터의 수가 많을 경우 restkey 속성에 지정한 이름에 남는 데이터가 저장 됨

```
1  import csv
2  with open('member1.csv', encoding='UTF8') as csvfile:
3      dict_reader = csv.DictReader(csvfile,
4                                   fieldnames=["Name", "Age", "Email"], restkey="Etc")
5      for row in dict_reader:
6          print(row["Name"], row["Age"], row["Email"], row["Etc"])
```

```
홍길동 20 kildong@hong.com ['서울시 강동구']
홍길서 25 kilseo@hong.com ['서울시 강서구']
```

restval 속성

3절. CSV 형식 파일 읽기/쓰기

- 지정한 필드의 수 보다 값이 적을 경우 restval 속성에 지정한 값이 저장됨

```
Name,Age,Email,Address,Etc
홍길동,20,kildong@hong.com,서울시 강동구,동해번쩍
홍길서,25,kilseo@hong.com,서울시 강서구
```

```
1  import csv
2  with open('member3-2.csv', encoding='UTF8') as csvfile:
3      dict_reader = csv.DictReader(csvfile, restval="없음")
4      for row in dict_reader:
5          print(row)
```

```
OrderedDict([('Name', '홍길동'), ('Age', '20'), ('Email', 'kildong@hong.com'), ('Address', '서울시 강동구'), ('Etc', '동해번쩍')])
OrderedDict([('Name', '홍길서'), ('Age', '25'), ('Email', 'kilseo@hong.com'), ('Address', '서울시 강서구'), ('Etc', '없음')])
```

DictWriter

3절. CSV 형식 파일 읽기/쓰기

- DictWriter 클래스는 딕셔너리 데이터 행 별로 CSV 파일에 저장

```
csv.DictWriter(f, fieldnames, restval='',
               extrasaction='raise', dialect='excel',
               *args, **kwds)
```

- 구문에서…
  - *fieldnames* : 저장할 데이터의 헤더 정보를 지정
  - *restval* : 필드 이름보다 필드 수가 적으면 restval에 지정된 값으로 채워짐
  - *extrasaction* : extrasaction 인수는 'raise'일 경우 딕셔너리 데이터에 필드 이름에 없는 추가 값이 있을 경우 ValueError를 발생시킴. 'ignore'일 경우 딕셔너리의 추가 값이 무시됨

writerow()

3절. CSV 형식 파일 읽기/쓰기

- writerow() 함수는 한 개 행을 저장

```
user1 = {"Name":"홍길동", "Age":20, "Email":"kildong@hong.com",
         "Address":"서울시 강동구"}
user2 = {"Name":"홍길서", "Age":25, "Email":"kilseo@hong.com",
         "Address":"서울시 강서구"}
```

```
import csv
with open('member4.csv', 'w', newline='', encoding='utf8') as csvfile:
    fieldnames = ['Name', 'Age', 'Email', 'Address']
    dict_writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    dict_writer.writeheader()
    dict_writer.writerow(user1)
    dict_writer.writerow(user2)
```

```
Name,Age,Email,Address
홍길동,20,kildong@hong.com,서울시 강동구
홍길서,25,kilseo@hong.com,서울시 강서구
```

writerows()

3절. CSV 형식 파일 읽기/쓰기

writerows() 함수는 모든 딕셔너리 데이터를 저장

```
user1 = {"Name":"홍길동", "Age":20, "Email":"kildong@hong.com",
         "Address":"서울시 강동구"}
user2 = {"Name":"홍길서", "Age":25, "Email":"kilseo@hong.com",
         "Address":"서울시 강서구"}
```

```
import csv
with open('member4.csv', 'w', newline='', encoding='utf8') as csvfile:
    fieldnames = ['Name', 'Age', 'Email', 'Address']
    dict_writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    dict_writer.writeheader()
    dict_writer.writerow(user1)
    dict_writer.writerow(user2)
```

```
Name,Age,Email,Address
홍길동,20,kildong@hong.com,서울시 강동구
홍길서,25,kilseo@hong.com,서울시 강서구
```

extrasaction='raise'

3절. CSV 형식 파일 읽기/쓰기

```
1  import csv
2  with open('member5.csv', 'w', newline='', encoding='utf8') as csvfile:
3      fieldnames = ['Name', 'Age', 'Email']
4      dict_writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
5
6      dict_writer.writeheader()
7      dict_writer.writerows(user_list)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-57b843508365> in <module>()
      5
      6     dict_writer.writeheader()
----> 7     dict_writer.writerows(user_list)

C:\ProgramData\Anaconda3\lib\csv.py in writerows(self, rowdicts)
    156
    157     def writerows(self, rowdicts):
--> 158         return self.writer.writerows(map(self._dict_to_list, rowdicts))
    159
    160 # Guard Sniffer's type checking against builds that exclude complex()

C:\ProgramData\Anaconda3\lib\csv.py in _dict_to_list(self, rowdict)
    149             if wrong_fields:
    150                 raise ValueError("dict contains fields not in fieldnames: "
--> 151                                  + ", ".join([repr(x) for x in wrong_fiel
ds]))
    152         return (rowdict.get(key, self.restval) for key in self.fieldnames)
    153

ValueError: dict contains fields not in fieldnames: 'Address'
```

extrasaction='ignore'

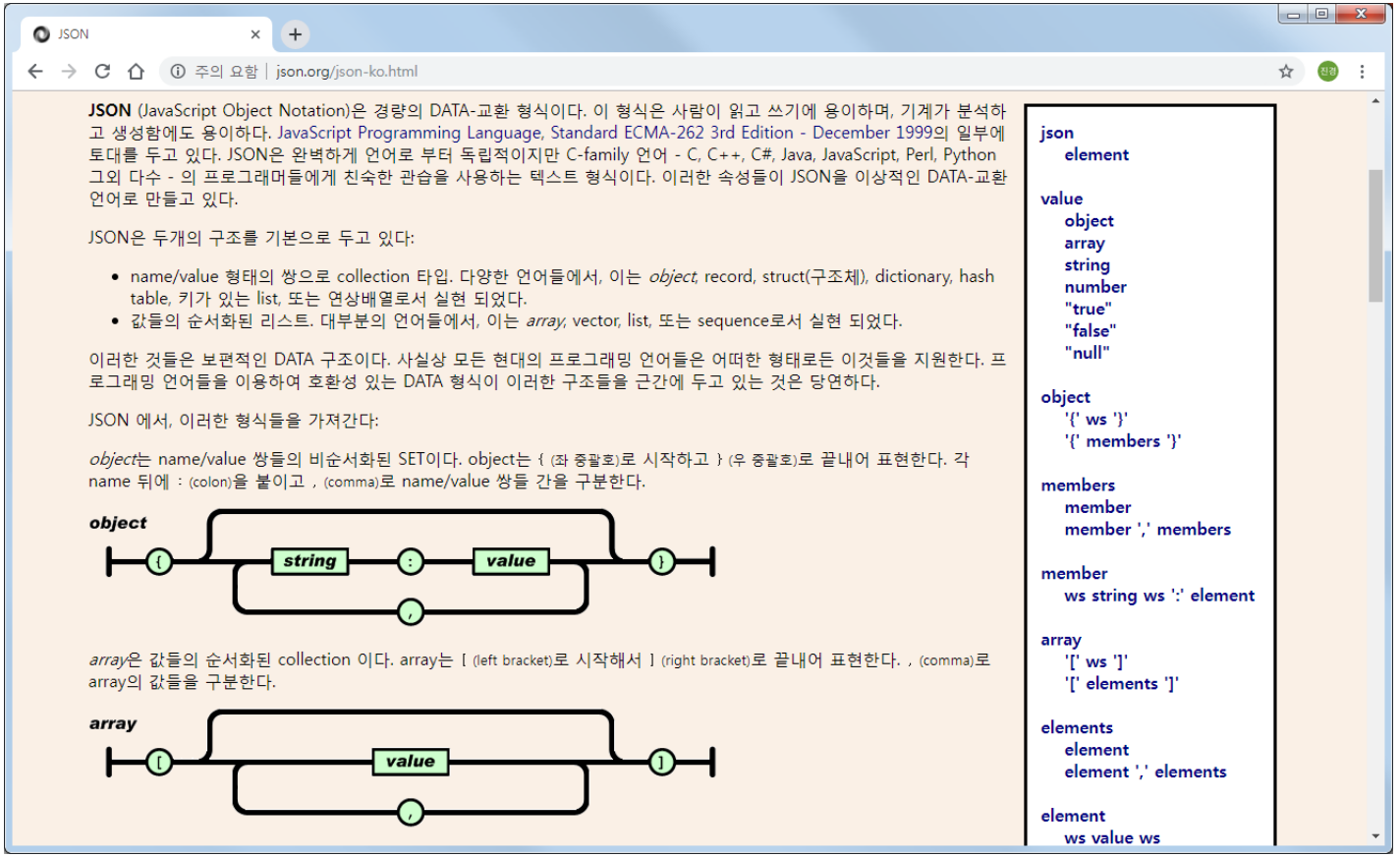

```
1  import csv
2  with open('member5-2.csv', 'w', newline='', encoding='utf8') as csvfile:
3      fieldnames = ['Name', 'Age', 'Email']
4      dict_writer = csv.DictWriter(csvfile, fieldnames=fieldnames,
5                                   extrasaction='ignore')
6
7      dict_writer.writeheader()
8      dict_writer.writerows(user_list)
```

```
Name,Age,Email
홍길동,20,kildong@hong.com
홍길서,25,kilseo@hong.com
```

JSON 데이터

4절. JSON 데이터 저장하고 불러오기

- JSON(JavaScript Object Notation)은 데이터를 교환하기 위한 형식
- 사람이 읽고 쓰기 쉽고, 기계가 분석하고 생성하기도 쉬움



json.dump

4절. JSON 데이터 저장하고 불러오기

- dump() 함수는 파이썬 객체를 JSON 형식으로 변환

```
json.dump(obj, fp, *, skipkeys=False, ensure_ascii=True,
          check_circular=True, allow_nan=True, cls=None,
          indent=None, separators=None, default=None,
          sort_keys=False, **kw)
```

- 구문에서...
  - *obj* : JSON 데이터로 변환할 파이썬 객체
  - *fp* : JSON 데이터를 저장하기 위한 파일 객체
  - *skipkeys* : skipkeys가 True이면 기본 유형(str, int, float, bool, None)이 아닌 키의 TypeError를 발생시키지 않음
  - 나머지 인수에 대한 설명은 교재를 참고하세요.

JSON 형식으로 저장

4절. JSON 데이터 저장하고 불러오기

```
1  data = ['foo', {'bar': ('baz', None, 1.0, 2)}]
```

```
1  import json
2  with open('sample.json', 'w') as jsonfile:
3      json.dump(data, jsonfile)
```

```
["foo", {"bar": ["baz", null, 1.0, 2]}]
```

```
1  import json
2  with open('sample.json', 'w') as jsonfile:
3      json.dump(data, jsonfile, indent=True)
```

```
[
 "foo",
 {
  "bar": [
   "baz",
   null,
   1.0,
   2
  ]
 }
]
```

```
1  import json
2  with open('sample.json', 'w') as jsonfile:
3      json.dump(data, jsonfile, indent='\t')
```

```
[
	"foo",
	{
		"bar": [
			"baz",
			null,
			1.0,
			2
		]
	}
]
```

JSON으로 직렬화하기

4절. JSON 데이터 저장하고 불러오기

```
[
    {
        "name": "홍길동",
        "age": 20,
        "email": "kildong@hong.com",
        "address": "서울시 강동구"
    },
    {
        "name": "홍길서",
        "age": 25,
        "email": "kilseo@hong.com",
        "address": "서울시 강서구"
    }
]
```

```
class Member:
    def __init__(self, name, age, email, addres
        self.name = name
        self.age = age
        self.email = email
        self.address = address

    def as_dict(self):
        return {"name":self.name, "age":self.age, "email":self.email,
                "address":self.address}

```

```
user1 = Member("홍길동", 20, "kildong@hong.com", "서울시 강동구")
user2 = Member("홍길서", 25, "kilseo@hong.com", "서울시 강서구")
user_list = [user1, user2]
```

```
import json
with open('member.json', 'w', encoding="UTF8") as jsonfile:
    json.dump(user_list, jsonfile, ensure_ascii=False,
              indent="₩t", default=Member.as_dict)
```

json.load

4절. JSON 데이터 저장하고 불러오기

- load() 함수는 JSON 형식 데이터를 파이썬 객체로 변환

```
json.load(fp, *, cls=None,
          object_hook=None, object_pairs_hook=None,
          parse_float=None, parse_int=None,
          parse_constant=None, **kw)
```

- 구문에서…
  - *fp* : JSON 데이터가 저장되어 있는 파일 객체
  - *cls* : JSONDecoder의 서브클래스를 지정
    그렇지 않으면 JSONDecoder가 사용
  - *object_hook* : object_hook은 디코딩 된 객체 리터럴의 결과로 호출되는 함수를 지정
    dict 대신 object_hook의 반환 값이 사용
  - 나머지 인수에 대한 설명은 교재를 참고하세요.

JSON 파일 읽기

4절. JSON 데이터 저장하고 불러오기

```
1  import json
2  with open('member.json', encoding="UTF8") as jsonfile:
3      user_list = json.load(jsonfile)
4      print(user_list)
```

```
[{'name': '홍길동', 'age': 20, 'email': 'kildong@hong.com', 'address': '서울시 강동구'}, {'name': '홍길서', 'age': 25, 'email': 'kilseo@hong.com', 'address': '서울시 강서구'}]
```

object_hook를 이용한 역직렬화

4절. JSON 데이터 저장하고 불러오기

```python
class Member:
    def __init__(self, name, age, email, address):
        self.name = name
        self.age = age
        self.email = email
        self.address = address

    def __str__(self):
        return "Name:{}, Age:{}, Email:{}, Address:{}"₩
               .format(self.name, self.age, self.email, self.address)

    def as_dict(cls):
        return {"name":cls.name, "age":cls.age, "email":cls.email,
                "address":cls.address}
```

```python
def as_member(dct):
    return Member(dct['name'], dct['age'], dct['email'], dct['address'])
```

```python
import json
with open('member.json', encoding="UTF8") as jsonfile:
    user_list = json.load(jsonfile, object_hook=as_member)
    for user in user_list:
        print(user)
```

읽은 데이터를 객체로 만들어 반환하는 메서드를 지정

```
Name:홍길동, Age:20, Email:kildong@hong.com, Address:서울시 강동구
Name:홍길서, Age:25, Email:kilseo@hong.com, Address:서울시 강서구
```

HDF5(Hierarchical Data Format)

5절. HDF5 파일 읽기/쓰기

- HDF 그룹에 의해 관리되고 있는 이기종 데이터를 저장, 관리, 처리하기 위한 고성능 데이터 소프트웨어 라이브러리 및 파일 형식
- HDF5는 빠른 입/출력 저장 및 처리를 위해 만들어졌음

- HDF5 포맷을 읽고 쓰기 위한 방법
  - 판다스(Pandas) 패키지의 read_hdf() 함수와 to_hdf() 함수를 사용해서 데이터 프레임으로 읽거나 쓰는 방법
  - h5py 모듈의 File 클래스를 이용해 파일 객체를 생성하고 읽는 방법

샘플 데이터

5절. HDF5 파일 읽기/쓰기

```
import seaborn as sns
iris_df = sns.load_dataset("iris")
iris_dic = iris_df.to_dict()
print(iris_dic["sepal_length"])
```

- 에드거 엔더슨(Edgar Anderson)의 iris 데이터셋
- 붓꽃의 3가지 종(setosa(세토사), versicolor(버시컬러), virginica(버지니카))별로 각각 50개 데이터의 꽃받침과 꽃잎의 길이/너비를 센티미터 단위로 측정하여 정리한 150개 데이터셋

```
{0: 5.1, 1: 4.9, 2: 4.7, 3: 4.6, 4: 5.0, 5: 5.4, 6: 4.6, 7: 5.0, 8: 4.4, 9: 4.9, 10:
5.4, 11: 4.8, 12: 4.8, 13: 4.3, 14: 5.8, 15: 5.7, 16: 5.4, 17: 5.1, 18: 5.7, 19: 5.1,
20: 5.4, 21: 5.1, 22: 4.6, 23: 5.1, 24: 4.8, 25: 5.0, 26: 5.0, 27: 5.2, 28: 5.2, 29:
4.7, 30: 4.8, 31: 5.4, 32: 5.2, 33: 5.5, 34: 4.9, 35: 5.0, 36: 5.5, 37: 4.9, 38: 4.4,
39: 5.1, 40: 5.0, 41: 4.5, 42: 4.4, 43: 5.0, 44: 5.1, 45: 4.8, 46: 5.1, 47: 4.6, 48:
5.3, 49: 5.0, 50: 7.0, 51: 6.4, 52: 6.9, 53: 5.5, 54: 6.5, 55: 5.7, 56: 6.3, 57: 4.9,
58: 6.6, 59: 5.2, 60: 5.0, 61: 5.9, 62: 6.0, 63: 6.1, 64: 5.6, 65: 6.7, 66: 5.6, 67:
5.8, 68: 6.2, 69: 5.6, 70: 5.9, 71: 6.1, 72: 6.3, 73: 6.1, 74: 6.4, 75: 6.6, 76: 6.8,
77: 6.7, 78: 6.0, 79: 5.7, 80: 5.5, 81: 5.5, 82: 5.8, 83: 6.0, 84: 5.4, 85: 6.0, 86:
6.7, 87: 6.3, 88: 5.6, 89: 5.5, 90: 5.5, 91: 6.1, 92: 5.8, 93: 5.0, 94: 5.6, 95: 5.7,
96: 5.7, 97: 6.2, 98: 5.1, 99: 5.7, 100: 6.3, 101: 5.8, 102: 7.1, 103: 6.3, 104: 6.5,
105: 7.6, 106: 4.9, 107: 7.3, 108: 6.7, 109: 7.2, 110: 6.5, 111: 6.4, 112: 6.8, 113:
5.7, 114: 5.8, 115: 6.4, 116: 6.5, 117: 7.7, 118: 7.7, 119: 6.0, 120: 6.9, 121: 5.6,
122: 7.7, 123: 6.3, 124: 6.7, 125: 7.2, 126: 6.2, 127: 6.1, 128: 6.4, 129: 7.2, 130:
7.4, 131: 7.9, 132: 6.4, 133: 6.3, 134: 6.1, 135: 7.7, 136: 6.3, 137: 6.4, 138: 6.0,
139: 6.9, 140: 6.7, 141: 6.9, 142: 5.8, 143: 6.8, 144: 6.7, 145: 6.7, 146: 6.3, 147:
6.5, 148: 6.2, 149: 5.9}
```

HDF 포맷 저장하고 불러오기

5절. HDF5 파일 읽기/쓰기

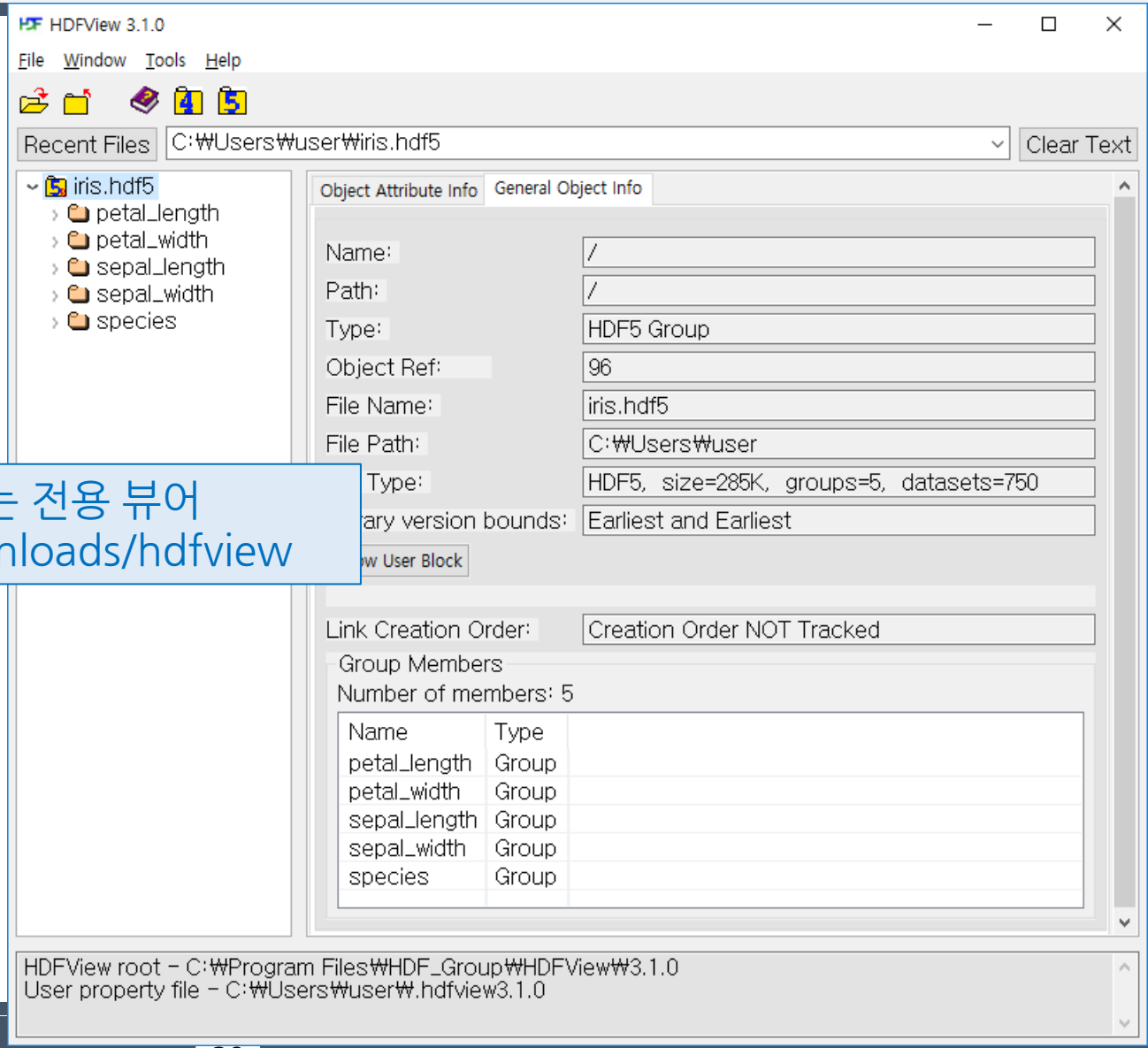
```
import h5py
with h5py.File("iris.hdf5", 'w') as f:
    for group, v in iris_dic.items():
        grp = f.create_group(group)
        for k, data in v.items():
            grp.create_dataset(str(k), data=data)
```

```
import h5py
iris_dic = {}
with h5py.File("iris.hdf5", 'r') as f:
    for group, v in f.items():
        cols = {}
        for k, data in v.items():
            cols.update({int(k):data.value})
        iris_dic.update({group: cols})
```

HDFView

5절. HDF5 파일 읽기/쓰기

저장한 HDF5 형식 파일을 볼 수 있는 전용 뷰어
https://www.hdfgroup.org/downloads/hdfview



판다스를 이용한 HDF5 파일 입출력

5절. HDF5 파일 읽기/쓰기

```
import pandas as pd
iris_df = pd.DataFrame(iris_dic)
iris_df.to_hdf("iris2.hdf5", key="iris")
```

```
iris_df2 = pd.read_hdf("iris2.hdf5", key="iris")
iris_df2.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

5절. 연습문제(실습형)

```
if __name__ == "__main__":
    main()
```

```
데이터가 로드 되었습니다.
1:입력|2:전체출력|3:삭제|4:이름찾기|5:내보내기(CSV)|9:종료 메뉴 선택 : 1
이름 : 홍길동
전화번호 : 010-8888-8888
이메일 : hong2@abc.com
나이 : 31
고객등급(1~5) : 5
기타 정보 : abc
1:입력|2:전체출력|3:삭제|4:이름찾기|5:내보내기(CSV)|9:종료 메뉴 선택 : 2
================================================================
          고객  정보
----------------------------------------------------------------
GRADE 이름  전화                메일                 나이 기타
================================================================
   ** 유길동 010-5432-2222   yu@hong.com        20 가칠해
***** 홍길동 010-9999-9999   hong@mega_it.com   30 열심히
***** 홍길동 010-8888-8888   hong2@abc.com      31 abc
================================================================
```

연습문제(실습형)

```
1:입력|2:전체출력|3:삭제|4:이름찾기|5:내보내기(CSV)|9:종료 메뉴 선택 : 4
찾을 고객 이름은 ? 홍길동
=============================================================
            고객  정보
-------------------------------------------------------------
GRADE 이름  전화                  메일                  나이 기타
=============================================================
***** 홍길동 010-9999-9999   hong@mega_it.com  30 열심히
***** 홍길동 010-8888-8888   hong2@abc.com    31 abc
=============================================================
1:입력|2:전체출력|3:삭제|4:이름찾기|5:내보내기(CSV)|9:종료 메뉴 선택 : 5
저장할 파일이름은 : abc.csv
CSV 파일이 저장되었습니다. 파일명은  abc.csv
1:입력|2:전체출력|3:삭제|4:이름찾기|5:내보내기(CSV)|9:종료 메뉴 선택 : 4
찾을 고객 이름은 ? 유길동
=============================================================
            고객  정보
-------------------------------------------------------------
GRADE 이름  전화                  메일                  나이 기타
=============================================================
   ** 유길동 010-5432-2222   yu@hong.com       20 가칠해
=============================================================
1:입력|2:전체출력|3:삭제|4:이름찾기|5:내보내기(CSV)|9:종료 메뉴 선택 : 9
```

연습문제(문제풀이형)

1.다음 중 open 함수에 대해 잘못 설명한 것은?
① 함수가 반환하는 것은 열린 파일 객체이다.
② 파일을 열 때 사용하는 모드는 읽기(r), 쓰기(w), 추가(a), 바이너리(b) 모드가 있다.
③ 바이너리(b) 모드와 쓰기(w) 모드는 같이 사용할 수 없다.
④ 파일의 인코딩은 encoding 속성을 이용해 설정할 수 있다.

2.피클링(pickling)에 대해 잘 못 설명한 것은?
① 파이썬의 객체 직렬화(Object Serialization) 방법이다.
② pickle 모듈의 dump()와 load() 함수를 이용해 객체를 쓰고 읽는다.
③ 파이썬 객체를 별도의 텍스트 변환과정 없이 파일에 직접 쓰고 읽는 것을 의미한다.
④ 피클링하기 위한 모드는 쓰기모드('w') 여야 한다.

3.다음 중 파이썬의 입/출력 모듈에 대한 설명 중 잘 못된 것은?
① Pandas 패키지를 이용하면 리스트 데이터를 쉽게 파일에 쓰고 읽을 수 있다.
② json 모듈의 dump(), load() 함수를 이용해 JSON 데이터를 파일에 읽고 쓸 수 있다.
③ h5py 모듈의 File 클래스를 이용해 HDF5 데이터를 파일에 읽고 쓸 수 있다.
④ csv 모듈의 reader(), writer() 함수를 이용해 CSV 파일을 읽고 쓸 수 있다.