파일 읽기 / 쓰기 활용

CSV, JSON, XML

AI

AI

파일 읽기 / 쓰기

› 파이썬에서 파일 다룰 때는 기본 내장함수 open() 사용

› 첫번째 인수 file 경로 만이 필수 이며, 나머지는 필요에 따라 옵션을 줄 수 있음

```
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None,
closefd=True, opener=None)
```

mode	설명
r	읽기 모드(기본값)
w	쓰기 모드(기존 파일 삭제)
x	쓰기 모드(파일 존재 시 오류 발생)
a	쓰기 모드(파일 존재 시 내용 추가)

mode	설명
b	바이너리 모드
t	텍스트 모드(기본값)
+	읽기쓰기 모드
U	유니버설 개행 모드(사용되지 않음)

AI

파일 읽기 / 쓰기

› 파일 모드는 기본 값으로 rt 설정

```
>>> f = open('file.txt', 'rt')
```

› 바이너리 쓰기모드 wb

```
>>> f = open('file.txt', 'wb')
```

› 텍스트 읽기쓰기모드 r+t, 맨 앞에서부터 내용을 덮어 씀 (파일이 존재 하지 않으면 오류 발생)

```
>>> f = open('file.txt', 'r+t')
```

AI

파일 읽기 / 쓰기

› 텍스트 읽기쓰기 모드 w+t, 파일 내용을 다 지우고 다시 씀

```
>>> f = open('file.txt', 'w+t')
```

› 텍스트 읽기쓰기 모드 a+t, 파일의 모든 내용을 남겨두고 맨 뒤에서부터 씀

```
>>> f = open('file.txt', 'a+t')
```

› '+'는 기본적으로 읽기쓰기모드지만 기존 파일 내용을 처리하는 방식에 차이가 생김

AI

파일 읽기 / 쓰기

› 파일 쓰기 예제

```
f = open('file.txt', 'w')
f.write('파일에 기록될 문자열입니다.\n')
f.write('파일 기록하는거 참 쉽죠?!\n')
f.close()

f = open('file.txt', 'a')
f.writelines(['하나', '둘', '셋', '넷', '다섯'])
f.write('\n')
f.writelines('\n'.join(['여섯', '일곱', '여덟', '아홉', '열']))
f.close()
```

AI

파일 읽기 / 쓰기

› 파일 읽기 예제

› read 함수

– 파일에 있는 데이터를 한번에 모두 불러옴

```
f = open('file.txt', 'r')
s = f.read()
print(s)
f.close()
```

AI

파일 읽기 / 쓰기

› readline 함수

- 파일 한 라인씩 읽어옴(₩n으로 구분)
- 파일 객체는 이터레이블하므로 readline을 호출 하였을때 다음 라인으로 자동으로 커서가 내려감

```
f = open('file.txt', 'r')
s = f.readline()
print(s, end='')
for s in f:
    print(s, end='')
f.close()
```

AI

파일 읽기 / 쓰기

› readline 함수

- 파일 한 라인씩 읽어옴(₩n으로 구분)
- 이터레이터 이므로 readline을 호출 하였으면 다음 라인으로 자동으로 커서가 내려감

```
f = open('file.txt', 'r')
s = f.readline()
print(s, end='')
for s in f:
    print(s, end='')
f.close()
```

AI

파일 읽기 / 쓰기

› readlines 함수

– 라인 별 끊어서 리스트에 저장(₩n으로 구분)

```
f = open('file.txt', 'r')
lines = f.readlines()
print(lines)
s = ''.join(lines)
print(s)
f.close()
```

– 리스트로 관리되기 때문에 필요하다면 for문을 통해서 라인관리 가능

AI

파일 읽기 / 쓰기

› 파일 현재 위치 확인(tell) 및 변경(seek)

```
f = open('file.txt', 'r')
print('파일 현재 위치:', f.tell())
print(f.read())
print('파일 현재 위치:', f.tell())
# 처음으로
f.seek(0)
print('파일 현재 위치:', f.tell())
print(f.readline(), end='')
print('파일 현재 위치:', f.tell())
f.close()
```

AI

파일 읽기 / 쓰기

› 파일 처리는 예상치 못한 오류가 언제든지 발생 가능하므로 예외처리는 필수적

› 예외 처리 경우 finally 블록으로 파일 닫기(close)를 해주는 것이 보편적

```
try:
    f = open('file.txt', 'r')
    ...(내용)...
except:
    print('여기서 예외처리를 합니다.')
finally:
    f.close()
```

AI

파일 읽기 / 쓰기

› with문을 사용하여 파일을 처리하면
  with 블록을 벗어 났을 때 자동으로 파일이 닫힘

```
with open('file.txt', 'r') as f:
    try:
        for s in f:
            print(s, end='')
    except:
        print('여기서 예외처리를 합니다.')
```

AI

CSV 다루기

› CSV(Comma-separated values)의 약자이며, 각 라인의 컬럼들이 콤마로 분리된 텍스트 파일 포맷

› 파이썬은 기본적으로 csv 모듈이 존재

```
import csv

f = open('data-iris-1.csv', 'r')
rdr = csv.reader(f)
for line in rdr:
    print(line)
f.close()
```

AI

CSV 다루기

› csv 특성 상 가장 윗 부분은 각 데이터의 설명 및 이름명이 존재 할 경우가 많으며 실제 데이터 처리에는 필요 없는 부분이므로 next()함수를 실행하여 생략 가능

```
import csv

with open('covid_19_data.csv', 'r') as f:
    rdr = csv.reader(f)
    next(rdr)
    for line in rdr:
        print(line)
```

AI

CSV 다루기

› csv 쓰기 예제

› newline을 빈칸을 해줌으로서 자동 줄바꿈을 제거

› f.write(f"{1},{'대구'},{6781}") -> wr.writerow([1, '대구', 6781])

```
import csv

with open('write.csv', 'r', newline='') as f:
    wr = csv.writer(f)
    wr.writerow([1, '대구', 6781])
    wr.writerow([2, '경북', 1316])
    wr.writerow([3, '경기', 580])
```

AI

CSV 다루기

› writerows() 함수를 통해서 리스트에 미리 데이터를 넣어두고 한번에 write도 가능

```
import csv

w_list = [[4, '서울', 563], [5, '충남', 136]]
with open('write.csv', 'a', newline='') as f:
    wr = csv.writer(f)
    wr.writerows(w_list)
```

AI

JSON 다루기

- JSON은 JavaScript Object Notation의 약자이며, 데이터를 교환하는 한 포맷으로서 단순함과 유연함 때문에 널리 사용
- JSON 포맷은 Key-Value Pair 이며, {}로 데이터를 둘러쌓아 표현
- 파이썬에서 딕셔너리와 동일한 양식을 가지고 있어 쉽게 사용 및 가공이 가능

AI

JSON 다루기

› JSON 구조

```
{
    "name": "식빵",
    "family": "웰시코기",
    "age": 1,
    "weight": 2.14
}
```

› JSON은 숫자, 문자열, 불리언, 배열, NULL 타입을 작성 가능

AI

JSON 다루기

› JSON 배열은 []로 둘러쌓아 표현하여 여러 JSON 데이터를 포함

```
[
    {"name": "식빵", "family": "웰시코기", "age": 1, "weight": 2.14},
    {"name": "콩콩", "family": "포메라니안", "age": 3, "weight": 2.5},
    {"name": "젤리", "family": "푸들", "age": 7, "weight": 3.1}
]
```

AI

JSON 다루기

› 파이썬에서 json 모듈을 기본적으로 지원

```
import json

dog = {"name": "식빵", "family": "웰시코기", "age": 1, "weight": 2.14}
# json 인코딩
json_string = json.dumps(dog, indent=4)
print(json_string)
print(type(json_string))
# json 디코딩
json_dict = json.loads(json_string)
print(json_dict)
print(type(json_dict))
```

AI

JSON 다루기

› JSON 파일 읽기

```
import json

with open('covid_19_data_json.json', 'r') as f:
    cov_list = json.load(f)
    print(cov_list)
    for cov_info in cov_list:
        print("="*25)
        print(cov_info)
```

AI

JSON 다루기

› JSON 파일 쓰기

```
import json

with open('cov_count.json', 'w') as f:
    cov_info = [{"id": 1, "region": "대구", "count:" 6781},
                {"id": 2, "region": "경북", "count": 1316},
                {"id": 3, "region": "경기", "count:" 580}]
    json.dump(cov_info, f)
```

AI

JSON 다루기

› 윈도우 경우 cp949라는 고유의 유니코드 사용

› 다른 OS에서 문자열이 깨지는 문제가 발생 할 수 있으므로 UTF-8 유니코드로 저장 할 수 있도록 변경

```
import json

with open('cov_count.json', 'w', encoding="utf-8") as f:
    cov_info = [{"id": 1, "region": "대구", "count:" 6781},
                {"id": 2, "region": "경북", "count": 1316},
                {"id": 3, "region": "경기", "count:" 580}]
    json.dump(cov_info, f)
```

AI

JSON 다루기

› indent를 설정하여 들여쓰기를 통해 JSON의 가독성을 높힘

› ensure_ascii를 False로 하여 유니코드 16진수를 문자 그대로 표현 가능

```
import json

with open('cov_count.json', 'w', encoding="utf-8") as f:
    cov_info = [{"id": 1, "region": "대구", "count:" 6781},
                {"id": 2, "region": "경북", "count": 1316},
                {"id": 3, "region": "경기", "count:" 580}]
    json.dump(cov_info, f, ensure_ascii=False, indent=4)
```

AI

JSON 다루기

› 딕셔너리는 순서가 정해져 있지 않으므로 많은 데이터들을 딕셔너리에 넣고 JSON으로 저장하면 원하지 않는 모양으로 저장 될 수 있음

› json 모듈에서는 keys를 기준으로 정렬하여 저장 가능 (sort_keys)

```
import json

with open('cov_count.json', 'w', encoding="utf-8") as f:
    . . .
    json.dump(cov_info, f, ensure_ascii=False, indent=4,
sort_keys=True)
```

AI

JSON 다루기

› sort_keys는 keys 정렬을 생각해서 네이밍을 지어야 되는 문제점이 존재

› 지정된 순서로 정확하게 json 파일을 생성하기 위해서는 OrderedDict 모듈 사용

```
from collections import OrderedDict


cov_info = OrderedDict()
cov_info["id"] = 1
cov_info["region"] = "대구"
cov_info["count"] = 6781
 . . .
```

AI

JSON 다루기

› OrderedDict 작성의 다양한 방법

› 변수 = OrderedDict()
변수[key] = value

› 변수 = OrderedDict([(key, value), (key, value), ...]

```
from collections import OrderedDict

keys = ["id", "region", "count"]
value_list = [[1, "대구", 6781], [2, "경북", 1316], [3, "경기", 580]]
cov_info = [OrderedDict((key, value) for key, value in zip(keys,
values))
                                                    for values in value_list]
```

AI

XML 다루기

- XML(eXtensible Markup Language) 마크업 언어를 정의하기 위한 언어
- XML은 데이터가 무엇인지에 초점을 맞춰 데이터를 기술하기 위해 고안 되었으며, HTML은 데이터가 어떻게 보일지에 초점을 맞춰 데이터를 표시하기 위해 고안
- XML은 데이터를 구조화시키는데 사용
  HTML은 동일한 데이터를 표시하고 꾸미는데 사용

AI

XML 다루기

› 파이썬에서 xml 모듈을 기본적으로 지원
  외부 모듈로는 lxml이 있으며 좀 더 효율적으로 사용 가능

```
import xml.etree.ElementTree as ET

tree = ET.parse("motorcycle-01.xml")
root = tree.getroot()

for elem in root:
    . . .
```

```
<?xml version="1.0"?>
<data>
<student>
    <name>peter</name>
    <age>24</age>
    <score math="80" english="97"/>
</student>
<student>
    <name>elgar</name>
    <age>21</age>
    <score math="67" english="56"/>
</student>
<student>
    <name>hong</name>
    <age>36</age>
    <score math="76" english="81"/>
</student>
</data>
```

AI

XML 다루기

› getroot() 메서드를 통해 xml 의 최상단의 요소로 이동

› 각 요소에는 tag, text, attrib가 존재

```
<data> ← 첫 getroot() 선언 시 요소 위치
<student> ← tag는 요소의 이름 (student)
    <name>peter</name> ← text는 요소의 내용 (peter)
    <age>24</age>
    <score math="80" english="97"/> ← attrib는 요소의 속성값
(math)
</student>
</data>
```

AI

XML 다루기

› getroot() 반환되는 요소 데이터는 이터레이블 하므로 아래에 자식 요소가 있다면 반복문을 통해서 접근 가능

```
import xml.etree.ElementTree as ET

tree = ET.parse("motorcycle-01.xml")
root = tree.getroot()
print(root.tag)

for elem in root:
    print(f"tag: {elem.tag}\ntext: {elem.text}\nattrib: {elem.attrib}")
```

AI

XML 다루기

› 일치하는 모든 요소를 가져오기 위해서는 findall("요소명")을 사용

› 일치하는 첫 번째 요소만을 가져오려면 find("요소명") 사용

```
import xml.etree.ElementTree as ET

tree = ET.parse("motorcycle-01.xml")
root = tree.getroot()

image_elems = root.findall("image")
for elem in image_elems:
    print(f"tag: {elem.tag}\ntext: {elem.text}\nattrib: {elem.attrib}")
```

AI

XML 다루기

› 요소의 특정 속성값만을 가져오고 싶을 때는
get("속성 key값"[, "없을때 출력 문자열"]) 메서드 사용

```
import xml.etree.ElementTree as ET

tree = ET.parse("motorcycle-01.xml")
root = tree.getroot()

image_elems = root.findall("image")
image_attribs = [elem.get("id", "None") for elem in image_elems]
print(image_attribs)
```

AI

XML 다루기

› 이외에도 요소의 속성값의 keys만 가져오고 싶을 때 keys() 메서드를 key-value 쌍을 가져오고 싶을 땐 items() 메서드를 사용

```
import xml.etree.ElementTree as ET

tree = ET.parse("motorcycle-01.xml")
root = tree.getroot()

image_elems = root.findall("image")
image_attribs = [elem.items() for elem in image_elems]
print(image_attribs)
```

AI

XML 다루기

› XML(str) 함수를 통해 문자열을 xml 형태로 작성 후 xml로 전환 가능

```
import xml.etree.ElementTree as ET


text = """
<student>
    <name>peter</name>
    <score english="97" math="80" science="75" />
</student>
"""
t_elem = ET.XML(text)
ET.dump(t_elem)
```

AI

XML 다루기

› Element(elem_tag) 함수는 요소 한 개를 직접 생성

› append(elem) 메서드를 통해 자식 요소 추가

```
import xml.etree.ElementTree as ET

student = ET.Element("student")

name = ET.Element("name")
name.text = "peter"

student.append(name)
ET.dump(student)
```

AI

XML 다루기

› SubElement(p_elem, elem_tag[, *attrib]) 함수를 사용하여 자식 요소를 바로 생성

› 첫번째 인자는 부모 요소, 두번째 인자는 자식 요소 태그명, 세번째 부터는 자식 요소의 속성을 추가

```
import xml.etree.ElementTree as ET


student = ET.Element("student")
score = ET.SubElement(student, "score", science="75")
score.set("math", "80")
score.attrib["english"] = "97"
ET.dump(student)
```

AI

XML 다루기

› insert(index, elem) 메서드를 통해 원하는 위치에 자식 요소를 추가

```
import xml.etree.ElementTree as ET

student = ET.Element("student")

score = ET.SubElement(student, "score", science="75")

dummy = ET.Element("dummy")
student.insert(0, dummy)
ET.dump(student)
```

AI

XML 다루기

› remove(elem) 메서드를 통해 원하는 요소를 삭제

```
student = ET.Element("student")

score = ET.SubElement(student, "score", science="75")

dummy = ET.Element("dummy")
student.insert(0, dummy)
ET.dump(student)

student.remove(dummy)
ET.dump(student)
```

AI

XML 다루기

› ElementTree(elem).write(path) 메서드를 사용하여 xml 파일 쓰기

```
import xml.etree.ElementTree as ET

student = ET.Element("student")
. . . (요소 추가) . . .

ET.ElementTree(student).write("student.xml")
```

AI

XML 다루기

› xml 모듈은 json 처럼 indent 들여쓰기에 대한 옵션이 존재하지 않으므로 별도로 함수 구현

```
def indent(elem, level=0):
    i = "\n" + level*"  "
    if len(elem):
        if not elem.text or not elem.text.strip():
            elem.text = i + "  "
        if not elem.tail or not elem.tail.strip():
            elem.tail = i
        for elem in elem:
            indent(elem, level+1)
        if not elem.tail or not elem.tail.strip():
            elem.tail = i
    else:
        if level and (not elem.tail or not elem.tail.strip()):
            elem.tail = i
```