



파일 처리

안 화 수

파일처리

- ❖ 어플리케이션이 운영체제에게 파일 처리를 API 함수를 통해 요청하면, 운영체제가 요청한 업무를 수행해주고 그 결과를 어플리케이션에게 돌려줌.
- ❖ 어플리케이션이 파일 처리 업무를 의뢰하는 과정과 각 과정에서 사용되는 파이썬 함수



파일처리

❖ open() 함수

✓ 파일 쓰기

- 파일에 기록하는 경우에는 2개나 3개의 매개변수를 대입
- 첫번째 매개변수는 파일 경로이고 두번째는 'w' 이고 세번째는 인코딩 방식입니다.
- 인코딩 방식의 경우 생략이 가능한데 생략을 하게되면 시스템의 기본적인 문자 인코딩으로 저장 되게 됩니다.
- open()으로 연 파일 객체에 기록을 할 때는 write 메소드를 이용해서 기록할 내용을 전달하면 파일에 기록이 됩니다.
- \n이 포함된 문자열의 컬렉션의 경우에는 writelines를 이용해서 줄 단위 기록이 가능
- \n이 포함된 문자열의 컬렉션의 경우에는 join을 이용하면 write 메소드로도 줄 단위 기록이 가능하고 \n이 없는 경우에도 가능

✓ 파일 읽기

- 파일 경로만 입력하면 읽기 모드로 열리게 됩니다.
- 두번째 매개변수로 'r'을 전달해도 읽기 모드가 되고 생략하고 인코딩 방식을 대입해도 됩니다.
- 전체 데이터를 하나의 문자열 읽고자 하는 경우에는 read()를 이용하면 전체를 읽어 옵니다.
- 줄단위로 읽을 때는 파일 객체의 반복자나 readline() 또는 파일 전체를 줄 단위로 끊어서 리스트에 저장하는 readlines 메소드를 이용해서 줄 단위로 읽을 수 있습니다.
- read(정수)를 이용하면 정수만큼의 바이트를 읽어옵니다.

파일처리

❖ 파일 열기 모드: `open()` 함수의 두번째 매개변수

문자	의미
'r'	읽기용으로 열기 (기본값)
'w'	쓰기용으로 열기. 이미 같은 경로에 파일이 존재하면 파일내용을 비움.
'x'	배타적 생성모드로 열기. 파일이 존재하면 <code>IOError</code> 예외 일으킴.
'a'	쓰기용으로 열기. 단, 'w'와는 달리 이미 같은 경로에 파일이 존재하는 경우 기존 내용에 덧붙이기를 함.
'b'	바이너리 모드
't'	텍스트 모드 (기본값)
'+'	읽기/쓰기용으로 파일 읽기

인코딩

- ❖ 문자 집합(Character Set) – 텍스트를 기호로 표현한 것.
- ❖ ASCII(American Standard Code for Information Interchange)
 - 미국 정보 교환 표준 부호
 - 1960년대에 제정된 문자 집합으로, 이후에 개발된 문자 집합들의 토대가 됨.
 - ASCII는 7비트만을 이용하여 음이 아닌 수(0~127)에 문자 집합 내의 문자를 할당.
예) 61에는 '='를, 65에는 'A'를, 97에는 'a'를 할당
 - 52개의 알파벳 대소문자(A~Z, a~z), 10개의 숫자(0~9), 32개의 특수 문자(!@#\$%&'*~ 등등), 하나의 공백 문자, 그리고 33개의 출력 불가능한 제어문자, 모두 128개의 문자를 표현.

인코딩

❖ ISO/IEC 8859-1

- 128(2^7)개의 문자를 표현하는 ASCII가 7비트만 사용했던 것에 비해 8비트를 사용하여 256(2^8)개의 문자를 표현

❖ ISO/IEC 8859-N

- 중앙 유럽어, 남유럽어, 북유럽어, 아랍어 등을 지원

❖ DBCS(Double-Byte Character Set)

- 2바이트(16비트)를 활용해서 문자 집합을 구성하는 방법.
- 최대 65,536(2^{16})개의 문자를 할당할 수 있으며, 한글(총 11,172자), 중국과 일본의 문자를 컴퓨터로 표현 가능.
- DBCS는 ASCII와의 호환을 유지하기 위해 최상위 비트가 0이면 ASCII, 1이면 DBCS로 인식
- DBCS에서는 하나의 문자열 안에서도 1바이트로 표현되고 있는 문자와 2바이트로 표현되는 문자를 혼용.
- 한글 문자 집합 표준으로는 KS X 1001, EUC-KR, CP949 등이 있음.
- EUC-KR은 한글 표준인 KS X 1001과 ASCII에서 ' \backslash ' 기호를 '₩'로 바꾼 영문자 표준 KS X 1003의 합집합.

인코딩

❖ 유니코드(Unicode)

- 문자 집합 하나로 모든 문자를 표현할 수 있게 하는 것이 목적
- 초기에는 전세계의 언어별 문자들을 2바이트 안에서 영역을 나눠 할당
- 유니코드에서 문자에 부여되는 번호를 코드 포인트(Code Point)라고 함.
 - 코드포인트는 "U+" 뒤에 2바이트의 수를 16진수로 표현하여 붙여 표시
예) 'A'의 코드 포인트는 U+0041, '한'의 코드 포인트는 U+D55C, '글'의 코드 포인트는 U+AE00
 - U+0000부터 U+007F까지를 ASCII와 동일하게 맞춰두고 그 뒷번호부터 각국의 문자를 할당.
- 누락된 현대 문자와 기호를 추가적으로 할당하고 고대 문자와 음악 기호 등을 추가하자 코드포인트가 2바이트를 넘어서게 됨.

❖ UTF(Unicode Transformation Format)

- 유니코드 변환 인코딩 형식
- UTF-8은 코드포인트의 크기에 따라 1바이트에서부터 4바이트까지 가변폭으로 인코딩하므로 1 바이트로 표현 가능한 U+0000(십진수 0)부터 U+007F(십진수 127)까지는 ASCII와 완벽하게 호환
 - UTF-8 인코딩 방식으로 저장된 문서는 유니코드를 알지 못하는 시스템에서도 사용 가능.
- UTF-8 외에도 UTF-7, UTF-16, UTF-32 인코딩 등이 있음.

파일처리

❖ 파일의 경로 설정

✓ 절대 경로

- 파일의 경로를 루트부터 직접 기재하는 방식
- 리눅스나 매킨토시는 디렉토리 구분 기호로 /를 사용하고 윈도우의 경우에는 \를 사용
- 절대 경로를 사용할 때는 윈도우에서는 \\\\로 설정해야 합니다.

✓ 상대경로

- 파일의 경로를 현재 위치로부터의 상대적인 경로로 설정하는 방식
- /를 이용해서 디렉토리를 구분
- ./는 현재 디렉토리로 생략해도 현재 디렉토리가 됩니다.
- ../는 상위 디렉토리가 됩니다.

파일 쓰기

❖ 프로그램을 실행하고 현재 작업 디렉토리 확인

```
file = open('test.txt', 'w')
#파일 객체 정보 출력
print(file)
#데이터를 한번에 기록
file.write('Hello File')
file.write('\n\n')
#데이터를 줄 단위로 기록
lines = ['안녕하세요', '반갑습니다.', '파이썬입니다.']
file.write('\n'.join(lines))
'''
\n이 포함된 경우는
file.write('').join(lines))
file.writelines(lines)
가능
'''

file.close()
```

Hello File

안녕하세요
반갑습니다.
파이썬입니다.

파일 읽기

```
file = open('test.txt', 'r')
#한꺼번에 전부 읽기
content = file.read()
print(content)
file.close()
print("=====")
#줄단위로 읽기
file = open('test.txt', 'r')
for line in file:
    print(line)
file.close()
print("=====")
file = open('test.txt', 'r')
lines = file.readlines()
print(lines)
file.close()
```

Hello File

안녕하세요
반갑습니다.
파이썬입니다.

=====
Hello File

안녕하세요

반갑습니다.

파이썬입니다.

=====
['Hello File\n', '\n', '안녕하세요\n', '반갑
습니다.\n', '파이썬입니다.']

with ~ as

- ❖ `open()` 함수와 함께 **with** ~ **as**문을 사용하면 명시적으로 `close()` 함수를 호출하지 않아도 파일이 항상 닫힘.
- ❖ **with** ~ **as** 문을 사용하는 방법은 다음과 같음.

with open(파일이름) as 파일객체:

코드블록

이곳에서 읽거나

쓰기를 한 후

그냥 코드를 빠져나가면 됩니다.

"파일객체 = `open(파일이름)`"
와 같다고 생각하면 됩니다.

with 문 덕분에 `close()`를 하지 않아도 됩니다.

```
with open('test.txt', 'r') as file:  
    msg = file.read(10)  
    print(msg)
```

바이너리 파일

- ❖ 바이너리 파일은 바이트 단위로 데이터를 읽고 쓰는 것
- ❖ 바이너리 파일은 문자열을 기록할 수 없고 byte 단위로만 기록해야 합니다.
- ❖ 문자열의 경우는 encode 함수를 이용해서 byte로 변형 할 수 있고 바이트를 문자열로 변환 할 때는 decode 함수를 이용하면 됩니다.

```
f = open('test.bin', 'wb')  
f.write("안녕하세요".encode())  
f.close()  
f = open('test.bin', 'rb')  
byteAr = f.read()  
#print(byteAr)  
print(byteAr.decode())
```

문자열을 바이트 단위로 기록했으므로 읽을 때는 문자열로 변환해서 읽어야 합니다.

임의접근파일

- ❖ 파일 포인터의 위치를 임의의 위치로 옮기기 위한 메소드
 - seek(n): n번째 바이트로 이동
 - seek(n, os.SEEK_CUR): 현재 위치에서 n번째 바이트로 이동하는데 이진 파일에서만 가능
 - seek(n, os.SEEK_END): 맨 마지막에서 n번째 바이트로 이동
 - tell(): 현재 파일 포인터의 위치를 리턴

임의접근파일

```
import os
f = open('t.txt', 'wb+')
s = b'01234567890123456789'
f.write(s)
f.seek(5)
print("현재위치:{0}".format(f.tell()))
#1byte 읽기
print("읽은 값:{0}".format(f.read(1)))
f.seek(-3, os.SEEK_END)
print("읽은 값:{0}".format(f.read(1)))
f.close()
```

현재위치:5
읽은 값:b'5'
읽은 값:b'7'

출력방향전환

❖ print의 매개변수로 file에 파일 객체를 대입하면
콘솔에 출력하지 않고 파일에 출력

```
with open('test.txt', 'w') as f:  
    print('line 1', file=f)  
    print('line 2', file=f)  
with open('test.txt') as file:  
    print(file.read())
```

line 1
line 2

Serializable

❖객체를 파일에 저장하는 것

- 피클링 모듈이나 DBM 관련 모듈을 이용
- 피클링 모듈은 임의의 파이썬 객체를 저장하는 가장 일반화된 모듈
- 파일에 내용을 기록하는 경우
 - ✓`pickle.dump(출력할 객체, 파일객체)`
- 파일에서 내용을 읽어오는 경우
 - ✓`pickle.load(파일객체)` => 객체를 1개씩 읽기
 - ✓`Pickle.loads(파일객체)` => 객체 전체를 바이트 단위로 읽기

Serializable

```
class Dto:
    def setNum(self, num):
        self.num = num
    def setName(self, name):
        self.name = name
    def getNum(self):
        return self.num
    def getName(self):
        return self.name
    def toString(self):
        return "{번호:" + str(self.num) + ",이름:" + self.name +
        "}"
```

Serializable

```
data1 = Dto()
data1.setNum(1)
data1.setName("park")
data2 = Dto()
data2.setNum(2)
data2.setName("kim")
li = [data1, data2]
import pickle
with open('test.txt', 'wb') as f:
    pickle.dump(li, f)

with open('test.txt', 'rb') as f:
    result = pickle.load(f)
    for temp in result:
        print(temp.toString())
```

{번호:1,이름:park}
{번호:2,이름:kim}

파일 및 디렉토리 다루기

- ❖ 파일 및 디렉토리를 다룰 때는 os 모듈을 이용합니다.
- ❖ os 모듈의 `listdir()`을 이용하면 지정된 디렉토리에서 전체 파일 목록을 리스트로 가져옵니다.
- ❖ os 모듈의 `isfile`, `isdir`, `islink` 함수는 파일, 디렉토리, 링크 여부를 판단해서 리턴하는 함수
- ❖ 파일에 대한 정보는 `stat` 함수 이용
- ❖ 작업 중 디렉토리 변경은 `chdir()`
- ❖ 디렉토리 생성은 `mkdir()`, 디렉토리 삭제는 `rmdir()`인데 디렉토리 내에 데이터가 있으면 삭제되지 않습니다.
- ❖ 디렉토리 내의 데이터를 전부 삭제하려면 `rmtree()`

파일 및 디렉토리 다루기

```
import os
curlist = os.listdir("./")
print("현재 디렉토리내의 파일 목록")
for name in curlist:
    print(name)

#현재 디렉토리 정보 확인
print("현재 디렉토리내 정보")
print(os.stat("./"))

#디렉토리 생성
os.mkdir("temp")
```

파일 및 디렉토리 다루기

- ❖ 상대경로를 절대경로로 변경하는 함수는 `abspath`
- ❖ 경로의 존재여부는 `exists`
- ❖ 현재 디렉토리는 `curdir`
- ❖ 부모 디렉토리는 `pardir`
- ❖ 디렉토리 분리 문자는 `sep`
- ❖ 파일명 분리는 `basename` 이고 디렉토리 경로 분리는 `dirname` 함수
- ❖ 디렉토리 경로와 파일명을 분리해서 보관하고자 하면 `splitdrive()`
- ❖ 디렉토리 경로와 확장자명을 분리해서 보관하고자 하면 `splittext()`

파일 압축

❖ zip 파일 압축은 zipfile 모듈 사용

- ✓ ZipFile 이라는 함수로 압축 객체를 생성하고 write 함수를 이용해서 하나씩 압축
- ✓ 압축 해제는 압축 파일을 가지고 ZipFile을 만든 후 extractall()을 호출하면 됩니다.

❖ tar 파일 압축은 tarfile 모듈 사용

- ✓ open 함수를 이용해서 압축 객체를 만든 후 add 함수를 이용해서 파일을 추가
- ✓ 압축 해제는 압축 파일 이름을 가지고 open 함수로 객체를 생성한 후 extractall()을 호출

파일 압축

```
import zipfile
filelist = ["c:\\\\test.txt"]
with zipfile.ZipFile('test.zip', 'w',
compression=zipfile.ZIP_BZIP2) as myzip:
    for temp in filelist:
        myzip.write(temp)
zipfile.ZipFile('test.zip').extractall()
```

```
import tarfile
filelist = ["c:\\\\test.txt"]
with tarfile.open('test.tar.gz', 'w:gz') as mytar:
    for temp in filelist:
        mytar.add(temp)
tarfile.open('test.tar.gz').extractall()
```

파일 입.출력

❖파일 입출력

1. csv 파일 읽기
2. csv 파일 저장
3. excel 파일 읽기
4. excel 파일 저장
5. xml 파일 읽기
6. json 파일 읽기

파일 입.출력

❖ pandas로 csv 파일 읽기

csv_read_pd.py

```
import pandas as pd
```

```
# utf-8로 저장된 CSV 파일 읽기  
filename = "list-utf8.csv"
```

```
csv = pd.read_csv(filename, encoding='utf-8')  
print(csv)  
# print(csv['이름'])  
# print(csv['가격'])
```

파일 입.출력

❖ pandas로 csv 파일 저장

csv_write_pd.py

```
import pandas as pd
```

```
data = [[1, 2, 3, 4], [5, 6, 7, 8]]
```

```
dataframe = pd.DataFrame(data)
```

```
dataframe.to_csv("dataframe.csv", header=False, index=False)
```

```
print("저장 되었습니다.")
```

파일 입.출력

❖ excel 파일 읽기

excel_read.py

```
import openpyxl

# 엑셀 파일 열기
filename = "stats_104102.xlsx"
book = openpyxl.load_workbook(filename)

# 맨 앞의 시트 추출하기
sheet = book.worksheets[0]
# 시트의 각 행을 순서대로 추출하기
data = []
for row in sheet.rows:
    data.append([
        row[0].value,
        row[9].value
    ])
])
```

파일 입.출력

필요없는 줄(헤더, 연도, 계) 제거하기

```
del data[0]
```

```
del data[1]
```

```
del data[2]
```

데이터를 인구 순서로 정렬합니다.

```
data = sorted(data, key=lambda x:x[1])
```

하위 5위를 출력합니다.

```
for i, a in enumerate(data):
```

```
    if (i >= 5): break
```

```
    print(i+1, a[0], int(a[1]))
```

파일 입.출력

❖ pandas로 excel 파일 읽기

pandas, xlrd 모듈 설치

excel_read_pd.py

```
import pandas as pd
```

```
# 엑셀 파일 열기
```

```
filename = "stats_104102.xlsx"
```

```
# 파일 이름
```

```
sheet_name = "stats_104102"
```

```
# 시트 이름
```

```
book = pd.read_excel(filename, sheetname=sheet_name, header=1)
```

```
# 첫 번째 줄부터 헤더
```

```
# 2015년 인구로 내림차순 정렬
```

```
book = book.sort_values(by=2015, ascending=False)
```

```
print(book)
```

파일 입.출력

❖ excel 파일 저장

excel_write.py

```
import openpyxl
```

```
# 엑셀 파일 열기
```

```
filename = "stats_104102.xlsx"
```

```
book = openpyxl.load_workbook(filename)
```

```
# 활성화된 시트 추출하기
```

```
sheet = book.active
```

파일 입.출력

```
# 서울을 제외한 인구를 구해서 쓰기
for i in range(0, 9):
    total = int(sheet[str(chr(i + 66)) + "3"].value)        # B3 ~ J3 셀
    seoul = int(sheet[str(chr(i + 66)) + "4"].value)        # B4 ~ J4 셀
    output = total - seoul
    print("서울 제외 인구 =", output)

# 쓰기 : B21 ~ J21 셀에 출력
sheet[str(chr(i + 66)) + "21"] = output
cell = sheet[str(chr(i + 66)) + "21"]

# 폰트와 색상 변경해보기
cell.font = openpyxl.styles.Font(size=14,color="FF0000")
cell.number_format = cell.number_format

# 엑셀 파일 저장하기
filename = "population.xlsx"
book.save(filename)
print("ok")
```

파일 입.출력

❖ xml파일 읽기

xml_forecast.py

```
from bs4 import BeautifulSoup
import urllib.request as req
import os.path
```

```
url = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108"
savename = "forecast.xml"
```

```
if not os.path.exists(savename):
    req.urlretrieve(url, savename)
```

```
# BeautifulSoup로 분석하기
xml = open(savename, "r", encoding="utf-8").read()
soup = BeautifulSoup(xml, 'html.parser')
```


파일 입.출력

```
# 각 지역 확인하기
```

```
info = {}
```

```
for location in soup.find_all("location"):
```

```
    name = location.find('city').string
```

```
    wf = location.find('wf').string
```

```
    tmx = location.find('tmx').string
```

```
    tmn = location.find('tmn').string
```

```
# 도시명
```

```
# 날씨
```

```
# 최고기온
```

```
# 최저기온
```

```
weather = wf + ':' + tmn + ' ~ ' + tmx
```

```
if name not in info:
```

```
    info[name] = []
```

```
info[name].append(weather)
```

```
# info = { name : weather }
```

```
# 각 지역의 날씨를 구분해서 출력하기
```

```
for name in info.keys():
```

```
    print("+", name)
```

```
    for weather in info[name]:
```

```
        print("| - ", weather)
```

JSON

❖ JSON (JavaScript Object Notation)

- JSON은 JavaScript Object Notation 의 약어로 XML 데이터를 대신 하여 데이터 교환용으로 많이 사용되는 문서포맷이다.
- JSON은 키와 값을 쌍으로 가지는 구조이다.
- 배열을 사용할 때는 대괄호([])안에 중괄호({ })를 사용하여 조합한다.

JSON

❖ JSON (JavaScript Object Notation)

```
[
  {
    "id": "1",
    "name": "레몬",
    "price": " 3000",
    "description": "레몬에 포함되어 있는 쿠엔산은 피로회복에 좋다. 비타민C도 풍부하다."
  },
  {
    "id": "2",
    "name": "키위",
    "price": " 2000",
    "description": "비타민C가 매우 풍부하다. 다이어트와 미용에도 매우 좋다."
  },
  {
    "id": "3",
    "name": "블루베리",
    "price": " 5000",
    "description": "블루베리에 포함된 anthocyanin(안토시아닌)은 눈피로에 효과가 있다."
  }
]
```

파일 입.출력

❖ json 파일 읽기

json_github.py

```
import urllib.request as req
import os.path, random
import json

# JSON 데이터 내려받기
url = "https://api.github.com/repositories"
savename = "repo.json"
if not os.path.exists(savename):
    req.urlretrieve(url, savename)

# JSON 파일 분석하기
items = json.load(open(savename, "r", encoding="utf-8"))

# 출력하기
for item in items:
    print(item["name"] + " - " + item["owner"]["login"])
```

파일 입.출력

❖ json 파일 읽기

json_items.py

```
import json
```

```
# json파일 읽기
```

```
items = json.load(open('data/item.json', 'r', encoding='utf-8'))
```

```
print(type(items))          # 'list'
```

```
print(items)                # [{'id': '1', 'name': '레몬', 'price': ' 3000', ...
```

```
for item in items:
```

```
    print(item['id']+'-'+item['name']+'-'+item['price']+'-'+item['description'])
```