ML&DL

WITH PYTHON

PART Ⅲ

DATA 가공 - File

DATA TYPE

◆바이너리 & 텍스트

[데이터 분류]

데이터 타입	장점	단점
텍스트	- 텍스트 편집기로 편집 가능 - 데이터 설명 추가 가능	- 바이너리에 비해 크기가 큼 - 문자 인코딩 주의 (대부분 UTF-8)
바이너리	- 텍스트 데이터에 비해 크기 작음 - WEB에서 사용되는 데이터	- 텍스트 편집기로 편집 불가 - 데이터 설명 추가 불가

[텍스트 데이터 파일]

파일	특징
XML 파일	- 범용적인 형식, 웹 API 활용 형식
JSON 파일	- 자바스크립트 객체 표기 방법 기반 형식 파일 - 데이터 교환에 활용
YAML	- JSON 대용으로 사용, 어플리케이션 설정 파일에 많이 사용되는 파일
CSV/TSV	- WEB상에서 많이 사용되는 파일

CH01. 데이터 파일 - XML

DATA FILE - XML

◆XML

- eXtensible Markup Language
- 특정 목적에 따라 데이터를 태그로 감싸는 언어
- W3C에서 웹 표준 재정
- HTML과 달리 **사용자 정의 태그로 데이터 표현**

DATA FILE - XML

◆XML

- 데이터 표현 => 계층 구조

형태 : <요소 속성='속성값'>내용</요소>

```
<product id='101010' price='5000'>SD카드</product>

<products type=''전자제품'>
    <product id='101010' price='5000'>SD카드</product>
    <product id='101011' price='9900'>키보드</product>
    <product id='101012' price='3500'>패드</product>
</ products >
```

DATA FILE - XML

◆XML – Text Data & UTF-8

```
#모듈 로딩---------------------------------------- xml_read.py
from bs4 import BeautifulSoup
import urllib.request as req
import os.path

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

# WEB 데이터 다운로드 후 파일 저장 ----------------------------------
if not os.path.exists(savename):
    req.urlretrieve(url, savename)

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

DATA FILE - XML

◆XML

```
# WEB 데이터 정보 추출 ----------------------------------------
info = {}
for location in soup.find_all("location"):
    name = location.find('city').string
    weather = location.find('wf').string

    if not (weather in info):
        info[weather] = []
    info[weather].append(name)

# 각 지역의 날씨를 구분해서 출력 ----------------------------------------
for weather in info.keys():
    print("+", weather)
    for name in info[weather]:
        print("| - ", name)
```

CH02. 데이터 파일 - JSON

DATA FILE - JSON

◆JSON

- JavaScript Object Notation 약자
- 자바스크립트에서 사용하는 객체 표기 방법
- 다양한 프로그래밍 언어에서 **데이터 교환**에 사용
- 인코딩/디코딩 표준으로도 사용

DATA FILE - JSON

◆JSON

Python 표준 라이브러리 – JSON
https://docs.python.org/3.7/library/json.html

- Internet Data Handling
  - `email` — An email and MIME handling package
  - `json` — JSON encoder and decoder
  - `mailcap` — Mailcap file handling
  - `mailbox` — Manipulate mailboxes in various formats

DATA FILE - JSON

◆JSON

- **JSON 인코딩**

- python List, Tuple, Dic 등 **객체 타입을 JSON 문자열**로 **변환**

JSON문자열 = **json.dumps(** Python Object **)**

JSON문자열 = **json.dumps(** Python Object, indent=n **)**

DATA FILE - JSON

◆JSON

- JSON 디코딩

- **JSON 문자열을 Python 객체 타입**으로 **변환**

Python Object = **json.loads(** JSON 문자열 **)**

DATA FILE – JSON

◆JSON

- 데이터 자료형

자료형	표현방법	예시
숫자	숫자	20
문자열	큰 따옴표로 감싸 표현	“good"
boolean	true / false	true
배열	[n1,n2,n3,...]	[1,2,10,100]
객체	{"key":value,"key":value,"key":value,...}	{"a":100,"b":10}
null	null	null

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

DATA FILE – JSON

◆JSON

- 데이터 표현 => **계층 구조**

형태 : [{ 키:값, 키:값, 키:{ 키:값, 키:값 } }]

```
[ {   'id':1,
      'name':'jane',
      'data' : { 'major':'science',
                 'grade': 1 }
  }, { 'id':2,
       'name':'Tom',
       'data' : { 'major':'math',
                  'grade': 2 }
  } ]
```

DATA FILE - JSON

◆JSON

```
# 모듈로딩----------------------------------------
import urllib.request as req
import os.path
import json

# 번수 선언 ----------------------------------------
url = "https://api.github.com/repositories"
savename = "../DATA/repo.json"

# WEB 데이터 json 파일 다운로드 ----------------------
if not os.path.exists(savename):
    req.urlretrieve(url, savename)

# JSON 파일 분석 ------------------------------------
s = open(savename, "r", encoding="utf-8").read()
items = json.loads(s)

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

DATA FILE - JSON

◆JSON

```
# 모듈로딩----------------------------------------
import urllib.request as req
import os.path
import json

# 번수 선언 ----------------------------------------
url = "https://api.github.com/repositories"
savename = "../DATA/repo.json"

# WEB 데이터 json 파일 다운로드 ----------------------
if not os.path.exists(savename):
    req.urlretrieve(url, savename)

# JSON 파일 분석 ----------------------------------
items = json.loads( open(savename, "r", encoding="utf-8").read() )

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

DATA FILE - JSON

◆JSON

```
# 모듈로딩----------------------------------------
import json

# 번수 선언 ----------------------------------------
data={
    'date':'2020-01-01',
    'price':{
        'apple':500,
        'banana':2500
    }}
savename = "../DATA/jdata.json"

# JSON형식 저장 ----------------------------------------
jdata = json.dumps(data)

# JSON 파일 생성 ----------------------------------------
with open(savename,mode='w', encoding='utf-8') as file:
    file.write(jdata)
```

CH03. 데이터 파일 - YAML

DATA FILE - YAML

◆YAML

- Yet Another Markup Language 약자
- **공백문자 들여쓰기**를 사용해 **계층 구조 표현**
- XML보다 간단, JSON과 비슷
- XML, C, 파이썬, 펄, RFC2822 정의된 e-mail 양식 개념 활용
- **모든 데이터를 리스트, 해쉬, 스칼라 데이터 조합으로 표현**
- JSON대용으로 어플리케이션 **설정 파일에 많이 사용**

DATA FILE - YAML

◆YAML

- 설치 ➔ pip install PyYAML

PyYAML

PyYAML is a YAML parser and emitter for Python.

Overview

YAML is a data serialization format designed for human readability and interaction with scripting languages.

PyYAML is a YAML parser and emitter for the Python programming language.

PyYAML features

- a *complete* YAML 1.1 parser. In particular, PyYAML can parse all examples from the specification. The parsing algorithm is simple enough to be a reference for YAML parser implementors.
- Unicode support including UTF-8/UTF-16 input/output and *
- low-level event-based parser and emitter API (like SAX).
- high-level API for serializing and deserializing native Python objects (like DOM or pickle).
- support for all types from the YAML types repository. A simple extension API is provided.
- both pure-Python and fast LibYAML-based parsers and emitters.
- relatively sensible error messages.

- https://pyyaml.org/wiki/PyYAML

DATA FILE - YAML

◆YAML

- YAML 인코딩

- python List, Tuple, Dic 등 **객체 타입을 YAML 문자열**로 **변환**

YAML문자열 = **yaml.dumps(** Python Object **)**

DATA FILE - YAML

◆YAML

- YAML 디코딩

- **YAML 문자열을 Python 객체 타입**으로 **변환**

Python Object = **yaml.loads(** YAML 문자열 **)**

DATA FILE – YAML

◆YAML

- 데이터 표현 => **하이픈(-) 행 표시 , 들여쓰기 계층 구조**

```
[ 배열 데이터 ]
-   amburger
-   blog
-   home

[ 중첩 배열 데이터 ]
-   menu
-
        -   copy
        -   cut
        -   sudo
```

```
[ 해시 데이터 ]
name : kkk
age : 12
grade : 2

[ 해시 데이터 계층 구조]
id : 1234
info :
        name : kkk
         age : 12
         grade : 2
```

DATA FILE – YAML

◆YAML

```
# 모듈 로딩 ------------------------------------
import yaml

# 데이터 변수 선언 ----------------------------------
yaml_str = """
Date: 2017-03-10
PriceList:
    -
        item_id: 1000
        name: Banana
        color: yellow
        price: 800
    -
        item_id: 1001
        name: Orange
        color: orange
        price: 1400
"""
```

DATA FILE – YAML

◆YAML

```
# YAML 분석 ----------------------------------------
data = yaml.load( yaml_str, Loader=yaml.FullLoader )


# 이름, 가격 데이터 출력 ----------------------------
for item in data['PriceList']:
    print(item["name"], item["price"])
```

DATA FILE – YAML

◆YAML

```
# 모듈 로딩 ------------------------------------- yaml_convert.py
import yaml

# 데이터 변수 ----------------------------------------------
# 사용자 정보 리트스 데이터
customer = [
    { "name": "InSeong", "age": "24", "gender": "man" },
    { "name": "Akatsuki", "age": "22", "gender": "woman" },
    { "name": "Harin", "age": "23", "gender": "man" },
    { "name": "Yuu", "age": "31", "gender": "woman" }
]

# YAML 데이터 생성 -----------------------------------
yaml_str = yaml.dump( customer )
print(" Python LIST => YAML DATA")
print( yaml_str )
```

DATA FILE – YAML

◆YAML

```
# YAML 데이터를 Python 데이터 변환
data = yaml.load( yaml_str, Loader=yaml.FullLoader )
print(" Python LIST => YAML DATA")
print('type(data) => ', type(data))
for p in data:
    print(p)
```

CH04. 데이터 파일 – CSV/TSV/SSV

DATA FILE – CSV/TSV/SSV

◆CSV/TSV/SSV

- Common/Tab/Space Separated Values 약자
- 콤마, 탭, 공백으로 데이터 구분하는 방식
- 데이터 표현에 많이 사용되는 형식

DATA FILE – CSV/TSV/SSV

◆CSV/TSV/SSV

- 데이터 표현 => 구분자, 줄바꿈으로 구성된 데이터 구조

형태 : 필드, 필드, 필드 (줄바꿈 CTRL, U+00D, U+000A)

list-euckr.csv

```
ID,이름,가격

1000,비누,300

1001,장갑,150

1002,마스크,230
```

DATA FILE – CSV/TSV/SSV

◆CSV/TSV/SSV

codecs

- https://docs.python.org/ko/3.7/library/codecs.html
- 텍스트 데이터 인코딩 & 디코딩 표준 라이브러리

csv

- https://docs.python.org/ko/3.7/library/csv.html?highlight=csv
- csv 파일 처리 표준 라이브러리

DATA FILE – CSV/TSV/SSV

◆CSV/TSV/SSV

```
# 모듈 로딩 ----------------------------------------
import codecs

# 데이터 변수 선언 ----------------------------------
filename = "../DATA/list-euckr.csv"
csv = codecs.open(filename, "r", "euc_kr“).read()

# CSV을 파이썬 리스트로 변환 ------------------------
data = []
rows = csv.split("₩r₩n")      # 레코드 구분 줄바꿈
for row in rows:
    if row == "": continue
    cells = row.split( ",“ )      # 필드 데이터 구분
    data.append(cells)

# 결과 출력하기 -------------------------
for c in data:
    print(c[1], c[2])
```

DATA FILE – CSV/TSV/SSV

◆CSV/TSV/SSV

```
# 모듈 로딩 ----------------------------------------------
import csv, codecs

# CSV 파일 생성 ------------------------------------------
# encoding = euc_kr,  utf-8
with codecs.open( "../DATA/test_02.csv", "w", "utf-8“ ) as fp:
    writer = csv.writer( fp, delimiter=",", quotechar='"‘ )
    writer.writerow( ["ID", "이름", "가격"] )
    writer.writerow( ["1000", "SD 카드 ", 30000] )
    writer.writerow( ["1001", "키보드", 21000] )
    writer.writerow( ["1002", "마우스", 15000] )
```

CH05. 데이터 파일 – XLSX

DATA FILE – XLSX

◆XLSX

- 설치

```
pip install openpyxl == 3.0.1
```

- 라이브러리

https://pypi.org/project/openpyxl/3.0.1/

DATA FILE – XLSX

◆XLSX

```
# 모듈로딩 ------------------------------------------------
from openpyxl import Workbook


# 데이터 변수 선언 ----------------------------------------
filename = 'sample.xlsx'


# 엑셀 파일 생성 ------------------------------------------
wb = Workbook()                    # 엑셀 파일 생성, Sheet 자동 생성
ws = wb.active                     # 시트 활성화
ws.title = 'new sheet'             # 시트명 변경
ws['A1'] = 'Language'              # 시트 데이터 삽입
ws['B1'] = 'Create'


wb.save( filename = filename )     # 엑셀 파일 생성
```

DATA FILE – XLSX

◆XLSX

```
# 모듈로딩 ------------------------------------------------
from openpyxl import Workbook

# 데이터 변수 선언 ----------------------------------------
filename = 'sample.xlsx'

# 엑셀 파일 생성 ------------------------------------------
wb = Workbook()                          # 엑셀 파일 생성, Sheet 자동 생성
ws = wb.active                           # 시트 활성화
ws.title = 'new sheet'                   # 시트명 변경
ws['A1'] = 'Language'                    # 시트 데이터 삽입
ws['B1'] = 'Create'

#wb.remove_sheet(ws)                     # 1번째 시트 삭제
ws2=wb.create_sheet('data_sheet')        # 2번째 시트 생성
ws2.cell(1,1).value='Name'
ws2.cell(1,2).value='Phone'
ws2.cell(2,1).value='Tom'
ws2.cell(2,2).value='010-111-2222'
wb.save( filename = filename )           # 엑셀 파일 생성
```

DATA FILE – XLSX

◆XLSX

```
# 모듈 로딩 ----------------------------------------------
import openpyxl

# 데이터 변수 선언 ----------------------------------------
filename = "sample.xlsx"

# 엑셀 파일 열기 ------------------------------------------
book = openpyxl.load_workbook( filename )

# 시트 추출
sheet = book.worksheets[1]
sheetnames = book.sheetnames;
print('sheet =>',  sheet.title)
print('sheet.max_row = {}, sheet.max_column={}'.format(sheet.max_row,
sheet.max_column))
```

DATA FILE – XLSX

◆XLSX

```
rowdata=[]
for row in sheet.rows:
    rowdata.append([
        row[0].value,
        row[sheet.max_column-1].value
    ])

print('len(rowdata) => ', len(rowdata), rowdata)
```

DATA FILE – XLSX

◆XLSX

```
# 모듈 로딩 ----------------------------------------------------------
import openpyxl

# 데이터 변수 선언 ---------------------------------------------------
filename = "../DATA/stats_100701.xlsx"

# 엑셀 파일 특정 데이터 추출 -----------------------------------------
book = openpyxl.load_workbook(filename)
sheet = book.worksheets[0]

# 엑셀에서 데이터만 추출
data = []
skip = 1
for row in sheet.rows:
    if skip > 4:
        data.append([
            row[0].value,
            row[sheet.max_column-2].value
        ])
    skip += 1
```

DATA FILE – XLSX

◆XLSX

```
# 데이터를 인구 순서로 정렬
data = sorted(data, key=lambda x:x[1])

# 하위 5위 출력
for i, a in enumerate(data):
    if (i >= 5): break
    print(i+1, a[0], int(a[1])
```

DATA FILE – XLSX

◆XLSX

```
# 모듈 로딩 ---------------------------------------
import openpyxl

# 데이터 변수 선언 ---------------------------------
filename = "../DATA/stats_100701.xlsx"
B_CODE = 66

# 엑셀 특정 데이터 추출 -----------------------------
book = openpyxl.load_workbook(filename)
sheet = book.active
MAX_COL = sheet.max_column
MAX_ROW = sheet.max_row

# 서울 제외 인구 추출 파일 저장 ---------------------------
for i in range(0, MAX_COL-1):
    total = int(sheet[str(chr(i + B_CODE)) + "4"].value)
    seoul = int(sheet[str(chr(i + B_CODE)) + "5"].value)
    output = total - seoul
    print("서울 제외 인구 =", i, output)
```

DATA FILE – XLSX

◆XLSX

```
    # 새로운 ROW 추가
    sheet[str(chr(i + B_CODE)) + str(MAX_ROW+1)] = output
    cell = sheet[str(chr(i + B_CODE)) + str(MAX_ROW+1)]
    cell.font = openpyxl.styles.Font(size=14,color="FF0000")
    cell.number_format = cell.number_format

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

PART Ⅲ

DATA 가공 - DB

CH01. SQLite DB

DATABASE

◆SQLite

- 내장 가능한 Open Source 데이터베이스
- 다양한 플랫폼 이식 사용 가능
- C언어로 작성되었고 SQL 쿼리 가능
- 안드로이드 폰, iOS 폰에 사용되는 가벼운 데이터베이스
- https://www.sqlite.org/index.html

- **파일 하나 ➔ 데이터베이스**
- 전용 어플리케이션 사용 필요 없음
- **python에 sqlite3 표준라이브러리 사용**

DATABASE

◆SQLite

(1) 데이터베이스 연결 **conn = sqlite3.connect( )**

(2) 테이블 제어 커서 생성 **cursor= conn.cursor()**

(3) SQL문 실행 **ursor.executescript(“””SQL”””)**

cursor.execute(””” SQL문”””)

(4) 데이터베이스 반영 **conn.commit()**

DATABASE

◆SQLite

```
# 모듈 로딩 ------------------------------------------------
import sqlite3

# 데이터 변수 선언 ------------------------------------------
# sqlite 데이터베이스 연결
dbpath = "test.sqlite"

# 데이터베이스 생성 및 제어 ----------------------------
# (1)데이터 베이스 생성 및 연결
conn = sqlite3.connect(dbpath)

# (2) 테이블 생성 및 데이터 넣기
cur = conn.cursor()
cur.executescript("""
/* items 테이블이 이미 있다면 제거하기 */
DROP TABLE IF EXISTS items;
```

◆SQLite

```
/* 테이블 생성하기 */
CREATE TABLE items(
    item_id INTEGER PRIMARY KEY,
    name TEXT UNIQUE,
    price INTEGER
);
/* 데이터 넣기 */
INSERT INTO items(name, price)VALUES('Apple', 800);
INSERT INTO items(name, price)VALUES('Orange', 780);
INSERT INTO items(name, price)VALUES('Banana', 430);
""")

# (3) 데이터베이스 반영
conn.commit()
```

DATABASE

◆SQLite

```
# 데이터 추출 ----------------------------------------
# (1) 데이터베이스 접근 포인트 즉 커서 획득
cur = conn.cursor()

# (2) SQL명령문 실행
cur.execute("SELECT item_id,name,price FROM items")

# (3) SQL명령문 실행 결과
item_list = cur.fetchall()  # 모든 데이터 가져오기 (튜플타입)
# 출력
for it in item_list:
    print(it)
```

◆SQLite

```
#모듈 로딩 ----------------------------------
import sqlite3

# 데이터 변수 선언 ----------------------------
filepath = "test2.sqlite"

# 데이터베이스 제어 ----------------------------
# (1) 데이터베이스 연결
conn = sqlite3.connect(filepath)

# (2) 테이블 생성 및 데이터 넣기
cur = conn.cursor()
cur.execute("DROP TABLE IF EXISTS items")
cur.execute("""CREATE TABLE items (
    item_id INTEGER PRIMARY KEY,
    name    TEXT,
    price   INTEGER)""")

# (3) 변경 내용 적용
conn.commit()
```

◆SQLite

```
# 데이터 넣기 ----------------------------------
# (1) 데이터베이스 접근 포인트(커서) 획득
cur = conn.cursor()

# (2) 데이터 삽입
cur.execute(
    "INSERT INTO items (name,price) VALUES (?,?)",
    ("Orange", 5200))

# (3) 변경 내용 적용
conn.commit()

# 여러 데이터 연속으로 넣기 ---------------------------
# (1) 데이터베이스 접근 포인트(커서) 획득
cur = conn.cursor()

# (2) 데이터 리스트
data = [("Mango",7700), ("Kiwi",4000), ("Grape",8000),
    ("Peach",9400),("Persimmon",7000),("Banana", 4000)]
```

DATABASE

◆SQLite

```
# (3) 데이터 넣기
cur.executemany(
    "INSERT INTO items(name,price) VALUES (?,?)",
    data)

# (4) 변경 정보 설정
conn.commit()

# 4000-7000원 사이의 데이터 추출
cur = conn.cursor()
price_range = (4000, 7000)
cur.execute(
    "SELECT * FROM items WHERE price>=? AND price<=?",
    price_range)

fr_list = cur.fetchall()
for fr in fr_list:
    print(fr)
```

CH02. MySQL DB

DATABASE

◆MySQL

- 세계에서 가장 많이 쓰이는 오픈 소스 관계형데이터베이스관리 시스템
- 오라클이 관리 및 지원하며 유료
- 유료 정책에 반하여 MySQL개발자가 => MariaDB 개발 및 배포

- 운영체제별 관리 전용 어플리케이션 설치 필수

DATABASE

◆MySQL

```
# 모듈로딩-----------------------
import MySQLdb

# MySQL 데이터베이스 제어 ----------------------
# (1) MySQL 연결
conn = MySQLdb.connect(
    user='root',
    passwd='test-password',
    host='localhost',
    db='test')

# (2) 데이터베이스 제어
# (2-1)데이터베이스 접근 커서 추출
cur = conn.cursor()
```

DATABASE

◆MySQL

```
# (2-2) 테이블 생성
cur.execute('DROP TABLE items')
cur.execute('''
    CREATE TABLE items (
        item_id INTEGER PRIMARY KEY AUTO_INCREMENT,
        name TEXT,
        price INTEGER
    )
    ''')

# (2-3)데이터 추가
data = [('Banana', 300),('Mango', 640), ('Kiwi', 280)]
for i in data:
    cur.execute("INSERT INTO items(name,price) VALUES(%s,%s)", i)
```

DATABASE

◆MySQL

```
# (2-4) 데이터 추출
cur.execute("SELECT * FROM items")
for row in cur.fetchall():
    print(row)
```