UNISTUDY

파이썬

30강. 파일 자료처리

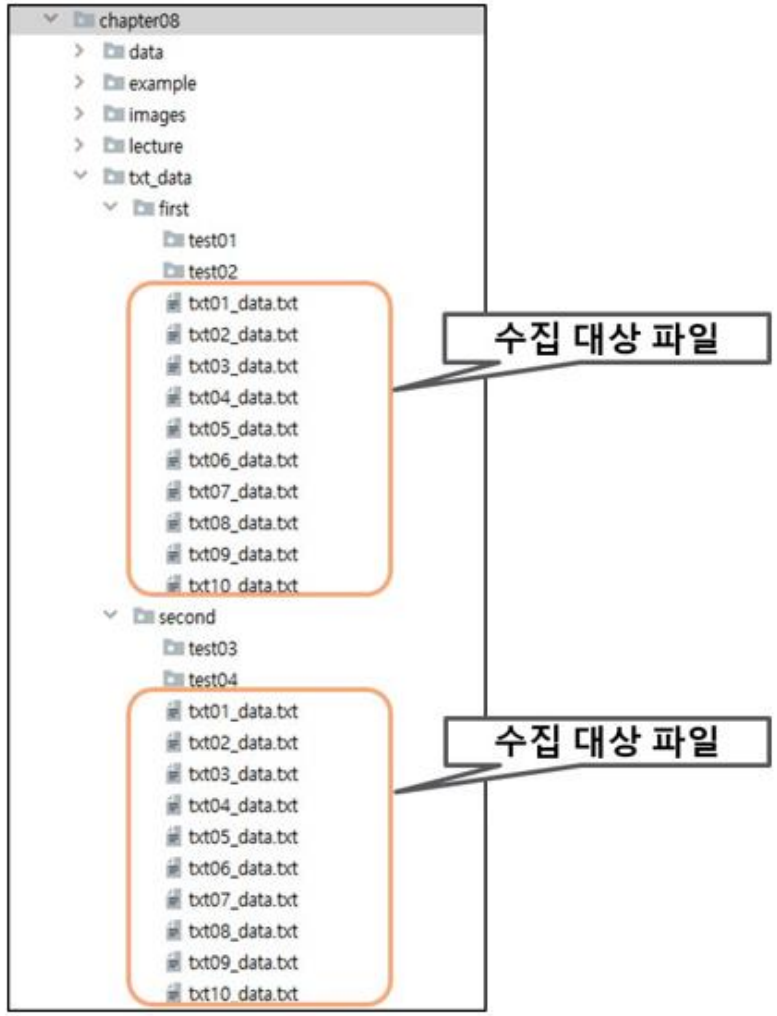
1. 파일 자료 처리

- 텍스트 파일과 파일 시스템의 관련 함수들을 이용하여 특정 디렉터 리에 있는 여러 개의 목록을 대상으로 텍스트 파일의 자료를 수집하는 방법, 특정 이미지 파일만 선택 하는 방법 그리고 이진파일(binary file)로 저장하는 방법 등에 대해서 알아본다.

2. 텍스트 자료 수집

- 특정 디렉터리에서 텍스트 파일만 선택하여 자료를 수집하려면 어떻게 해야 할까? 디렉터리에는 여러 유형의 파일과 하위 디렉터리를 포함하고 있기 때문에 사용자가 원하는 것이 디렉터리인지 파일인지를 먼저 구분하고 파일이면 어떤 유형의 파일인지를 구분한 이후에 다음 과정의 작업을 적절하게 수행해 야 한다. 텍스트 파일로부터 텍스트 자료를 수집하기 위한 작업환경이다.

2. 텍스트 자료 수집


chapter08
data
example
images
lecture
txt_data
first
test01
test02
txt01_data.txt
txt02_data.txt
txt03_data.txt
txt04_data.txt
txt05_data.txt
txt06_data.txt
txt07_data.txt
txt08_data.txt
txt09_data.txt
txt10_data.txt
수집 대상 파일
second
test03
test04
txt01_data.txt
txt02_data.txt
txt03_data.txt
txt04_data.txt
txt05_data.txt
txt06_data.txt
txt07_data.txt
txt08_data.txt
txt09_data.txt
txt10_data.txt
수집 대상 파일

2. 텍스트 자료 수집

- 최상위 디렉터리인 chapter08을 기준으로 ‘chapter08/text_data/’ 디렉터리 경로를 갖는 first와 second 하위 디렉터리에는 각각 2개의 디렉터리와 10개의 텍스트 파일이 존재한다. 다음 예문은 위 와 같은 환경에서 각각의 디렉터리에 저장된 10개의 텍스트 파일만 선택하여 텍스트 자료를 수집한다.

2. 텍스트 자료 수집

chapter08.lecture.step04_text_process.py

```
import os # os모듈 import

# (1) 텍스트 디렉터리 경로 지정
print(os.getcwd()) # 기본 작업 디렉터리
txt_data = 'chapter08/txt_data/' # 상대경로 지정

# (2) 텍스트 디렉터리 목록 반환
sub_dir = os.listdir(txt_data) # txt_data 목록 반환
print(sub dir) # ['first', 'second']

# (3) 각 디렉터리의 텍스트 자료 수집 함수
def textPro(sub_dir): # ['first', 'second']
    first_txt = [] # first 디렉터리 텍스트 저장
    second_txt = []  # second 디렉터리 텍스트 저장



    # (3-1) 디렉터리 구성
    for sdir in sub_dir : # ['first', 'second']
        dirname = txt_data + sdir # 디렉터리 구성
        file_list = os.listdir(dirname) # 파일 목록 반
환
```

Python Console

```
D:\Pywork\workspace
['first', 'second']
```

2. 텍스트 자료 수집

```
        # (3-2) 파일 구성
        for fname in file_list:
            file_path = dirname + '/' + fname # 파일
구성


            # (3-3) file 선택
            if os.path.isfile(file_path) :
                try :
                    # (3-4) 텍스트 자료 수집
                    file = open(file_path, 'r')
d())                if sdir == 'first':
                        first_txt.append(file.rea


ad())               else :
                        second_txt.append(file.re


                except Exception as e:
                    print('예외발생 :', e)
                finally:
                    file.close()
    return first_txt, second_txt # 텍스트 자료 반환
                                                    first_tex 길이 = 10
# (4) 함수 호출
```

2. 텍스트 자료 수집

```
first_txt, second_txt = textPro(sub_dir) # ['first
', 'second']

# (5) 수집한 텍스트 자료 확인
print('first_tex 길이 =', len(first_txt)) # first_te
x 길이 = 10
print('second_tex 길이 =', len(second_txt))
# second_tex 길이 = 10

# (6) 텍스트 자료 결합
tot_texts = first_txt + second_txt
print('tot_texts 길이=', len(tot_texts)) # tot_text
s 길이= 20

# (7)전체 텍스트 내용
print(tot texts)
print(type(tot_texts)) # <class 'list'>
```

```
second_tex 길이 = 10

tot_texts 길이= 20

["maroon thinks tha
t roger's wife , jes
sica's
중간 생략 ...
 s\nbut it doesn\'t
really matter. ']
<class 'list'>
```

2. 텍스트 자료 수집

- 텍스트 자료 수집 예
- # (1) 텍스트 디렉터리 경로 지정
- os.getcwd()함수를 이용하여 기본 작업 디렉터리를 확인한 후 txt_data 디렉터리를 상대경로로 지정한다.
- # (2) 텍스트 디렉터리 목록 반환
- os.listdir()함수를 이용하여 txt_data 디렉터리의 전체 목록을 반환받는다. 반환 결과는
- txt_data 디렉터리의 하위 디렉터리인 'first'와 'second'가 리스트로 반환된다.
- # (3) 각 디렉터리의 텍스트 자료 수집 함수
- ‘first’와 ‘second’ 디렉터리에서 텍스트 파일만 선택하여 텍스트 자료를 수집하기 위한 함수이다.

2. 텍스트 자료 수집

- # (3-1) 디렉터리 구성
- 'first'와 'second' 디렉터리를 리스트의 원소로 갖는 sub_dir 인수를 받아서 '텍스트 디렉터리' 경로와 '하위 디렉터리'를 이용하여 디렉터리 경로를 구성하고 하위 디렉터리에서 전체 목록을 반환받 는다.
- # (3-2) 파일 구성
- 하위 디렉터리에서 전체 목록을 이용하여 파일 경로를 구성한다.
- # (3-3) file 선택
- 파일 경로가 파일인지를 구분한다. if문에서 파일인 경우 True를 반환한다.
- # (3-4) 텍스트 자료 수집
- 파일인 경우 해당 텍스트 파일을 대상으로 파일 객체를 생성하고, 'first' 디렉터리 소속의 파일이면
- first_txt에 추가하고 'second' 디렉터리 소속의 파일이면 second_txt에 추가한다.

2. 텍스트 자료 수집

- # (4) 함수 호출
- 'first'와 'second' 디렉터리를 리스트의 원소로 갖는 sub_dir을 실인수로 하여 textPro() 함수 를 호출한다. 함수에서 반환한 first_txt와 second_txt를 변수로 받는다.
- # (5) 수집한 텍스트 자료 확인
- 'first'와 'second' 디렉터리에 포함된 텍스트 파일 10개에서 수집한 10개의 문장을 len()함수를 이용하여 확인한다.
- # (6) 텍스트 자료 결합
- 'first'와 'second' 디렉터리에 포함된 각각의 텍스트 자료 10개를 하나로 합친다.
- # (7)전체 텍스트 내용
- 하나로 합친 20개의 문장을 확인한다. 리스트와 리스트의 결합은 리스트 자료구조가 된다.

3. pickle 저장

- 텍스트 파일을 저장하기 위해서 open()함수에 mode='w'로 지정하면 저장이 가능하다. 하지만 모든 자료를 텍스트 형식으로만 저장할 수 없다. 예를 들면 텍스트 자료를 수집하여 리스트 자료구조에 저장 한 경우 리스트 변수를 텍스트 파일 형식으로 저장하면 리스트 자료구조가 깨지고 단지 한 줄의 텍스트 형식으로 저장된다. 만약 리스트 자료구조를 갖는 객체를 그대로 파일에 저장하기 위해서 기계어 형식 으로 저장해야 한다. 이러한 파일을 이진파일(binary file)이라고 한다.
- 이미지나 음악, 동영상 파일 등은 모두 이진파일 형태로 저장해야 원본의 내용을 그대로 유지하여 파일 에 보관할 수 있다.
- 파이썬에서 제공하는 pickle 모듈은 이진파일 형식으로 변수에 저장된 객체를 저장하기 때문에 객체 타입을 그대로 유지하여 파일에 저장하고 읽어올 수 있다. 다음 예문은 '텍스트 자료 수집' 예문에서 수 집된 자료를 보관하고 있는 tot_texts 변수를 대상으로 pickle 모듈을 이용하여 이진파일로 저장하 고, 저장된 파일을 읽어오는 과정이다.

3. pickle 저장

chapter08.lecture.step04_text_process.py

```
# (1) pickle모듈 import
import pickle # file save

# (2) file save : write binary
pfile_w = open("chapter08/data/tot_texts.pck
", mode='wb')
pickle.dump(tot_texts, pfile_w)

# (3) file load : read binary
pfile_r = open("chapter08/data/tot_texts.pck
", mode='rb')
tot_texts_read = pickle.load(pfile_r)
print('tot_texts 길이 =', len(tot_texts_read))
# 20
print(type(tot_texts_read)) # <class 'list'>
print(tot_texts_read)
```

Python Console

```
tot_texts 길이 = 20
<class 'list'>
['first', 'second']s
중간 생략 ...
 s\nbut it doesn\'t reall
y matter. ']
```

3. pickle 저장

- pickle 저장과 읽기 예
- # (1) pickle모듈 import
- pickle 모듈의 함수를 이용하기 위해서 import 한다.
- # (2) file save : write binary
- open()함수에서 저장할 파일의 경로와 파일명을 지정하고, mode='wb'를 지정하여 이진파일 형식으 로 쓰기용 파일 객체를 생성한다. 'wb'는 write binary 모드를 의미한다.
- # (3) file load : read binary
- open()함수에서 읽어올 파일의 경로와 파일명을 지정하고, mode='rb'를 지정하여 이진파일 형식으 로 읽기용 파일 객체를 생성한다. 'rb'는 read binary 모드를 의미한다.
- 이진파일로부터 읽어온 결과를 출력하면 저장하기 전 리스트 객체를 유지하고 있음을 알 수 있다.

4. 이미지 파일 이동

- 특정 디렉터리에서 포함된 전체 파일을 대상으로 이미지 파일(*.png)을 선택하여 특정 디렉터리로 이 미지 파일을 옮기는 방법에 대해서 알아 본다. 작업 절차는 다음과 같다.
- 1. os.path.exists(path) : 디렉터리 유무 확인
- 2. glob(path/*.png) : png 파일 검색
- 3. png 이진 파일 읽기
- 4. png 이진 파일 쓰기 & 디렉터리 이동

4. 이미지 파일 이동

chapter08.lecture.step05_image_file_move.py

```
import os # dir or file path
from glob import glob # *, ? 파일 검색

# (1) image 파일 경로
print(os.getcwd()) # D:\Pywork\workspace
img_path = 'chapter08/images/' # 이미지 원본 디렉터리
img_path2 = 'chapter08/images2/' # 이미지 이동 디렉터리

# (2) 디렉터리 존재 유무
if os.path.exists(img_path) :
    print('해당 디렉터리가 존재함')

    # (3) image 파일 저장, 파일 이동 디렉터리 생성
    images = [] # png 파일 저장
    os.mkdir(img_path2) # 디렉터리 생성

    # (4) images 디렉터리에서 png 검색
    for pic_path in glob(img_path + '*.png'):
# png 검색
```

Python Console

```
D:\Pywork\workspace
해당 디렉터리가 존재함
```

4. 이미지 파일 이동

```
        # (5) 경로와 파일명 분리, 파일명 추가
        img_path = os.path.split(pic_path)
        images.append(img_path[1]) # png 파일명
추가

        # (6) 이진파일 읽기
        rfile = open(file=pic_path, mode='rb')
        output = rfile.read()

        # (7) 이진파일 쓰기 -> chapter08/png 폴더
        이동
        wfile = open(img_path2+img_path[1], mo
de='wb')
        wfile.write(output)
    rfile.close(); wfile.close() # 파일 객체 닫기
else:
    print('해당 디렉터리가 없음')

print('png file =', images)
```

```
png file = ['101.png', '
102.png', '103.png', '1
04.png', '105.png']
```

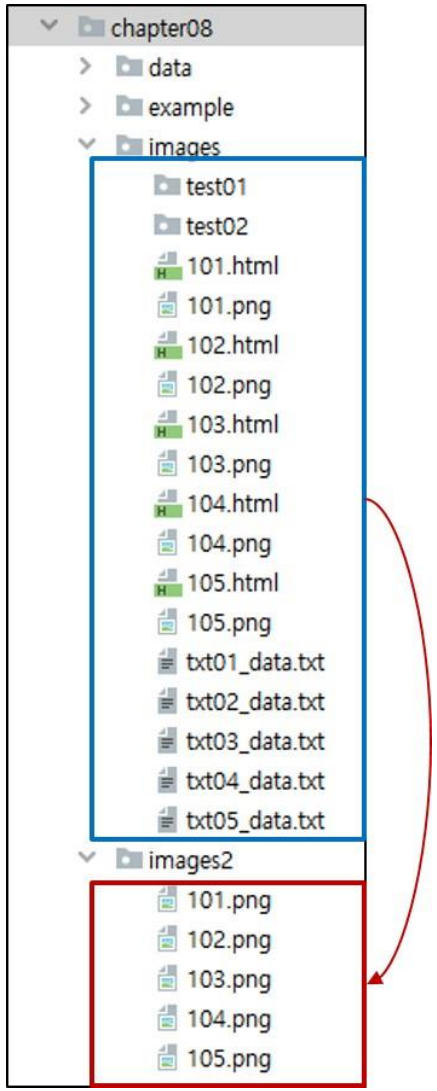
4. 이미지 파일 이동

- 해설 이미지 파일 이동 예
- # (1) image 파일 경로
- 이미지 원본 파일의 경로와 이미지 파일 이동 경로를 지정한다.
- # (2) 디렉터리 존재 유무
- os.path.exists()함수를 이용하여 이미지 원본 파일의 경로 가 있는지를 비교 판단한다. 해당 디렉터리가 없으면 False를 반환한다.
- # (3) image 파일 저장, 파일 이동 디렉터리 생성
- 이미지 파일명만 별도로 저장할 수 있는 빈 리스트를 만들고, 이미지 파일 별도로 저장할 수 있는 디렉터리를 생성한다.
- # (4) images 디렉터리에서 png 검색
- glob()함수를 이용하여 *.png 파일만 검색한다.

4. 이미지 파일 이동

- # (5) 경로와 파일명 분리, 파일명 추가
- os.path.split() ()함수를 이용하여 디렉터리와 파일명을 분 리한다.
- # (6) 이진파일 읽기
- open() 함수에서 mode='rb' 파라미터를 지정하여 이진파일 읽기 객체를 생성한다.
- # (7) 이진파일 쓰기
- open() 함수에서 mode='wb' 파라미터를 지정하여 이진파일 쓰기 객체를 생성하여 png 디렉터리로 이미지 파일을 이동한다. 이미지 파일 이동 결과이다.

4. 이미지 파일 이동


chapter08
data
example
images
test01
test02
101.html
101.png
102.html
102.png
103.html
103.png
104.html
104.png
105.html
105.png
txt01_data.txt
txt02_data.txt
txt03_data.txt
txt04_data.txt
txt05_data.txt
images2
101.png
102.png
103.png
104.png
105.png

THANK YOU