☞ 텍스트 파일 읽기

파이썬에서 데이터를 처리 및 조작하고 분석하려면 파일에 접근하여 읽는 방법을 알아야 한다.

1초에 아주 많은 파일을 처리할 수 있는 프로그램을 만들었다면, 파일 하나하나를 일회성으로 수작업으로 처리하는 것보다 그 프로그램이 훨씬 효율적이라는 사실을 확인할 수 있을 것이다.

먼저 스크립트가 어떤 파일을 처리할 지 파이썬에 알려주어야 한다.

처리할 파일명을 스크립트네에 하드코딩할 수도 있지만, 그렇게 하면 다양한 파일을 프로그램에 적용하기 어려워진다.

융통성 있는 파일 읽기 방법은 명령 프롬프트나 터미널 창의 명령줄에서 파이썬 스크립트 뒤에 읽을 파일을 적어서 실행하는 것이다.

이렇게 하려면 스크립트 상단에 파이썬 기본 모듈인 sys를 import 해야 한다.

```
#!/usr/bin/env python3

import sys
```

sys 모듈을 임포트하면 리스트 형식의 argv 변수를 사용할 수 있다.

이 변수는 명령 줄 인수들로 구성된 리스트를 파이썬 스크립트로 가져온다.

이 리스트에는 스크립트 이름을 포함해 명령 줄에 입력된 모든 것이 포함된다.

다른 리스트와 마찬가지로 argv는 인덱스를 갖는다.

argv[0]는 스크립트 이름이고,

argv[1]은 명령줄을 통해 전달된 첫번째 인수이다.

실습1. 텍스트 파일 생성하여 읽기

먼저 텍스트 파일을 읽으려면 읽을 대상인 파일이 있어야 한다.

사전작업

네이버나 구글등에서 팝송가사 3곡을 검색하여 각각의 Text 파일에 저장한다.

1. Baby_Shark.txt

2. Three_bears.txt

3. twinkle_little_star.txt

스크립트 파일 생성

텍스트 파일을 읽을 수 있는 스크립트 파일을 생성하여 txt 파일과 같은 경로에 저장해준다.

경로가 다를 경우 Text 파일을 지정할 경우 절대경로로 표시해준다.

```
#!/usr/bin/env python3
from math import exp, log, sqrt
import re
from datetime import date, time, datetime, timedelta
from operator import itemgetter
import sys

print("Output: Baby_Shark.txt")
input_file = sys.argv[1]
filereader = open(input_file, 'r')
for row in filereader :
    print(row.strip())
filereader.close()
```

구문 설명

input_file = sys.argv[1]

읽을 파일의 경로와 파일명을 지정하여 input_file 변수에 할당한다.

filereader = open(input_file, 'r')

파일 객체인 filereader를 생성하고 open 함수로 지정된 파일을 'r' 읽기모드로 열고 파일 내용을 각 행을 저장한다.

for row in filereader :

filereader 객체에 잇는 행을 한번에 하나씩 읽는다.

    print(row.strip())

    strip 함수를 이용하여 각 행의 끝에 있는 공백, 탭, 개행문자 등을 제거한다.

filereader.close()

입력 파일의 모든 행을 읽어으면 filereader 객체를 닫는다.

작성이 완료되었으므로 **first_script.py** 파일명으로 저장한다.

스크립트 파일 실행을 위하여 명령프롬프트 <cmd> 창을 연다.

cmd> python 스크립트 파일명 input_file이름(읽어들이 txt 파일을 지정한다)

예) cmd> python first_script.py Baby_Shark.txt

텍스트 파일의 내용이 출력되는 것을 확인할 수 있다. (나머지 두개의 파일도 반복 실습한다.)

만약, 스크립트와 입력 파일이 다른 경로에 있을 경우에는 다음과 같이 파일의 경로도 지정해준다.

cmd> python first_script.py "D:₩Baby_Shark.txt"

실습2. 텍스트 파일 생성하여 읽기 : 두번째 방법

실습1에서 filereader 객체를 생성하는데 사용한 방식은 오래된 방식으로, close( ) 함수로 명시적으로 객체를 닫거나 혹은 스크립트가 종료될 때까지 파일 객체가 열려 있기 때문에 주의해야 한다. 이러한 현상은 보통은 문제가 되지 않지만, 더 복잡한 스크립트에서는 오류를 일으킬 수 있으므로 , 현재 파이썬 버전에서는 with 문을 이용하여 파일을 생성하는 방법을 사용하며, with 문이 끝날 때 자동으로 파일 객체를 닫아준다.

스크립트 파일 생성

텍스트 파일을 읽을 수 있는 스크립트 파일을 생성하여 txt 파일과 같은 경로에 저장해준다.

경로가 다를 경우 Text 파일을 지정할 경우 절대경로로 표시해준다.

```
#!/usr/bin/env python3
from math import exp, log, sqrt
import re
from datetime import date, time, datetime, timedelta
from operator import itemgetter
import sys

print("Output: Baby_Shark.txt")
input_file = sys.argv[1]
with open(input_file, 'r', newline=' ') as filereader :
        for row in filereader :
                print("{ }".format(row.strip( ))
```

구문 설명

with문 방식은 기존 방식과 매우 유사하지만, filereader 객체를 닫기 위한 close( ) 함수를 호출하는 부분이 필요 없어졌다.

input_file = sys.argv[1]

읽을 파일의 경로와 파일명을 지정하여 input_file 변수에 할당한다.

with open(input_file, 'r', newline=' ') as filereader :

파일 객체인 filereader를 생성하고 open 함수로 지정된 파일을 'r' 읽기모드로 열고 파일 내용을 각 행을 저장한다.

for row in filereader :

filereader 객체에 잇는 행을 한번에 하나씩 읽는다.

print("{ }".format(row.strip( ))

strip 함수를 이용하여 각 행의 끝에 있는 공백, 탭, 개행문자 등을 제거한다.

작성이 완료되었으므로 **second_script.py** 파일명으로 저장한다.

스크립트 파일 실행을 위하여 명령프롬프트 <cmd> 창을 연다.

cmd> python 스크립트 파일명 input_file이름(읽어들이 txt 파일을 지정한다)

예) cmd> python second_script.py Baby_Shark.txt

텍스트 파일의 내용이 출력되는 것을 확인할 수 있다. (나머지 두개의 파일도 반복 실습한다.)

만약, 스크립트와 입력 파일이 다른 경로에 있을 경우에는 다음과 같이 파일의 경로도 지정해준다.

cmd> python second_script.py "D:₩Baby_Shark.txt"

실습3. glob 모듈을 이용하여 텍스트 파일 읽기

명령줄에서 파이썬 스크립트 파일명 뒤에 입력 파일들이 포함된 디렉터리를 지정하여 텍스트 파일을 자동으로 읽어들이게 설정할 수 있다. 이렇게 하려면 파이썬 기본 모듈인 os와 glob 모듈을 import 해준다.

os 모듈을 임포트하면 유용한 경로 함수들을 이용할 수 있다. 예를들면 os.path.join 함수는 지능적으로 하나 이상의 경로를 구성하는 요소들을 서로 합친다. glob 모듈은 특정 패턴과 일치하는 모든 경로명을 찾는다. os와 glob 모듈을 결합하면 특정 디렉토리내에 존재하는 특정한 패턴을 지닌 모든 파일을 찾을 수 있다.

스크립트 파일 생성

텍스트 파일을 읽을 수 있는 스크립트 파일을 생성하여 txt 파일과 같은 경로에 저장해준다.

경로가 다를 경우 Text 파일을 지정할 경우 절대경로로 표시해준다.

```
#!/usr/bin/env python3
from math import exp, log, sqrt
import re
from datetime import date, time, datetime, timedelta
from operator import itemgetter
import sys
import glob
import os

print("Output: Three_bears.txt.txt")
input_Path = sys.argv[1]
for input_file in glob.glob(os.path.join(input_Path, '*.txt')) :
        with open(input_file, 'r', newline=' ') as filereader :
                for row in filereader :
                        print("{ }".format(row.strip( ))
```

구문 설명

input_Path = sys.argv[1]

읽을 파일의 경로 대신 디렉터리 경로를 지정하여 input_Path 변수에 할당한다.

for input_file in glob.glob(os.path.join(input_Path, '*.txt')) :

os.path.join 함수와 glob.glob 함수를 이용하여 특정 폴더에서 패턴과 일치하는 모든 파일을 찾는다.

os.path.join 함수는 glob.glob 함수에 의해 확장된 패턴과 일치하는 폴더 내의 모든 파일명과 폴더의 이름을 합친다. 이번 실습에서는 '*.txt' 패턴을 사용하여 폴더내의 모든 텍스트 파일을 찾는다.

with open(input_file, 'r', newline=' ') as filereader :

파일 객체인 filereader를 생성하고 open 함수로 지정된 파일을 'r' 읽기모드로 열고 파일 내용을 각 행을 저장한다.

> **for row in filereader :**
>
> filereader 객체에 있는 행을 한번에 하나씩 읽는다.
>
> > **print("{ }".format(row.strip( ))**
> >
> > strip 함수를 이용하여 각 행의 끝에 있는 공백, 탭, 개행문자 등을 제거한다.

작성이 완료되었으므로 **third_script.py** 파일명으로 저장한다.

스크립트 파일 실행을 위하여 명령프롬프트 <cmd> 창을 연다.

cmd> python 스크립트 파일명 디렉토리경로명(읽어들이 txt 파일이 있는 폴더 경로를 지정한다)

예) cmd> python third_script.py "D:/"

지정된 디렉터리내의 모든 텍스트 파일의 내용이 한꺼번에 출력되는 것을 확인할 수 있다.

실습4. 텍스트 파일 생성하여 쓰기

파이썬은 텍스트 및 파일을 작성하는 쉬운 두가지 방법을 제공해준다.

write 함수는 개별 문자열을 파일에 작성하고, writelines 함수는 일련의 문자열을 파일에 작성한다.

이번에 실습할 내용은 range( ) 함수와 len( ) 함수를 조합하여 리스트 내 값들의 인덱스를 추적하여 각 값 사이에 구분자를 넣고 개행문자를 마지막에 넣어 파일을 생성하는 실습을 진행한다.

스크립트 파일 생성

```
#!/usr/bin/env python3
from math import exp, log, sqrt
import re
from datetime import date, time, datetime, timedelta
from operator import itemgetter
import sys

my_letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
max_index = len(my_letters)

output_file = sys.argv[1]
filewriter = open(output_file, 'w')
for index_value in range(len(my_letters)) :
        if index_value < (max_index-1) :
                filewriter.write(my_letters[index_value]+'₩t')
        else :
                filewriter.write(my_letters[index_value]+'₩n')
filewriter.close( )
```

구문 설명

my_letters

이 변수는 문자열을 담고 있는 리스트이다. 각 문자들을 탭으로 구분하여 텍스트 파일에 출력하고 있다.

마지막 문자열의 차례가 되었다는 것을 알아내려면 리스트 내 원소의 인덱스 값을 추적해야 한다.

len( ) 함수로 리스트의 원소 개수를 구할 수 있으며, max_index의 값은 7이 된다.

output_file = sys.argv[1]

생성할 파일의 경로와 파일명을 지정하여 output_file 변수에 할당한다.

filewriter = open(output_file, 'w')

생성할 파일에 내용을 작성해야 하기 때문에 'w' 모드로 파일을 열어주었다.

for index_value in range(len(my_letters)) :

my_letters 리스트의 값을 반복할 range( ) 함수를 len( ) 함수와 조합하여 리스트 내 각 원소의 인덱스를 추적한다.

if index_value < (max_index-1) :

if ~ elsle 문으로 리스트의 마지막 원소와 나머지 모든 원소 사이를 구분한다.

if 블록은 인덱스값이 9보다 작은지 확인한다.

이 조건은 리스트의 마지막 원소를 만날때까지 True 이다.

filewriter.write(my_letters[index_value]+'₩t')

if 블럭은 리스트의 마지막 원소가 나오기 전까지는 원소 뒤에 Tab(₩t)을 붙여

출력 파일에 작성한다.

else :

filewriter.write(my_letters[index_value]+'₩n')

for 문이 리스트의 마지막 문자에 도달하면 인덱스는 9가 되므로 9보다 작지 않게 되어 False로 판단되어 else : 블록이 실행된다.

else 블록의 write 문은 원소 뒤에 개행문자를 추가하여 출력 파일에 작성한다.

filewriter.close( )

파일의 모든 행을 읽고 작성이 완료되면 filerwriter 객체를 닫는다.

작성이 완료되었으므로 **fourth_script.py** 파일명으로 저장한다.

스크립트 파일 실행을 위하여 명령프롬프트 <cmd> 창을 연다.

cmd> python 스크립트 파일명 output_file이름(새로 작성할 txt 파일을 지정한다)

예) cmd> python fourth_script.py "D:₩write_to_file.txt"

fourth_script.py 파일을 열어보면 리스트 my_letters의 리스트 문자열들이 탭으로 구분되어 작성된 것을 확인할 수 있다.

실습5. CSV 파일 생성하여 쓰기

이번에 실습할 내용은 실습4에서 생성한 **write_to_file.txt** 파일의 마지막에 내용을 추가하기 위해 'a' append 모드로 작성하는 실습을 진행한다.

스크립트 파일 생성

```
#!/usr/bin/env python3
from math import exp, log, sqrt
import re
from datetime import date, time, datetime, timedelta
from operator import itemgetter
import sys

my_numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
max_index = len(my_numbers)

output_file = sys.argv[1]

filewriter = open(output_file, 'a')
for index_value in range(len(my_numbers)) :
        if index_value < (max_index-1) :
                filewriter.write(str(my_numbers[index_value])+', ')
        else :
                filewriter.write(str(my_numbers[index_value])+'₩n')
filewriter.close( )
```

구문 설명

my_numbers

이 변수에는 정수형 리스트가 들어 있다.

write 함수는 문자열만을 작성하기 때문에 문자열이 아닌 값을 먼저 문자열로 변환해야 한다.

그래서 아래 구문에서 str 함수를 사용했다.

output_file = sys.argv[1]

데이터를 추가할 파일의 경로와 파일명을 지정하여 output_file 변수에 할당한다.

filewriter = open(output_file, 'a')

생성할 파일에 내용을 추가 작성해야 하기 때문에 'a' 모드로 파일을 열어주었다.

for index_value in range(len(my_numbers)) :

my_numbers 리스트의 값을 반복할 range( ) 함수를 len( ) 함수와 조합하여 리스트 내 각 원소의 인덱스를 추적한다.

if index_value < (max_index-1) :

if ~ elsle 문으로 리스트의 마지막 원소와 나머지 모든 원소 사이를 구분한다.

if 블록은 인덱스값이 9보다 작은지 확인한다.

이 조건은 리스트의 마지막 원소를 만날때까지 True 이다.

filewriter.write(str(my_numbers[index_value])+', ')

if 블럭은 리스트의 마지막 원소가 나오기 전까지는 원소 뒤에 Tab(₩t)을 붙여

출력 파일에 작성한다.

else :

filewriter.write(str(my_numbers[index_value])+'₩n')

for 문이 리스트의 마지막 문자에 도달하면 인덱스는 9가 되므로 9보다 작지 않게 되어 False로 판단되어 else : 블록이 실행된다.

else 블록의 write 문은 원소 뒤에 개행문자를 추가하여 출력 파일에 작성한다.

filewriter.close( )

파일의 모든 행을 읽고 작성이 완료되면 filerwriter 객체를 닫는다.

작성이 완료되었으므로 **fifth_script.py** 파일명으로 저장한다.

스크립트 파일 실행을 위하여 명령프롬프트 <cmd> 창을 연다.

cmd> python 스크립트 파일명 output_file이름(추가로 작성할 txt 파일을 지정한다)

예) cmd> python fifth_script.py "D:₩write_to_file.txt"

fifth_script.py 파일을 열어보면 리스트 my_numbers의 리스트 값들이 쉼표로 구분되어 추가된 것을 확인할 수 있다.