파일

파일

❖ 파일의 종류

- 텍스트 파일
  - 사람이 읽을 수 있는 글자로 저장
- 바이너리 파일
  - 컴퓨터가 읽고 이해할 수 있는 이진 데이터를 저장

- 파일 처리 절차
  - 파일 열기
    - 성공시 파일 객체 리턴

    f = open(파일이름, 모드)
  - 파일 처리
  - 파일 닫기

    f.close()

❖ **open(파일명, 모드)**

- 파일명 : 절대경로, 상대경로 지정
- 모드

모드	설명
r 또는 rt	텍스트 모드로 읽기
w 또는 wt	텍스트 모드로 쓰기
a 또는 at	텍스트 모드로 파일 마지막에 추가하기
rb	바이너리 모드로 읽기
wb	바이너리 모드로 쓰기
ab	바이너리 모드로 파일 마지막에 추가하기

❖ 기본 골격

```
f1 = open('text.txt', 'r')
f2 = open('c:/images/picture.jpg', 'rb')

# ----------------------------------------
# 오픈한 파일을 처리하는 코드
# ----------------------------------------

f1.close()
f2.close()
```

❖ 텍스트 파일을 읽고 출력하기

- 파일 객체의 read() 메소드
  - 파일의 전체 내용을 읽어 리턴

```
f = open('c:/temp/test.txt', 'r')
data = f.read()
print(data)
f.close()
```

❖ 텍스트 파일을 한 줄씩 읽고 출력하기

- readline() 메소드
  - 파일의 현재 위치에서 한 줄을 읽어서 리턴
  - 더 이상 읽을 내용이 없으면(파일의 끝) 빈 문자열 리턴

```
f = open('c:/temp/test.txt', 'r')

line_num = 1
line = f.readline()

while line:
    print('%d %s'%(line_num, line), end='')
    line = f.readline()
    line_num += 1

f.close()
```

❖ 텍스트 파일을 한 줄씩 읽고 출력하기

- readlines() 메소드
  - 읽어 들인 각 줄을 요소로 하는 리스트로 리턴

```
f = open('c:/temp/test.txt', 'r')

line_num = 1
lines = f.readlines()

for line_num, line in enumerate(lines):
    print('%d %s'%(line_num+1, line), end='')

f.close()
```

❖ 사용자 입력 받기

- input() 함수
  - 사용자가 키보드로 입력한 값을 문자열로 리턴
  - 인자는 입력 프롬프트 문자열

```
k = input('<값>을 입력하세요:')
print('당신이 입력한 값은 <' + k + '>입니다.')
```

```
<값>을 입력하세요:헬로우
당신이 입력한 값은 <헬로우>입니다.
```

❖ 화면에서 사용자 입력을 받고 파일로 쓰기(write)

```
text = input('파일에 저장할 내용을 입력하세요')
f = open('c:/temp/data.txt', 'w')

f.write(text)

f.close()
```

❖ 리스트 내용을 파일에 쓰기 (writelines)

```
count = 1
data = []

print('파일에 내용을 저장하려면 내용을 입력하지 말고 [Enter]를 누르세요')

while True:
    text = input('[%d] 파일에 저장할 내용을 입력하세요: '%count)
    if text == '' :
        break;
    data.append(text + '\n')
    count += 1

f = open('c:/temp/output.txt', 'w')
f.writelines(data)
f.close()
```

❖ 파일 복사

```
fin = open('c:/temp/test.txt', 'r')
fout = open('c:/temp/test_copy.txt', 'w')


data = fin.read()
fout.write(data)

fin.close()
fout.close()
```

❖ 바이너리 파일 복사하기(read, write)

```
bufsize = 104;
fin = open('c:/temp/test.jpg', 'rb')
fout = open('c:/temp/test_copy.jpg', 'wb')

data = fin.read(bufsize)
while data:
    fout.write(data)
    data = fin.read(bufsize)

fin.close()
fout.close()
```

❖ 파일을 열고 자동으로 닫기(with ~ as)

- close() 호출 생략(자동 호출됨)

```
with open('c:/temp/test.txt') as fin:
    for line_number, line in enumerate(fin.readlines()):
        print('%d %s'%(line_number, line), end='')
```

❖ 파일으 특정 부분만 복사하기(seek, read, write)

- seek() 함수
  - 파일 위치 포인터 조정

```
position = 105
size = 500

fin = open('c:/temp/test.txt', 'r')
fout = open('c:/temp/test.txt', 'w')

f.seek(position)
data = fin.read(size)
fout.write(data)

fin.close()
fout.close()
```

❖ 파일의 크기 구하기(os.path.getsize())

- os.path 모듈의 getsize() 함수

```
from os.path import getsize

file1 = 'c:/temp/test.txt'
file2 = 'c:/temp/test.jpg'

file1_size = getsize(file1)
file2_size = getsize(file2)

print('File Name : %s \tFile Size:%d'%(file1, file1_size))
print('File Name : %s \tFile Size:%d'%(file2, file2_size))
```

❖ 파일 삭제 (os.remove())

```
from os import remove

target_file = 'c:/temp/test_copy.txt'
k = input('[%s] 파일을 삭제하겠습니까?(y/n)'%target_file)

if k == 'y' :
  remove(target_file)
  print('[%s]를 삭제했습니다.'%target_file)
```

❖ 파일 이름 변경(os.rename)

```
from os import rename

old_name = 'c:/temp/test.txt'
new_name = input('[%s]에 대한 새로운 파일 이름을 입력하세요'%old_name);
new_name = 'c:/temp/' + new_name

rename(old_name, new_name)

print('[%s]-->[%s]로 파일 이름이 변경되었습니다'%(old_name, new_name))
```

❖ 디렉토리에 있는 파일 목록 얻기(os.listdir, glob.glob)

- os.listdir(경로)
  - 지정한 경로에 있는 파일 목록을 리스트로 리턴
  - 파일명만 리턴
- glob.glob(필터)
  - 지정한 조건을 만족하는 파일 목록만 리스트로 리턴
  - 필터에 경로가 있다면 경로를 포함해서 파일 목록 리턴

```
import os, glob

folder = 'c:/temp'

file_list = os.listdir(folder)
print(file_list)

files = 'c:/temp/*.txt'
file_list = glob.glob(files)
print(file_list)
```

❖ 현재 디렉토리 확인하고 바꾸기(os.getcwd, os.chdir)

- os.getcwd()
  - 현재 작업 디렉토리 경로 리턴
- os.chdir(경로)
  - 현재 작업 디렉토리 변경

```
import os

pdir = os.getcwd()
print(pdir)

os.chdir('..')
print(os.getcwd())

os.chdir(pdir)
print(os.getcwd())
```

❖ 디렉토리 생성하기(os.mkdir)

```
import os

newfolder = input('새로 생성할 디렉터리 이름을 입력하세요: ')
try:
    os.mkdir(newfolder)
    print('[%s] 디렉터리를 새로 생성했습니다.' %newfolder)
except Exception as e:
    print(e)
```

❖ 디렉토리 제거하기(os.rmdir)

- os.rmdir(경로)
  - 비어있지 않은 디렉토리인 경우 예외 발생

```
import os

target_folder = 'tmp'
k = input('[%s] 디렉터리를 삭제하겠습니까? (y/n)'%target_folder)
if k == 'y':
    try:
        os.rmdir(target_folder)
        print('[%s] 디렉터리를 삭제했습니다.'%target_folder)
    except Exception as e:
        print(e)
```

❖ 하위 디렉토리 및 파일 전체 삭제하기(shutil.rmtree)

- shutil.rmtree(경로)
  - 지정한 경로에 있는 모든 파일과 하위 디렉토리를 삭제

```
import shutil
import os

target_folder = 'c:/temp/test'
print('[%s] 하위 모든 디렉터리 및 파일들을 삭제합니다.' %target_folder)
for file in os.listdir(target_folder):
    print(file)

k = input('[%s]를 삭제하겠습니까? (y/n) ' %target_folder)
if k == 'y':
    try:
        shutil.rmtree(target_folder)
        print('[%s]의 모든 하위 디렉터리와 파일들을 삭제했습니다.'%target_folder)
    except Exception as e:
        print(e)
```

❖ 파일이 존재하는지 체크(os.path.exists)

```
import os
from os.path import exists

dir_name = input('새로 생성할 디렉터리 이름을 입력하세요: ')
if not exists(dir_name):
    os.mkdir(dir_name)
    print('[%s] 디렉터리를 생성했습니다.' %dir_name)
else:
    print('[%s]은(는) 이미 존재합니다.' %dir_name)
```

❖ 파일인지 디렉토리인지 확인하기

- os.path.isfile, os.path.isdir

```
import os
from os.path import exists, isdir, isfile

files = os.listdir()
for file in files:
   if isdir(file):
      print('DIR: %s' %file)

for file in files:
   if isfile(file):
      print('FILE: %s' %file)
```