**2.6.2 바이너리 파일**

이진 파일로 파일을 열면 텍스트 파일처럼 인코딩 작업이나 줄바꿈 문자에 대한 변환이 없이 항상 1바이트 크기의 배열인 bytes 객체로 읽고 쓰기를 수행한다. 다음은 이진파일로 열어 파일을 복사한 예이다.

**파일 복사**

f = open('ABBA.mp3','rb')

data = f.read() # bytes

f.close()

f = open('ABBA-copy.mp3','wb')

f.write(data)

f.close()

**MP3 파일 곡명 확인**

다음은 [Working with File Objects](http://www.diveintopython.net/file_handling/file_objects.html)에 소개된 코드를 발췌한 것으로 mp3 파일에서 곡명을 확인한 예이다. mp3 파일은 파일 마지막의 128 바이트에 곡명, 장르 등등 여러 정보를 저장한다(포맷 정보는 [한글 위키피디아](https://ko.wikipedia.org/wiki/ID3) 참조).

>>> f = open('ABBA.mp3','rb')

>>> f.seek(-128,2) # 끝에서 128 바이트로 위치 이동

>>> tagdata = f.read(128)

>>> title = tagdata[3:33].decode()

>>> title

'I Do I Do I Do I Do I Do '

>>> f.close()

* 위에서 bytes.decode() 를 이용해 유니코드 문자열인 str로 변환하고 있다.

**문자열 읽고 쓰기**

문자열을 읽고 쓰는 것은 텍스트 파일로 파일을 열고 읽고 쓰면 편리하다. 만약 이진 파일로 파일을 열어 다룬 때는 str.encode(encoding='utf-8')과 bytes.decode(encoding='utf-8')을 적절히 활용해야 한다. 다음은 문자열을 이진파일에 읽고 쓴 예이다.

# writing str to binary file

mytext = '이 일은 쉬운 일이 아닙니다.'

f = open('mydata.bin','wb')

f.write(mytext.encode())

f.close()

# reading str from binary file

f = open('mydata.bin','rb')

bdata = f.read()

mytext = bdata.decode()

f.close()

**수치 데이터 읽고 쓰기**

int, float 같은 수치데이터는 struct 모듈에서 제공하는 struct.pack(fmt,v1,v2,...)으로 bytes로 변환할 수 있으며(이를 packing이라고 함), 반대로 (v1,v2,...) = struct.unpack(fmt)로 수치데이터로 변환가능하다(이를 unpacking이라고 함). 이때 v1, v2 등이 수치데이터이고, fmt는 포맷문자열이다. 예를 들어 fmt가 "idd"이면 int, float, float 순서로 packing이나 unpacking을 한다는 의미이다.

**import** struct

# packing numerical data into bytes

data = struct.pack("idd",1,10.3,-11.3) # int, float, float

# unpacking bytes to numerical data

(i,x,y) = struct.unpack("idd",data) # i=1, x = 10.3, y=-11.3

파일에 읽고 쓰는 예는 다음과 같다.

**import** struct

# writing data

age = 27 # int

height = 175.2 # float

weight = 71.3 # float

data = struct.pack('idd',age,height,weight)

f = open('mydata.bin','wb')

f.write(data)

f.close()

# reading data

f = open('mydata.bin','rb')

data = f.read()

(age,height,weight) = struct.unpack('idd',data)

**문자열과 수치데이터를 동시에 읽고 쓰기**

문자열은 str.encode(), bytes.decode() 함수를 활용하고, 수치데이터는 data=struct.pack(fmt,v1,v2,...)와 (v1,v2,...)=struct.unpack(fmt,data)를 적절히 활용한다. 이때 문자열의 길이를 직접 계산해야 한다는 점에 주의한다.

**import** struct

# writing data

name = '홍길동' # str

age = 27 # int

height = 175.2 # float

weight = 71.3 # float

name\_bytes = name.encode()

name\_data = struct.pack('i',len(name\_bytes)) + name\_bytes

numeric\_data = struct.pack('idd',age,height,weight)

data = name\_data+numeric\_data

f = open('mydata.bin','wb')

f.write(data)

f.close()

# reading data

f = open('mydata.bin','rb')

data = f.read()

strlen, = struct.unpack('i',data[0:4])

name = data[4:(4+strlen)].decode()

(age,height,weight) = struct.unpack('idd',data[4+strlen:])

**참고**

이진 파일에 대한 입출력은 상당히 복잡하다. 보다 자세한 사항은 [Working binary data python](https://www.devdungeon.com/content/working-binary-data-python)을 참고한다.