

# CHAPTER 6. 파일 입출력

---

## section99. 파일 입출력의 개요

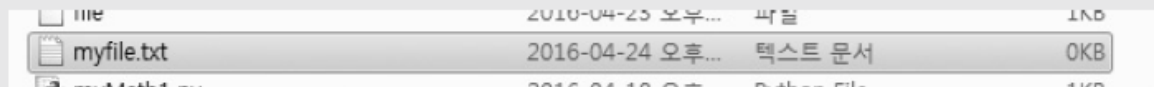
### step 1. 알고가기

- 핵심내용
- open() 함수를 w 모드로 사용하면 파일을 새로 만들 수 있다.
- 코딩하기 전에
  - 파일 입출력은 <파일열기>-<파일처리>-<파일닫기>의 3단계를 거친다.
  - open()함수는 파일열기 단계에서 사용한다.
  - 파일에는 텍스트 파일과 바이너리 파일이 있다.
  - 컴퓨터 안에 존재하는 모든 파일은 바이너리 파일이라고 하며, 이 중 사람들이 사용하는 문자열이 들어간 파일을 텍스트파일이라고 한다.

# section99. 파일 입출력의 개요

## step 2. 코딩

```
1: # section_099.py
2:
3: f = open('myfile.txt', 'w')
4:
5: f.close()
```



me	2010-04-25 오후...	폴더	1KB
myfile.txt	2016-04-24 오후...	텍스트 문서	0KB
me\myfile1.txt	2016-04-24 오후...	텍스트 문서	1KB

# section99. 파일 입출력의 개요

## step 3. 코드분석

### • 파일모드

파일 열기 모드	설명
w 쓰기 모드	<ul style="list-style-type: none"><li>파일에 데이터를 저장하고 싶을 때 사용함.</li><li>파일이 폴더에 존재하지 않으면 파일을 생성함.</li><li>만약 동일한 파일이 존재하면 기존 데이터를 삭제하고 파일을 새로 만듦. 따라서 사용 시 주의할 것</li></ul>
r 읽기 모드	<ul style="list-style-type: none"><li>파일에서 데이터를 가져올 때 사용함.</li><li>파일이 폴더에 존재하면 파일 객체를 반환해주고,</li><li>존재하지 않으면 OSError를 발생함.</li><li>파일 열기 모드의 기본 값이기 때문에 생략하면 읽기 모드로 파일이 열림.</li></ul>
a 추가 모드	<ul style="list-style-type: none"><li>파일에 데이터를 저장하고 싶을 때 사용함.</li><li>파일이 폴더에 존재하지 않으면 파일을 생성함.</li><li>만약 동일한 파일이 존재하면 파일 안의 기존 데이터에 이어서 데이터를 추가할 수 있음.</li><li>‘w’와 차이점은, ‘a’는 파일에 존재하는 데이터 이후에 추가로 쓸 수 있고, ‘w’는 기존의 데이터를 지우고 완전히 새로운 파일을 만든다는 점에서 다름.</li></ul>

## section99. 파일 입출력의 개요

### step 3. 코드분석

- `f=open('myfile.txt', 'w')`
  - 파일객체변수 = `open('파일명', '파일_열기_모드')` 형식이다.
  - `open()` 함수는 파일을 열 때 사용하는 함수이다.
  - 파일을 생성할 때는 `open('파일명', 'w')`라고 코딩한다. 파일 열기 모드 'w'의 특성 때문에 `open()` 함수로 파일을 새로 만들 수 있다.
  - 실행되고 나면 실행 파일이 있는 폴더에 `myfile.txt` 파일이 만들어진 것을 확인할 수 있다.
  - 그리고 파일에 대한 모든 권한이 파일객체변수 `f`에 저장된다.

# section99. 파일 입출력의 개요

## step 4.

### 3줄 요약

#### • 3줄 요약 •

- 파일 입출력은 파일 열기 → 파일 처리 → 파일 닫기의 3단계로 이루어진다.
- `open( )` 함수의 세 가지 파일 열기 모드 중 'w' 모드는 빈 파일을 만들 때 사용한다.
- 'w' 모드로 파일을 생성할 때 폴더 내에 동일한 파일이 존재할 경우 기존 데이터가 삭제된다.



# section100. 파일을 열고 데이터 쓰기 – write()

## step 1. 알고가기

- 핵심내용
- write() 함수는 파일에 데이터를 저장하는 기능을 가진 함수이다.
- 코딩하기 전에
  - 절대경로: 작업 폴더와 무관하게 절대적인 위치를 가리키는 경로
    - 예) C:\\Windows\\System32  
C:/temp
  - 상대경로: 소스 코드(\*.py)가 있는 작업 폴더를 기준으로 상대적인 위치를 가리키는 경로
    - 예) ..\\python35  
./src/image
  - (파이썬에서 상하위 폴더를 구분할 땐 ‘/’ 또는 ‘\\’를 사용함.)
  - 파일명만 적으면 상대경로를 사용한 것이다.

# section100. 파일을 열고 데이터 쓰기 – write()

## step 2. 코딩

```
1: # section_100.py
2:
3: f = open('c:/temp/myfile.txt', 'w')
4:
5: memo = '파일을 열었으니 데이터를 저장하자'
6: f.write(memo)
7:
8: f.close()
```

실행 전 체크할 것:  
C드라이브 아래에 temp 폴더가  
있는지 확인하고 없으면 폴더를  
생성한 후 실행해야 함

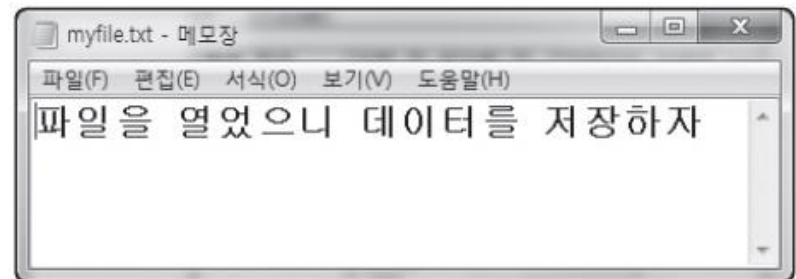
c:/temp/ 폴더에서  
myfile.txt를 열어 보.



# section100. 파일을 열고 데이터 쓰기 - write()

## step 3. 코드분석

- `f = open('c:/temp/myfile.txt', 'w')`
  - `myfile.txt`라는 파일을 'w' 모드로 열라는 명령이다.
  - 만약 파일이 없다면 `myfile.txt`를 생성하고, 파일이 존재한다면 파일 안에 있는 모든 데이터를 지우고 새롭게 파일을 만든다.
  - C 드라이브부터 시작하는 절대경로를 적어주면 지정한 폴더(`c:/temp`)에 파일을 생성한다.
- `f.write(memo)`
  - `write()` 함수를 이용해서 파일에 데이터를 저장한다. 괄호 안에 쓸 내용을 넣어준다.
- `f.close()`
  - 파일 종료하기



# section100. 파일을 열고 데이터 쓰기 - write()

## step 4.

### 3줄 요약

#### • 3줄 요약 •

- 파일 입출력의 3단계(파일 열기 → 파일 처리 → 파일 닫기)를 기억하자.
- 절대 경로와 상대 경로의 차이를 이해하자.
- write( ) 함수는 파일에 데이터를 저장하는 함수이다.



## section101. 파일을 열고 데이터 읽기 – readline()

### step 1. 알고가기

- 핵심내용
  - readline() 함수는 파일의 데이터를 한 줄씩 읽어오는 기능을 가지고 있다.
- 코딩하기 전에
  - open() 함수로 파일을 연 후 파일에 있는 데이터를 가져오는 방법은 여러 가지가 있다.
  - 이번에는 처음으로 readline() 함수를 사용해 본다.

# section101. 파일을 열고 데이터 읽기 – readline()

## step 2. 코딩

```
1: # section_101.py
2:
3: f = open('c:/temp/myfile.txt', 'w')
4: poem = ''
5: 살어리 살어리랏다 청산에 살어리랏다
6: 멀위랑 달래랑 먹고 청산에 살어리랏다
7: 알리 알리 알랑성 알라리 알라
8:
9: 우리라 우리라 새여 자고 니러 우리라 새여
10: 널라와 시름 한 나도 자고 니러 우리노라
11: 알리 알리 알랑성 알라리 알라
12: ''
13: f.write(poem)
14: f.close()
15:
16: f = open('c:/temp/myfile.txt', 'r')
17:
18: while True:
19:     sentence = f.readline()
20:     if not sentence:
21:         break
22:     print(sentence)
23:
24: f.close()
```

## section101. 파일을 열고 데이터 읽기 – readline()

### step 3. 코드분석

- `f = open('c:/temp/myfile.txt', 'r')`
  - 데이터를 읽기 위해서는 r 모드로 파일을 연다.
- `sentence = f.readline()`
  - `readline()` 함수는 파일에서 한 줄씩 읽어 온다. 이 것을 `sentence`라는 변수에 저장한다.
  - 파이썬은 개행문자인 `'\n'`가 나타날 때까지를 한 줄로 판단한다.
- `if not sentence:`
  - 파일의 끝(EOF)에 이르면 `readline()` 함수는 `'None'`을 돌려준다.
  - `sentence`의 값이 `None`이면 `False`인데 `not` 연산자와 함께 쓰여서 파일의 끝에 다다르면 `break`가 실행된다.

# section101. 파일을 열고 데이터 읽기 – readline()

## step 4.

### 3줄 요약

#### • 3줄 요약 •

- 파일의 데이터를 읽기 위해서는 반드시 'r' 모드로 파일을 열어야 한다.
- readline( ) 함수는 파일의 데이터를 한 줄씩 읽어오는 기능을 가진 함수이다.
- 파일의 끝(EOF)에 다다르면 readline( ) 함수는 None을 반환한다.



## section102. 파일을 열고 데이터 읽기 – readlines()

### step 1. 알고가기

- 핵심내용
  - readlines() 함수는 파일 전체를 줄 단위로 읽어서 리스트로 반환해준다.
- 코딩하기 전에
  - readlines() 함수는 파일의 모든 줄을 한꺼번에 읽어오는 기능을 가진다.

## section102. 파일을 열고 데이터 읽기 – readlines()

### step 2. 코딩

```
1: # section_102.py
2:
3: f = open('c:/temp/myfile.txt', 'w')
4: poem = '''
5: 살어리 살어리랏다 청산에 살어리랏다
6: 멀위랑 달래랑 먹고 청산에 살어리랏다
7: 알리 알리 알랑성 알라리 알라
8:
9: 우리라 우리라 새여 자고 니러 우리라 새여
10: 널라와 시름 한 나도 자고 니러 우리노라
11: 알리 알리 알라성 알라리 알라
12: '''
13: f.write(poem)
14: f.close()
15:
16: f = open('c:/temp/myfile.txt', 'r')
17:
18: all = f.readlines()
19:
20: print(all)
21:
22: for sentence in all:
23:     print(sentence)
24:
25: f.close()
```



## section102. 파일을 열고 데이터 읽기 – readlines()

### step 3. 코드분석

- `f = open('c:/temp/myfile.txt', 'r')`
  - 데이터를 읽기 위해서는 r 모드로 파일을 연다.
- `all = f.readlines()`
  - `readlines()` 함수를 이용하면 파일의 모든 내용을 줄 단위로 읽어서 각각의 줄을 항목으로 갖는 리스트 형태로 반환한다.
- `f.close()`
  - 파일 종료하기

## section102. 파일을 열고 데이터 읽기 – readlines()

### step 4. 3줄 요약

#### • 3줄 요약 •

- readlines( ) 함수는 파일의 모든 내용을 한꺼번에 가져온다.
- readlines( ) 함수는 파일 내용을 줄 단위로 끊어서 리스트 항목으로 만들어서 돌려준다.
- 파일의 용량이 매우 클 경우 처리 시간이 오래 걸릴 수 있다.



## section103. 파일을 열고 데이터 읽기 - read()

### step 1. 알고가기

- 핵심내용
  - read() 함수는 파일 내용 전체를 문자열로 반환해준다.
- 코딩하기 전에
  - read() 함수는 파일의 모든 내용을 문자열의 형태로 반환해 준다는 점에서 앞서 배운 readlines()와 다르다.

# section103. 파일을 열고 데이터 읽기 - read()

## step 2. 코딩

```
1: # section_103.py
2:
3: f = open('c:/temp/myfile.txt', 'w')
4: poem = ''
5: 살어리 살어리랏다 청산에 살어리랏다
6: 멀위랑 달래랑 먹고 청산에 살어리랏다
7: 알리 알리 알랑성 알라리 알라
8:
9: 우러라 우러라 새여 자고 니러 우러라 새여
10: 널라와 시름 한 나도 자고 니러 우리노라
11: 알리 알리 알랑성 알라리 알라
12: ''
13: f.write(poem)
14: f.close()
15:
16: f = open('c:/temp/myfile.txt', 'r')
17:
18: string = f.read()
19:
20: print(string)
21:
22: f.close()
```

## section103. 파일을 열고 데이터 읽기 – read()

### step 3. 코드분석

- `f = open('c:/temp/myfile.txt', 'r')`
  - 데이터를 읽기 위해서는 r 모드로 파일을 연다.
- `string = f.read()`
  - `read()` 함수를 이용해서 파일 내 모든 내용을 읽어 들이는데 이때 문자열형으로 반환해준다.
  - 따라서 `string` 변수는 문자열형이야.
- `f.close()`
  - 파일 종료하기

# section103. 파일을 열고 데이터 읽기 – read()

## step 4.

3줄 요약

### • 3줄 요약 •

- read( ) 함수는 파일의 모든 내용을 한꺼번에 가져온다.
- read( ) 함수는 파일 내용을 전체를 문자열로 반환한다.
- 파일의 용량이 매우 클 경우 처리 시간이 오래 걸릴 수 있다.



## section104. 파일을 열고 데이터 읽기(파일객체변수 사용)

### step 1. 알고가기

- 핵심내용
  - 파일 객체 변수는 파일 데이터를 줄 단위로 가지고 있다.
- 코딩하기 전에
  - 파일객체변수 `f`를 이용해 직접 출력할 수도 있다.

## section104. 파일을 열고 데이터 읽기(파일객체변수 사용)

### step 2. 코딩

```
1: # section_104.py
2:
3: f = open('c:/temp/myfile.txt', 'w')
4: poem = ''
5: 살어리 살어리랏다 청산에 살어리랏다
6: 멀위랑 달래랑 먹고 청산에 살어리랏다
7: 얄리 얄리 얄랑성 얄라리 얄라
8:
9: 우러라 우러라 새여 자고 니러 우러라 새여
10: 널라와 시름 한 나도 자고 니러 우리노라
11: 얄리 얄리 얄랑성 얄라리 얄라
12: ''
13: f.write(poem)
14: f.close()
15:
16: f = open('c:/temp/myfile.txt', 'r')
17:
18: for line in f:
19:     print(line)
20:
21: f.close()
```



## section104. 파일을 열고 데이터 읽기(파일객체변수 사용)

### step 3. 코드분석

- `f = open('c:/temp/myfile.txt', 'r')`
  - 데이터를 읽기 위해서는 r 모드로 파일을 연다.
- `for line in f:`
  - 파일객체변수 `f`는 줄 단위로 파일 내용을 가지고 있어서 각 항목을 `line`으로 받아서 바로 출력 가능하다.
  - 이 방법이 파일 읽는 방법 중 가장 간단하고 빠른 방법이다.
- `f.close()`
  - 파일 종료하기

# section104. 파일을 열고 데이터 읽기(파일객체변수 사용)

## step 4.

### 3줄 요약

#### • 3줄 요약 •

- 파일 객체 변수를 이용해서 직접 출력할 수 있다.
- 파일 객체 변수에는 파일 내용이 줄 단위로 저장되어 있다.
- 파일 객체 변수를 이용하면 코드가 간결해지고 속도도 빠른 편이다.



## section105. 파일을 열고 데이터 읽기(with문 사용)

### step 1. 알고가기

- 핵심내용
  - with문을 사용하면 안정적인 파일처리를 할 수 있다.
- 코딩하기 전에
  - with문을 이용하여 파일을 열 때의 장점은 close()함수를 호출하지 않아도 된다는 것이다.
  - with문이 종료되면서 자동으로 파일을 닫아주기 때문이다.
  - 또한 파일을 처리하다가 문제가 생겼을 경우에도 안정적으로 파일을 닫아주기 때문에 안정성이 높아진다.

## section105. 파일을 열고 데이터 읽기(with문 사용)

### step 2. 코딩

```
1:  # section_105.py
2:
3:  f = open('c:/temp/myfile.txt', 'w')
4:  poem = '''
5:  살어리 살어리랏다 청산에 살어리랏다
6:  멀위랑 달래랑 먹고 청산에 살어리랏다
7:  알리 알리 알랑성 알라리 알라
8:
9:  우러라 우러라 새여 자고 니러 우러라 새여
10:  널라와 시름 한 나도 자고 니러 우리노라
11:  알리 알리 알랑성 알라리 알라
12:  '''
13:  f.write(poem)
14:  f.close()
15:
16:  with open('c:/temp/myfile.txt', 'r') as f:
17:      for line in f:
18:          print(line)
```

## section105. 파일을 열고 데이터 읽기(with문 사용)

### step 3. 코드분석

- with open('c:/temp/myfile.txt', 'r') as f:
  - open() 함수를 with문과 함께 사용해 파일을 열고
  - 이 파일에 대한 권한을 파일객체 변수 f에 저장하는 코드이다.
- for line in f:
  - 파일객체변수 f는 줄 단위로 파일 내용을 가지고 있어서 각 항목을 line으로 받아서 바로 출력 가능하다.

# section105. 파일을 열고 데이터 읽기(with문 사용)

## step 4.

### 3줄 요약

#### • 3줄 요약 •

- with문을 이용하면 close( ) 함수를 사용하지 않아도 파이썬이 자동으로 파일을 닫아준다.
- with문을 이용한 파일 처리가 예외 처리(try-except)보다 더 안전하다.
- 파일 객체 변수를 이용해서 read( ), readline( ) 등 다른 함수들을 사용할 수 있다.



## section106. 파일에 데이터 추가하기

### step 1. 알고가기

- 핵심내용
  - ‘a’모드는 파일의 맨 끝에 데이터를 추가할 때 선택한다.

# section106. 파일에 데이터 추가하기

## step 2. 코딩

```
1: # section_106.py
2:
3: poem1 = '''
4:   살어리 살어리랏다 청산에 살어리랏다
5:   멀위랑 달래랑 먹고 청산에 살어리랏다
6:   알리 알리 알랑성 알라리 알라
7:   '''
8:
9: with open('c:/temp/myfile.txt', 'w') as f:
10:     f.write(poem1)
11:
12: with open('c:/temp/myfile.txt', 'r') as f:
13:     print('----- 원본 파일 -----')
14:     print(f.read())
15:
16: poem2 = '''
17:   우리라 우리라 새여 자고 니러 우리라 새여
18:   널라와 시름 한 나도 자고 니러 우리노라
19:   알리 알리 알랑성 알라리 알라
20:   '''
21:
22: with open('c:/temp/myfile.txt', 'a') as f:
23:     f.write(poem2)
24:
25: with open('c:/temp/myfile.txt', 'r') as f:
26:     print('----- 데이터를 추가한 후 -----')
27:     print(f.read())
```

기존 파일에 데이터를  
추가하기 위해 'a' 모드를  
선택했어.



## section106. 파일에 데이터 추가하기

### step 3. 코드분석

- with open('c:/temp/myfile.txt', 'a') as f:
  - myfile.txt를 추가 모드('a')로 열었다.
  - 'a' 모드는 기존 파일이 가지고 있는 데이터에 이어서 내용을 추가할 때 사용하는 옵션이다.
  - 'w' 모드는 기존의 데이터를 완전히 삭제한다는 점에서 'a' 모드와 다르다.
- f.write(poem2)
  - 'a' 모드로 파일을 열고, write() 함수를 사용해서 파일에 쓰기 작업을 한다.
  - 새로운 데이터가 기존 파일 데이터에 이어서 추가된다.

# section106. 파일에 데이터 추가하기

## step 4.

3줄 요약

### • 3줄 요약 •

- 파일 열기 모드 중 'a' 모드는 기존 파일에 데이터를 추가할 때 선택한다.
- 새롭게 추가하는 데이터는 기존 파일 데이터의 맨 끝에 추가된다.
- 파일에 데이터를 저장할 때는 `write( )` 함수를 사용한다.

