

14. 파일 입출력

1. 파일 열기 (open)
2. 파일 입력 - 파일로부터 데이터 읽어 오기
3. 파일 출력 - 파일에 데이터 저장하기
4. print() 함수로 파일에 데이터 저장하기
5. 파일 경로
6. 정리

1. 파일 열기 (open)



1. 파일 열기 (open)

◆ 파일 열기

- 프로그램을 수행하는 중에 파일에 저장된 데이터를 읽어 오거나, 파일로 데이터를 저장하려면 가장 먼저 해야 할 일이 프로그램과 파일을 연결하는 일임 → 이를 ‘**파일을 연다(file open)**’이라고 함.
- open() 함수

파일 객체 = open(**파일명**, 모드)

↑
파일 열기가 성공하면 파일 객체가 하나 생성됩니다.

1. 파일 열기 (open)

◆ 파일 열기

파일 객체 = open(파일명, 모드)

파일 열기가 성공하면 파일 객체가 하나 생성됩니다.

모드		설명
읽기 모드	r	파일로부터 데이터를 읽어오기 위해 사용하는 모드(디폴트 모드).
쓰기 모드	w	파일에 데이터를 저장(write)하기 위해 사용하는 모드. 만약에 첫 번째 인수에 없는 파일명을 넣으면, 새로 파일을 만들고, 이미 존재하는 파일명을 넣으면 기존의 파일 내용을 모두 지우고 새로운 데이터 저장.
추가 모드	a	파일에 데이터를 추가(append)하기 위해 사용하는 모드. 만약에 첫 번째 인수에 이미 존재하는 파일을 넣으면, 파일의 기존 내용은 그대로 두고 파일의 끝에 새로운 내용을 추가. 만약에 존재하지 않는 파일명을 넣으면 새로 파일을 생성하고 내용을 저장.

1. 파일 열기 (open)

◆ 파일 열기 예제

movie.txt

Alice in Wonderland is a fantasy adventure film.
It is directed by Tim Burton.
The film was produced by Walt Disney Pictures.
It was shot in the United Kingdom and the USA.

- ① `f = open('movie.txt', 'r')`
- ② `contents = f.read()`
- ③ `print(contents)`
- ④ `f.close()`

[결과]

Alice in Wonderland is a fantasy adventure film.
It is directed by Tim Burton.
The film was produced by Walt Disney Pictures.
It was shot in the United Kingdom and the USA.

1. 파일 열기 (open)

◆ 파일 열기 예제

```
#fileread.py
```

```
f = open('movie.txt', 'r')  
contents = f.read()  
print(contents)  
f.close()
```

movie.txt

Alice in Wonderland is a fantasy adventure film.
It is directed by Tim Burton.
The film was produced by Walt Disney Pictures.
It was shot in the United Kingdom and the USA.

파일 오픈이 성공하면 파일 객체가 파일에 붙습니다.
이 파일 객체를 통해서 read/write합니다.

1. 파일 열기 (open)

◆ 파일 객체

- open() 함수의 반환값은 파일 객체임.
- 파일 객체가 사용할 수 있는 메소드 목록

```
>>> f = open('movie.txt', 'r')
>>> dir(f)          # dir(파일 객체)를 하면 파일 객체가 사용할 수 있는 메소드가 나오죠.
[... 'buffer', 'close', 'closed', 'detach', 'encoding', 'errors', 'fileno', 'flush',
'isatty', 'line_buffering', 'mode', 'name', 'newlines', 'read', 'readable', 'readline',
'readlines', 'seek', 'seekable', 'tell', 'truncate', 'writable', 'write', 'writelines']
```

- 파일로부터 데이터를 읽어 올 때 사용하는 메소드 - read(), readline(), readlines()
- 파일에 데이터를 저장할 때 사용하는 메소드 - write(), writelines()

1. 파일 열기 (open)

◆ 파일 객체

- open() 함수를 with ~ as 구문에서 사용하기

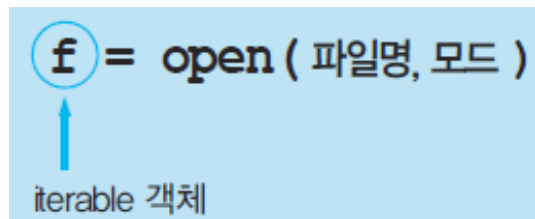
```
with open('파일명', 모드) as f:    # f = open('파일명', 모드)와 같습니다(콜론 필요).
    s = f.read()                  # with ... as 구문을 사용하면 들여쓰기를 해야 합니다.
    print(s)
f.close()
```

- 파일 객체는 iterable함.

```
>>> f = open('movie.txt')    # 모드가 생략되어 있으면 'r' 모드입니다.
>>> iter(f)                  # iter()의 인수가 iterable 자료형이면 TypeError가 나지 않습니다.
<_io.TextIOWrapper name='movie.txt' mode='r' encoding='cp949'>
>>> list(f)                  # list() 함수는 iterable 인수를 입력받아서 리스트로 변환합니다.
['Alice in Wonderland is a fantasy adventure film.', 'It is directed by Tim Burton.', 'The
film was produced by Walt Disney Pictures.', 'It was shot in the United Kingdom and the
USA.']
```


1. 파일 열기 (open)

◆ 파일 객체



`f = open (파일명, 모드)`
↑
iterable 객체

```
f = open('movie.txt', 'r')  
for line in f:  
    print(line)
```

[결과]

Alice in Wonderland is a fantasy adventure film.

It is directed by Tim Burton.

The film was produced by Walt Disney Pictures.

It was shot in the United Kingdom and the USA.

1. 파일 열기 (open)

◆ 파일 객체

- 한글 읽기 - 한글 읽으려면 한글에 대한 encoding 정보를 open() 함수에 넣어야 함.

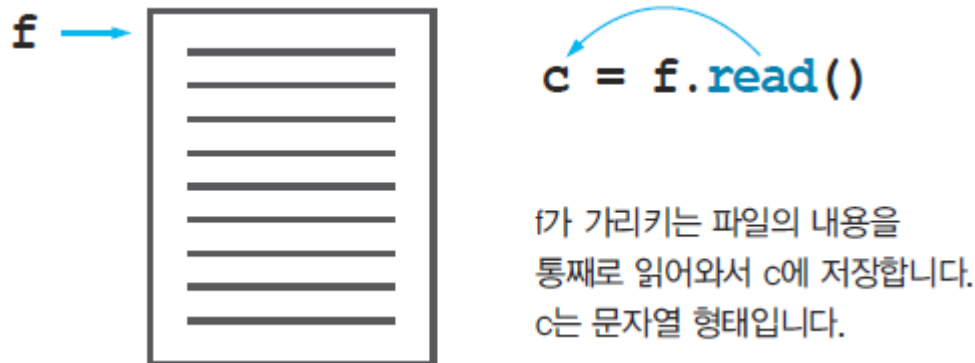
<pre>f = open('greet.txt', encoding='utf-8') c = f.read() print(c) f.close()</pre>	〈 파일 greet.txt 〉 안녕하세요. 만나서 반갑습니다.	[결과] 안녕하세요. 만나서 반갑습니다.
--	--	----------------------------------

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ read() 메소드

- 파일의 내용을 문자열로 읽어 오는 메소드.
- 인수는 없거나 한 개임. 읽어온 데이터를 문자열로 반환함.

인수	없음	파일 전체 데이터를 문자열로 읽어 옵니다.
	1개	인수로 정수를 넣을 수 있는데, 읽어 오려는 문자의 개수를 넣어 주어야 해요.
반환값		읽어 온 문자열을 반환합니다.



2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ read() 메소드

- 파일의 내용을 문자열로 읽어 오는 메소드.
- 인수는 없거나 한 개임. 읽어온 데이터를 문자열로 반환함.

CODE 78 파일 'python.txt'에 저장된 단어의 개수를 출력하는 프로그램을 작성해 볼게요.

```
f = open('python.txt')           # 파일을 엽니다.  
contents = f.read()              # 파일의 내용을 통째로 문자열로 읽어 옵니다.  
words = contents.split()         # contents를 스페이스로 잘라 리스트 words에 저장합니다.  
print("words in 'python.txt' : {}".format(len(words)))    # words 길이를 출력합니다.  
f.close()
```

< 파일 python.txt >

Python is a widely used high-level programming language.
It is created by Guido van Rossum.
It is released in 1991.

[결과]

words in 'python.txt' : 20

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ read() 메소드

```
f = open('python.txt')          # 파일을 엽니다.  
contents = f.read()             # 파일의 내용을 통째로 문자열로 읽어 옵니다.  
words = contents.split()        # contents를 스페이스로 잘라 리스트 words에 저장합니다.  
print("words in 'python.txt' : {}".format(len(words)))    # words 길이를 출력합니다.  
f.close()
```



```
f = open('python.txt')  
contents = f.read()  
print("words in 'python.txt' : {}".format(len(contents.split())))  
f.close()
```



```
print("words in 'python.txt' : {}".format(len(open('python.txt').read().split())))
```

2. 파일 입력 - 파일로부터 데이터 읽어 오기

CODE 79 파일명을 입력받고, 그 파일에 특정 단어가 있는지 알고 싶은 경우에는 다음과 같이 프로그램을 작성할 수 있어요.

```
filename = input('Enter file name : ')      # 파일명을 입력받습니다.
word = input('Enter word to search : ')     # 찾고자 하는 단어를 입력받습니다.
f = open(filename)
contents = f.read()                        # 파일의 모든 내용이 contents 변수에 저장됩니다.
if word in contents:                       # 문자열 contents에 word가 있는지 판단합니다.
    print("{} ' is in ' {}'.format(word, filename))
else: print("{} 'is not in ' {}'.format(word, filename))
f.close()
```

〈 파일 how_are_you.txt 〉

How are you, my friend?
How are you today?
May I come in?

[결과]

Enter file name : how_are_you.txt
Enter word to search : you
'you' is in 'how_are_you.txt'

[결과]

Enter file name : how_are_you.txt
Enter word to search : tomorrow
'tomorrow' is not in 'how_are_you.txt'

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ read(m) - 인수가 한 개인 경우

- m개의 문자를 읽어 옴.

```
f = open('python.txt')  
contents = f.read(15)  
print(contents)  
f.close()
```

[결과]

Python is a wid

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ read(m) - 인수가 한 개인 경우

```
f = open('python.txt') # 파일 열기
contents = f.read(15) # 파일에서 15문자 읽어 오기
print(contents)
print('-' * 15)

contents = f.read(10) # 파일에서 10문자 읽어 오기
print(contents)
print('-' * 10)

contents = f.read(20) # 파일에서 20문자 읽어 오기
print(contents)
print('-' * 20)
f.close()
```

[결과]

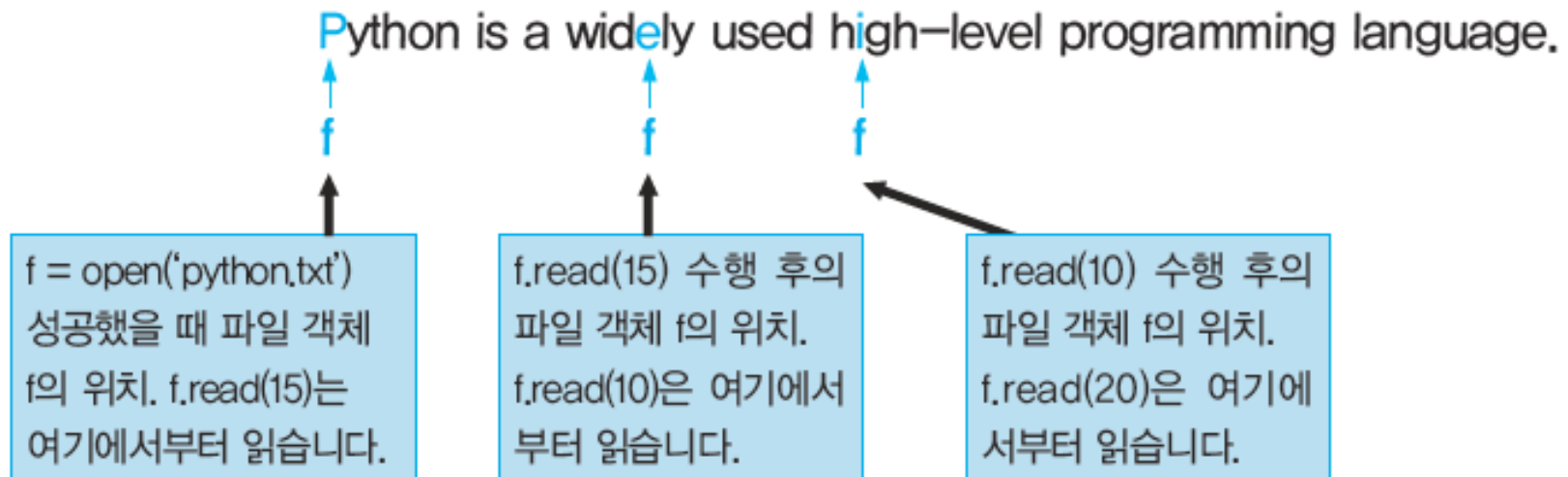
```
Python is a wid
-----
ely used h
-----
igh-level programmin
-----
```

< 파일 python.txt >

Python is a widely used high-level programming language.
It is created by Guido van Rossum.
It is released in 1991.

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ read(m) - 인수가 한 개인 경우



2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ read() 메소드 두 번 연속해서 호출하기

〈파일 python.txt〉

Python is a programming language.
It is created by Guido van Rossum.
It is released in 1991.

```
f = open('python.txt')
data = f.read() # 파일 통째로 읽어옴.
print(data)
print('-' * 20)
data = f.read() # 파일 객체가 파일 끝에 있음.
print(data)
f.close()
```

이 때 읽어올 데이터 없음

[결과]

Python is a programming language.
It is created by Guido van Rossum.
It is released in 1991.

>>>

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ 파일을 두 번 읽고 싶은 경우 - close() 후 다시 open() 하기

```
f = open('python.txt')
contents = f.read()
print(contents)
f.close()           # 파일 연결 끊기

print('-' * 20)

f = open('python.txt') # 파일 다시 열기
contents = f.read()
print(contents)
f.close()
```

[결과]

Python is a programming language.
It is created by Guido van Rossum.
It is released in 1991.

Python is a programming language.
It is created by Guido van Rossum.
It is released in 1991.

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ 파일을 두 번 읽고 싶은 경우 - seek() 메소드 이용하기

```
f = open('python.txt')
```

```
c = f.read()
```

```
print(c)
```

```
f.seek(0,0)
```

seek(0,0)파일의 맨 처음으로 파일 객체를 이동시킵니다.

```
c = f.read()
```

```
print(c)
```

```
f.close()
```

2. 파일 입력 - 파일로부터 데이터 읽어 오기

- ◆ readline() 메소드는 한 줄씩 읽어 오는 메소드임
- ◆ readline() - 인수가 없는 경우

코드

```
f = open('line.txt')  
① line = f.readline() # 한 줄 읽기  
② print(line)  
③ line = f.readline() # 한 줄 읽기  
④ print(line)  
⑤ line = f.readline() # 한 줄 읽기  
⑥ print(line)  
f.close()
```

결과

파일에 저장할 때 [enter]를 누른 부분에 '\n'이 저장됩니다.

< line.txt >

```
first line\nsecond line\nthird line\n
```

first line

second line

third line

공백

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ 앞의 코드에서 'Wn' 없애기

1	2	3
<pre>f = open('line.txt') line = f.readline() print(line[:-1]) line = f.readline() print(line[:-1]) line = f.readline() print(line[:-1]) f.close()</pre>	<pre>f = open('line.txt') line = f.readline() print(line, end='') line = f.readline() print(line, end='') line = f.readline() print(line, end='') f.close()</pre>	<pre>f = open('line.txt') line = f.readline().strip() print(line) line = f.readline().strip() print(line) line = f.readline().strip() print(line) f.close()</pre>

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ readline(m) - 인수가 한 개인 경우

- m개의 문자를 읽어 옵니다.
- m이 한 줄의 문자 수보다 크다면, 그냥 한 줄만 읽어 옵니다.

코드

```
f = open('line.txt')
a = f.readline(5) # 5 문자를 읽어 옵니다.
print(a)
f.close()
```

'first'만 a에 저장됩니다. ←

결과

< line.txt >

```
first line\n
second line\n
third line\n
```

first

코드

```
f = open('line.txt')
a = f.readline(30) # 한 줄의 문자 개수보다 큼.
print(a)          # 한 줄만 읽어 옵니다.
f.close()
```

결과

< line.txt >

```
first line\n
second line\n
third line\n
```

first line

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ readline(m)과 while 반복문 사용하기

코드

```
f = open('python.txt')  
line = f.readline()
```

맨 처음에 첫줄이 저장됩니다.

```
while line :  
    print(line, end='')  
    line = f.readline()
```

```
f.close()
```

맨 마지막 줄까지 읽고 나면 빈문자열을 읽습니다.

이 자리에 빈 문자열 ' '이 오면 False가 되어 while 루프를 빠져 나갑니다.

결과

```
f = open('python.txt')
```

```
while True :
```

```
    line = f.readline()
```

```
    if not line: break
```

```
    print(line, end='')
```

```
f.close()
```

Python is a programming language.
It is created by Guido van Rossum.
It is released in 1991.

line이 빈문자열을 갖게 되면 break가 걸리게 됩니다.

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ readline(m)과 for 반복문 사용하기

변수 오픈한 파일 객체

for line in f:

인덴트 → } for 블록

```
f = open('python.txt')
for line in f:
    print(line.strip())
f.close()
```

↑ 각 줄 끝에 있는 'n'을 제거합니다.

Python is a programming language.
It is created by Guido van Rossum.
It is released in 1991.

```
f = open('python.txt')
for line in f:
    print(line)
f.close()
```

[결과]

Python is a programming language.
It is created by Guido van Rossum.
It is released in 1991.

>>>

2. 파일 입력 - 파일로부터 데이터 읽어 오기

CODE 80 for 반복문을 이용하여 반복 처리를 하는 코드 예제입니다. 파일명과 찾고자 하는 문자열을 입력받아서 파일 안에 입력받은 문자열이 몇 번째 라인에 몇 번씩 나오는지를 출력합니다.

```
filename = input('Enter file name : ') # 파일명 입력받습니다.
word = input('Enter word to search : ') # 파일에서 찾고자 하는 단어를 입력받습니다.
f = open(filename)
line_number = 1
for line in f:
    if word in line:
        print('line {} - {}'.format(line_number, line.count(word)))
    line_number += 1
f.close()
```

[결과]

Enter file name : wash_face.txt

Enter word to search : wash

line 1 - 1

line 2 - 2

line 3 - 1

[입력파일]

< wash_face.txt >

This is the way we wash our face,
wash our face, wash our face,

This is the way we wash our face,
so early in the morning.

2. 파일 입력 - 파일로부터 데이터 읽어 오기

◆ readlines() 메소드

- 파일 전체를 읽어 와서 각 줄들로 구성된 리스트 객체를 반환함.

코드 1

```
f = open('line.txt')
t = f.readlines()
print(t)
f.close()
```

코드 2

```
with open('line.txt') as f :
    t = f.readlines()
    print(t)
f.close()
```

입력 파일

```
< line.txt >
first line
second line
third line
```

결과

```
['first line\n', 'second line\n', 'third line\n']
```

코드

```
with open('line.txt') as f :
    t = f.readlines() # f에서 통째로 읽어 옴
    for x in t: ←.....리스트
        print(x, end='')
f.close()
```

결과

```
first line
second line
third line
```

2. 파일 입력 - 파일로부터 데이터 읽어 오기

CODE 81 `read()` 메소드로 파일을 통째로 읽어 온 다음에 한 줄씩 리스트에 저장합니다. 그리고 앞에 라인 번호를 붙여서 각 라인을 출력하는 코드를 작성해 보겠습니다

코드

```
f = open('how_are_you.txt')
contents = f.read() # 파일 통째로 읽기
lines = contents.split('\n') # '\n' 제거

no = 1
for line in lines:
    print('{} : {}'.format(no, line))
    no += 1

f.close()
```

(Note: In the original image, arrows indicate that '\n' is used for line separation and 'no' is incremented for each line.)

입력 파일

< how_are_you.txt >

How are you, my friend?
How are you today?
May I come in?

결과

1 : How are you, my friend?
2 : How are you today?
3 : May I come in?
4 :

(Note: In the original image, the number 4 is circled in red.)

다음 코드는 결과 마지막 줄에 나오는 4: 을 없앤다

2. 파일 입력 - 파일로부터 데이터 읽어 오기

CODE 81 read() 메소드로 파일을 통째로 읽어 온 다음에 한 줄씩 리스트에 저장합니다. 그리고 앞에 라인 번호를 붙여서 각 라인을 출력하는 코드를 작성해 보겠습니다

코드

```
f = open('how_are_you.txt')
contents = f.read()
lines = contents.split('\n')
no = 1
for line in lines:
    if line == '': # 추가된 부분
        break
    print('{} : {}'.format(no, line))
    no += 1

f.close()
```

읽은 줄이 빈 문자열이면 루프를 끝냅니다.

입력 파일

< how_are_you.txt >

How are you, my friend?
How are you today?
May I come in?

결과

1 : How are you, my friend?
2 : How are you today?
3 : May I come in?

2. 파일 입력 - 파일로부터 데이터 읽어 오기

CODE 82 파일 score.txt에는 한 줄에 하나의 성적이 저장되어 있어요. 파일에 있는 성적들을 모두 읽어서 리스트에 저장한 다음에 리스트를 출력하려고 합니다. 성적을 읽어 와서 그냥 리스트에 저장하면 문자열로 저장되기 때문에 int() 함수를 적용해서 정수 데이터로 리스트에 저장해야 합니다. 그리고 성적의 평균도 소수점 두 자리에 맞추어 출력합니다.

코드

```
f = open('score.txt')
score = [] ← 성적을 저장할 빈 리스트를 만들어 둡니다.
for line in f:
    score.append(int(line)) ← 한 줄씩 읽어서 int로 변환하여 score에 저장합니다.
print('score : ', score)
print('average : {:.2f}'.format(sum(score)/len(score)))
f.close()
```

입력 파일

< score.txt >

90
88
79
95
78
92

결과

```
score : [90, 88, 79, 95, 78, 92]
average : 87.00
```

2. 파일 입력 - 파일로부터 데이터 읽어 오기

CODE 83 이번에는 파일 score.txt 각 줄에 숫자가 두 개씩 저장되어 있습니다. 첫 번째 숫자는 학번이고 두 번째 숫자는 성적이에요. 이 경우에는 학번과 성적을 쌍으로 묶어서 사전에 저장하면 좋겠죠. 다음은 한 줄씩 읽어서 '학번:성적'으로 묶어서 사전에 저장하는 코드입니다. 이때도 학번과 성적은 정수형으로 변환하여 저장합니다.

코드

```
f = open('score.txt')
score = {} # 빈 사전 만들어 놓기
for line in f:
    (key, val) = line.split()
    score[int(key)] = int(val)

for key, val in score.items():
    print('{} - {}'.format(key, val))
f.close()
```

입력 파일

파일에서 한 줄을 읽어서 스페이스 기준으로 잘라서(key, val)에 저장합니다.
key, val은 모두 문자열입니다.

< score.txt >

```
201812345 90
201712345 88
201800001 95
201800100 77
201710010 85
```

결과

```
201812345 - 90
201712345 - 88
201800001 - 95
201800100 - 77
201710010 - 85
```

사전 score에 학번을 키로, 성적을 값으로 저장합니다.

2. 파일 입력 - 파일로부터 데이터 읽어 오기

CODE 84 파일 exam.txt는 다음과 같습니다. 각 줄 첫 번째에는 시험명이 적혀 있고, 나머지는 그 시험에 등록한 사람들 이름이에요. 예를 들어서, TOEFL 시험에 등록한 사람은 Alice, Paul, David, Bob이에요. 파일의 정보를 모두 읽어서 사전에 저장하려고 해요. 이때 시험명이 키가 되고 학생들 이름을 값으로 저장하면 되겠죠. 학생들 이름은 리스트로 관리하려고 해요.

코드

```
f = open('exam.txt')
exam = {} # 빈 사전을 만듭니다.
for line in f:
    items = line.split()
    key, values = items[0], items[1:]
    exam[key] = values

for key, val in exam.items():
    print('{} Test - {}'.format(key, val))
f.close()
```

입력 파일

< exam.txt >

```
TOEFL Alice Paul David Bob
TOEIC Cindy Stella Bill
GRE Henry Jenny Jessica Erin Tim
GMAT John Joe Tom
```

items[0]

items[1:]

```
TOEFL Test - ['Alice', 'Paul', 'David', 'Bob']
TOEIC Test - ['Cindy', 'Stella', 'Bill']
GRE Test - ['Henry', 'Jenny', 'Jessica', 'Erin', 'Tim']
GMAT Test - ['John', 'Joe', 'Tom']
```


3. 파일 출력 - 파일에 데이터 저장하기

◆ 파일 출력할 때 사용하는 모드

w	없는 파일명에 'w'모드를 적어주면 파일을 새로 만듭니다. 존재하는 파일명에 'w'모드를 적어주면 <u>기존의 내용을 모두 지우고</u> 새로운 내용을 덧씌웁니다.
a	없는 파일명에 'a'모드를 적어주면 파일을 새로 만듭니다. 존재하는 파일명에 'a'모드를 적어주면 기존의 내용은 그대로 두고, 기존의 내용 끝에 새로 입력하는 데이터를 추가합니다.

3. 파일 출력 - 파일에 데이터 저장하기

◆ write() 메소드 - 'w' 모드

- 괄호 안에 내용을 파일에 저장하는 메소드 (없는 파일에 'w'할 때)

```
f = open('greeting.txt', 'w')
f.write('Nice to meet you!')
f.close()
```

현재 없는 파일, 이 경우
'greeting.txt'란 이름의 파
일을 만들어 줍니다.

↑
write() 메소드는 괄호 안에 내용을 파일에 저장합니다.
괄호에는 반드시 문자열을 넣어야 합니다.

< greeting.txt >

Nice to meet you!

- 존재하는 파일에 'w' 수행할 때

```
# 같은 코드를 파일에 저장할 내용을 바꿔서 한 번 더 수행합니다.
f = open('greeting.txt', 'w')
f.write('How are you?')
f.close()
```

< greeting.txt >

How are you?

3. 파일 출력 - 파일에 데이터 저장하기

◆ write() 메소드 - 'a' 모드

'greeting.txt' 파일이 이미 있으니까, 원래 내용 뒤에 새로운 내용을 추가합니다.

```
f = open('greeting.txt', 'a')  
f.write('Good morning!')  
f.close()
```

< greeting.txt >

How are you?Good morning!

3. 파일 출력 - 파일에 데이터 저장하기

CODE 85 파일 이름을 input() 함수를 이용해서 받습니다. 다음으로 input() 함수를 이용해서 수학 성적을 다섯 개 입력받아서 파일에 저장하려고 합니다.

코드

```
입력받은 파일명을 'w'모드로 open합니다.  
filename = input('Enter filename : ')  
f = open(filename, 'w')  
for count in range(5):  
    score = input('Enter score : ')  
    f.write(score)  
    f.write('\n')  
f.close()
```

루프를 5회 실행합니다.
score는 문자열입니다.
이 줄이 없으면?

결과

```
Enter filename : math.txt  
Enter score : 90  
Enter score : 88  
Enter score : 100  
Enter score : 70  
Enter score : 80
```

< math.txt >

```
90  
88  
100  
70  
80
```

3. 파일 출력 - 파일에 데이터 저장하기

CODE 86 파일을 복사하는 프로그램을 작성해 볼게요. original.txt 파일의 내용을 읽어서 copy.txt 파일에 저장하는 거예요.

코드

```
rf = open('original.txt')
wf = open('copy.txt', 'w')

for line in rf:
    wf.write(line)

rf.close()
wf.close()
```

읽어 올 파일 'original.txt'를 open 합니다.

복사본 파일 'copy.txt'를 'w' 모드로 생성합니다.

original.txt에서 한 줄씩 읽어와서 copy.txt에 저장합니다.

< original.txt >

```
hello world
python programming
have a nice day
```

< math.txt >

프로그램 수행 후에 copy.txt 파일이 생긴 것을 확인하면 됩니다.

3. 파일 출력 - 파일에 데이터 저장하기

◆ writelines() 메소드

- readlines() 메소드와 반대로 동작하는 메소드.
- 문자열로 구성된 리스트를 파일에 저장하는 메소드.

```
f = open('greeting2.txt', 'w')
L = ['good morning\n',
     'how are you\n',
     'nice to meet you\n',
     'good afternoon\n']
f.writelines(L)
f.close()
```

writelines() 메소드 인수로는
리스트를 넣어야 합니다.

< greeting2.txt >


```
good morning
how are you
nice to meet you
good afternoon
```

4. print() 함수로 파일에 데이터 저장하기

◆ print() 함수에 file 키워드 인수를 이용함

The diagram illustrates the process of writing to a file using the `print()` function. It features a light blue background with a dotted border. On the left, a code snippet is shown: `with open('newfile.txt', 'w') as f:` followed by two indented `print()` statements. The first `print('hello world', file = f)` has `file = f` circled in blue. The second `print('python programming', file = f)` also has `file = f` circled in blue. A blue box labeled `file =` with a small blue arrow icon is positioned above the first `print()` statement. A dotted arrow points from this box to the `file = f` parameter in the first `print()` statement. Another dotted arrow points from the `file = f` parameter in the second `print()` statement to the same box. To the right of the code, a text box contains the text: `< newfile.txt >` followed by `hello world` and `python programming` on separate lines, representing the contents of the file.

open한 파일 객체를 넣어주면 모니터 출력을
하지 않고 해당 파일로 출력합니다.

```
file = 
```

```
with open('newfile.txt', 'w') as f:  
    print('hello world', file = f)  
    print('python programming', file = f)
```

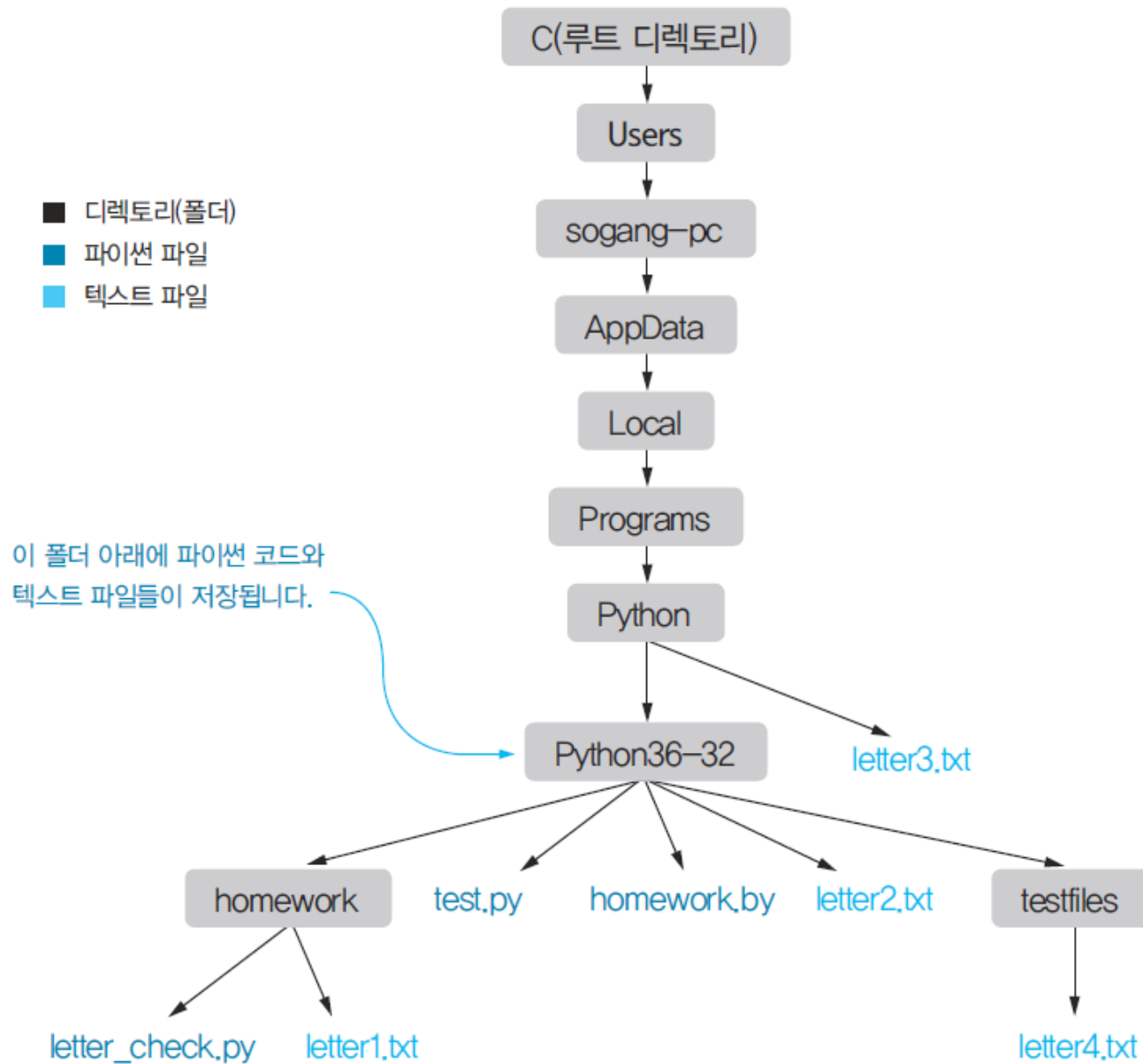
< newfile.txt >

```
hello world  
python programming
```

5. 파일 경로

- ◆ `open()` 메소드에 파일명을 넣을 때, 절대 경로와 상대 경로 두 가지 방식으로 파일 위치를 넣을 수 있음.
 - 절대 경로 - 가장 상위 폴더부터의 경로
 - 상대 경로 - 현재 파일의 위치에서부터의 경로

5. 파일 경로



5. 파일 경로

◆ 앞의 디렉토리 구조에서 다음과 같이 수행하려고 한다.

- 코드 letter_check.py에서 letter2.txt를 open하려고 할 때,

절대 경로 이용

```
open('C:/Users/sogang-pc/AppData/Local/Programs/Python/Python36-32/letter2.txt')
```

상대 경로 이용

```
open('../letter2.txt')
```

 부모 폴더(하나 상위 폴더)

- 코드 letter_check.py에서 letter3.txt를 open하려고 할 때,

절대 경로	<code>open('C:/Users/sogang-pc/AppData/Local/Programs/Python/letter3.txt')</code>
상대 경로	<code>open('../..letter3.txt')</code>

6. 정리

- ◆ 많은 데이터를 파일에 저장해 두고 입력 데이터로 사용하면 많은 데이터를 한꺼번에 처리할 수 있음.
- ◆ `open()` 내장 함수는 코드와 파일을 연결하는 함수임.
- ◆ `open()` 함수를 이용할 때는 파일의 절대 경로와 상대 경로를 알아야 함.
- ◆ 파일로부터 데이터를 읽어올 때는 `read()` 메소드를 사용하고, 파일에 데이터를 저장할 때는 `write()` 메소드를 사용해야 함.