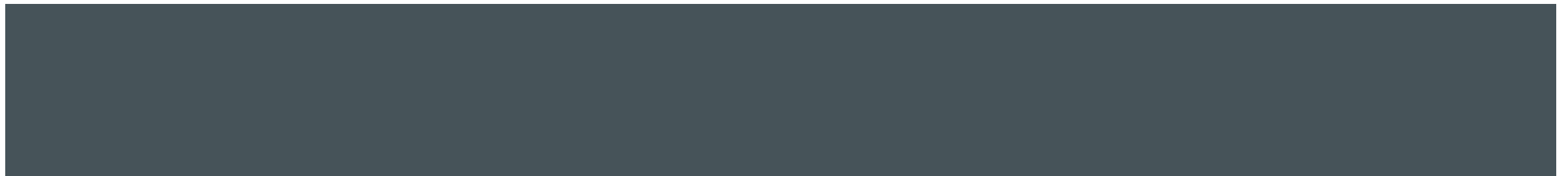


파이썬 기초

파일



파일



파일에서 데이터 읽기

파일을 연다.



파일에서 데이터를
읽거나 쓴다.



파일을 닫는다.

파일객체

파일이름

파일 열기 모드

```
f = open("input.txt", "r")
```

...

```
f.close()
```

파일 열기 모드	모드 이름	설명
"r"	읽기 모드 (read)	파일을 읽기만 할 때 사용. 파일의 처음부터 읽는다.
"w"	쓰기 모드 (write)	파일에 쓸 때 사용. 파일의 처음부터 쓴다. 파일이 없으면 생성된다. 파일이 존재하면 기존의 내용은 지워진다.
"a"	추가 모드 (append)	파일의 마지막에 새로운 내용을 추가 시킬 때 사용 파일이 없으면 생성된다.

파일 쓰기

■ `f = open("new.txt", "w")`

write.py

```
f = open("new.txt", "w")  
  
for i in range(1, 6):  
    data = "%d line\n" % i  
    f.write(data)  
  
f.close()
```

new. txt

```
1 line  
2 line  
3 line  
4 line  
5 line
```

파일 읽기(1/2)

(1) *readline()*

파일에서 한 줄 읽어옴

```
f = open("new.txt", "r")
line = f.readline()
print (line, end="")
line = f.readline()
print (line, end="")
f.close()
```

```
1 line
2 line
>>>
```

(2) *readlines()*

모든 라인을 한꺼번에 읽어서 각각의 줄을 요소로 갖는 리스트를 반환

```
f = open("new.txt", "r")
lines = f.readlines()
print (lines)
f.close()
```

```
['1 line\n', '2 line\n', '3 line\n', '4 line\n', '5 line\n']
>>>
```

파일 읽기(1/2)

(3) *f.read()*:

파일을 전부 읽은 문자열을 반환

```
f = open("new.txt", "r")
```

```
data = f.read()
```

```
print (data)
```

```
f.close()
```

```
1 line  
2 line  
3 line  
4 line  
5 line
```

```
>>>
```

(4) *for* 문 사용

```
f = open("new.txt", "r")
```

```
for line in f:  
    print (line, end="")
```

```
f.close()
```

```
1 line  
2 line  
3 line  
4 line  
5 line  
>>> |
```

파일 - 실습 1

- 파일에 있는 각각의 단어의 수 구하기

test.txt

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

실행예시

```
line 3  
second 1  
third 1  
first 1  
>>>
```

(힌트) 딕셔너리 사용

파일 - 실습 2

- 파일명을 입력 받아, 해당 파일을 한 줄씩 읽어 파일의 내용을 모두 대문자로 출력하는 프로그램을 작성하시오.

(힌트)

- 문자열을 대문자로 변환

```
>>> "hello".upper()
```

```
'HELLO'
```

- 문자열을 소문자로 변환

```
>>> "World".lower()
```

```
'world'
```

- `os.path.exists(파일명)`: 파일이나 디렉토리가 존재하는지 검사

```
>>> import os
```

```
>>> os.path.exists("test.txt")
```

```
True
```

```
Enter a file name: test.txt
FIRST LINE
SECOND LINE
THIRD LINE
>>>
```

파일에 내용 추가하기

```
f = open("new.txt", "a")

for i in range(6, 11):
    data = "%d line\n" % i
    f.write(data)

f.close()
```

new.txt

1 line
2 line
3 line
4 line
5 line
6 line
7 line
8 line
9 line
10 line

추가된 내용

파일을 열고 자동으로 닫기(with~as)

- open() ~ close() 사용

```
f = open( "output.txt", "w" )  
f.write("Python is fun!")  
f.close()
```

- with open() as 사용

```
with open( "output.txt", "w" ) as f :  
    f.write("Python is fun!")
```

with 블록을 벗어나는 순간 파일을 자동으로 닫아준다.

파일 포인터

- 파일 포인터
 - 파일의 현재 위치를 가리키는 것
- *f.tell()*: 현재 파일 포인터의 위치 반환
- *f.seek()*: 지정하는 곳으로 포인터의 위치를 변경

test.txt

first line
second line
third line

```
>>> f = open("test.txt", "r")
>>> f.tell()
0
>>> f.readline()
'first line\n'
>>> f.tell()
11
>>> 
```



파일(참고자료)



파이썬 프로그램에 입력인수 전달

- sys 모듈
- 도스(Dos)나 리눅스셸에서 파이썬 명령어를 실행하면서 입력인수를 넣어줄 수 있음.

Ex) 도스명령어 [인수1 인수2]

```
#sys1.py
import sys

args = sys.argv[1:]
for i in args:
    print (i)
```

```
$ python sys1.py aaa bbb ccc
```

```
c:\>python sys1.py aaa bbb ccc
```

argv[0]

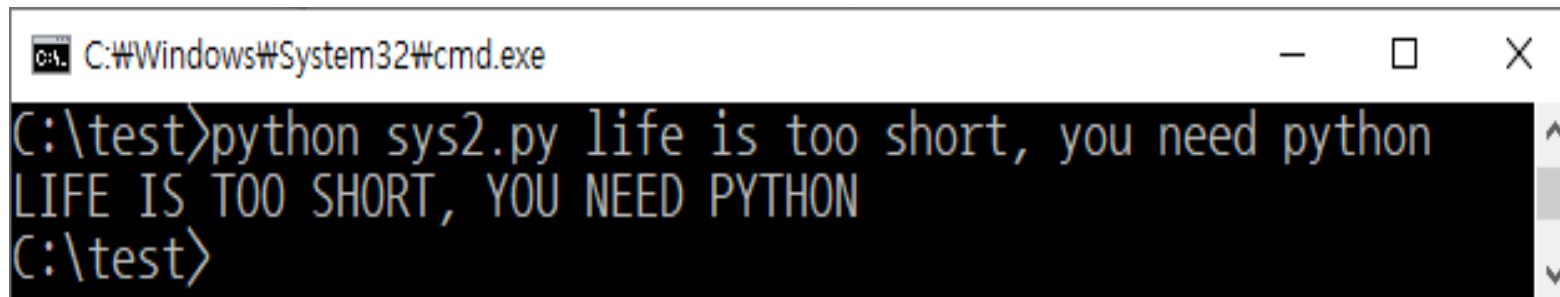
argv[3]

```
C:\test>python sys1.py this is test
this
is
test
C:\test>
```

입력인수 전달 실습

- 명령행에 입력된 소문자를 대문자로 바꾸어 주는 프로그램을 작성하자.

```
c:\>python sys2.py life is too short, you need python
```



A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\System32\cmd.exe'. The command prompt shows the command 'python sys2.py life is too short, you need python' being executed. The output is 'LIFE IS TOO SHORT, YOU NEED PYTHON'. The prompt is 'C:\test>'.

```
C:\Windows\System32\cmd.exe
C:\test>python sys2.py life is too short, you need python
LIFE IS TOO SHORT, YOU NEED PYTHON
C:\test>
```

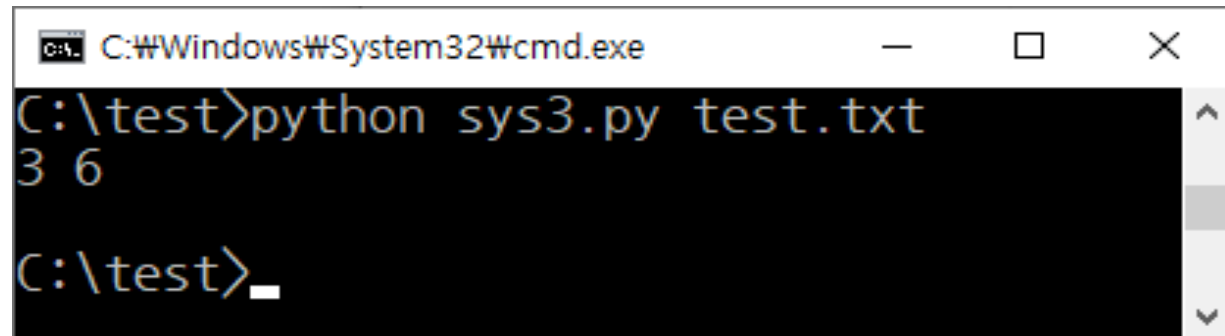
실습문제

- 파일의 라인 수, 단어 수를 구하여 출력하는 프로그램을 작성하시오.(단, 단어는 공백으로 구분된 문자열을 의미한다고 가정)

```
c:\>python sys3.py test.txt
```

test.txt

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



```
C:\Windows\System32\cmd.exe  
C:\test>python sys3.py test.txt  
3 6  
C:\test>
```