

기초빅데이터프로그래밍

파일 입출력



목차

- 1 파일
- 2 파일 생성 및 열기
- 3 텍스트 파일 쓰기
- 4 텍스트 파일 읽기
- 5 직렬화/역직렬화
- 6 with문
- 7 성적 처리 시스템 구현을 통한 파일 입출력 알아보기



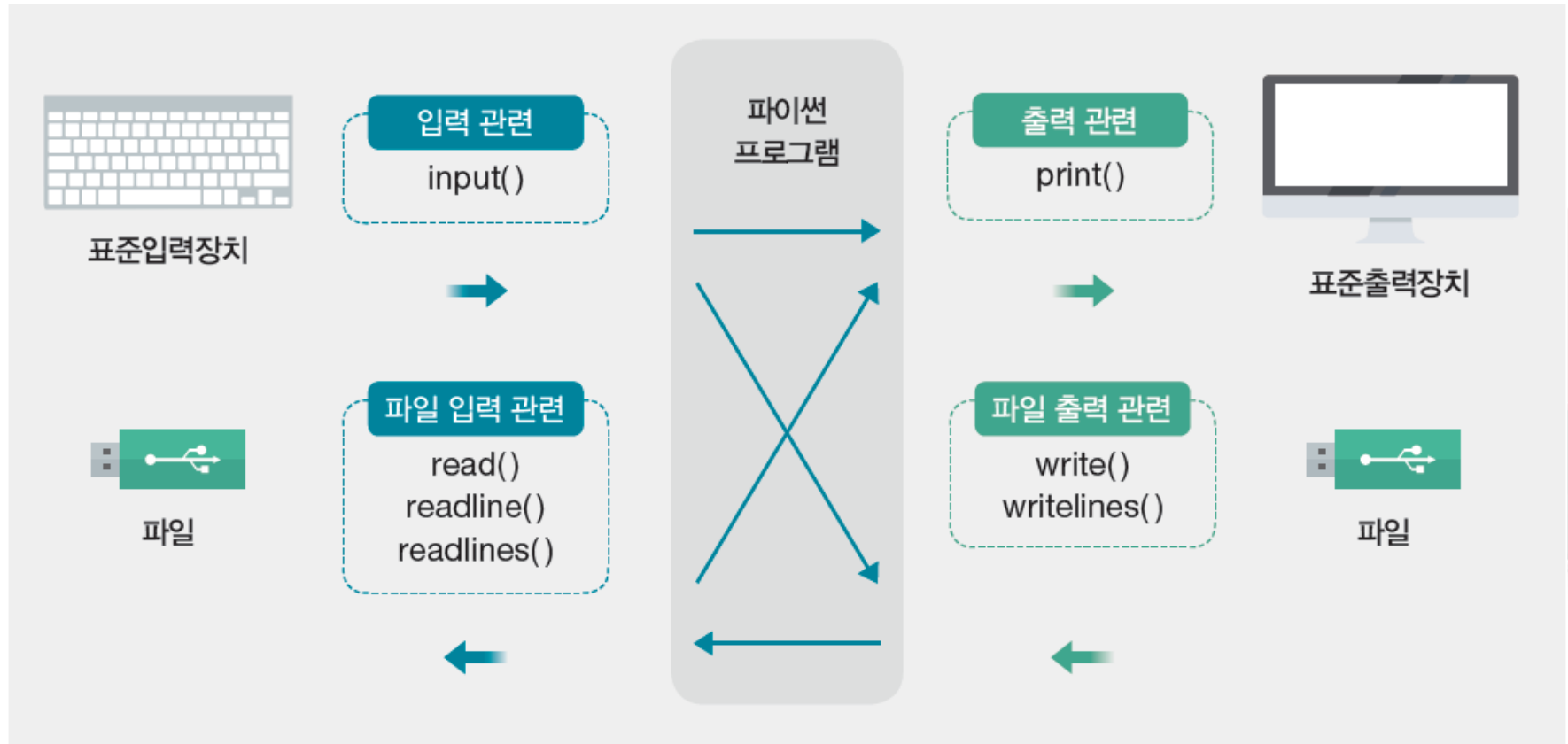
1 파일

- 운영체제는 하드 디스크를 파일 단위로 다룬다.
 - 저장해야 할 정보를 파일의 형태로 관리한다.
- 프로그래밍 언어에서는 프로그램에서의 정보를 파일에 저장할 수 있는 방법을 제공한다.



파일 입출력 이해

■ 표준 입출력과 파일 입출력 함수



2 파일 생성 및 열기

- 파일 입출력을 하기 위해서는 우선 파일을 생성하고 열기를 해야 한다.
- 파일을 생성하거나 열기 위해 open 함수를 이용한다.

open(파일_이름, [파일_열기_모드], [encoding = 인코딩_방식])

-> **파일_객체**

- 주어진 **파일_이름**과 **파일_열기_모드**에 따른 파일 객체를 열어서 돌려준다.

파일_객체.close()

- 열린 파일 객체를 닫기 위해서 close() 메서드를 호출한다.

파일의 열기 모드

파일 열기 모드	의미
생략	r과 동일
r	읽기 모드, 기본값
w	쓰기 모드, 기존에 파일이 있으면 덮어씀.
r+	읽기/쓰기 겸용 모드
a	쓰기 모드, 기존에 파일이 있으면 이어서 씀. Append의 약자
t	텍스트 모드, 텍스트 파일을 처리. 기본값
b	바이너리 모드, 바이너리 파일(=이진 파일)을 처리

- 파일 열기모드는 다양한 형태로 조합이 가능하며 조합에 따라 복합된 의미를 띄게 된다.
- 기본 모드는 'rt' (텍스트 읽기를 위해서 연다)이다.
- 파일을 쓰기 모드 'w'로 열게 되면 기존에 파일이 있던 경우는 해당 파일의 내용이 잘려나가고 없던 경우에는 파일을 새로이 생성한다.
- 파일을 첨부모드 'a'로 열게 되면 기존 파일의 내용을 지우지 않고 내용의 끝에 첨부할 수 있도록 해준다.

- encoding 옵션에 처리할 파일의 데이터 인코딩 방식을 지정할 수 있다.
 - 특별한 지정이 없는 경우 시스템의 기본 설정에 따라서 처리한다.
 - Windows – ‘cp949’
 - Linux – ‘utf-8’
- **Text mode**
 - 기본 모드
 - 모든 데이터를 문자열 형태로 다룸
 - 모든 데이터를 ASCII 또는 UNICODE로 입출력
- **Binary mode**
 - 데이터를 문자열형태로 바꾸지 않고 데이터 자체를 입출력

◆ file open 후에 사용할 수 있는 메소드

읽 기	read()	read() - 파일 내용을 모두 읽어서 문자열(str)로 반환
		read(n) - 파일에서 n 바이트 읽어서 문자열(str)로 반환
	readline()	한 줄씩 읽어서 문자열(str)로 반환한다.
	readlines()	파일 전체를 리스트(list)로 반환한다.
쓰 기	write()	문자열을 파일에 저장한다.
	writelines()	문자열 리스트를 파일에 저장한다.

3 텍스트 파일 쓰기

- 파일을 연후 파일에 데이터를 쓰기 위해서 write 메서드를 호출한다.

파일_객체.write(데이터)

```
# wirte_hello.py
```

```
if __name__ == "__main__":  
    fp = open("hello.txt", "wt", encoding="cp949")  
    fp.write("Hello World!")  
    fp.close( )
```

```
if __name__ == "__main__":  
    fp = open("hello.txt", "wt", encoding="cp949")  
    fp.write("Hello World!\n")  
    fp.write("Hello Sea \n")  
    fp.write("Hello C! \n")  
    fp.write("%d\n" % 99)  
    fp.write("%.2f\n" % 99.9)  
    fp.write("서강대학교")  
    fp.close()
```

```
with open('hello.txt', 'r') as f:  
    print(f.read())
```

```
Hello World!  
Hello Sea  
Hello C!  
99  
99.90  
서강대학교
```

텍스트 모드에서는
모든 데이터를 문자
열로 바꾼 후 write을
통해 출력한다.

```
if __name__ == "__main__":
    fp = open("hello.txt", "wt", encoding="cp949")
    fp.write("Hello World!\n")
    fp.write("Hello C! \n")
    fp.write("%d\n" %99)
    fp.write("%.2f\n" %99.9)
    fp.write("서강대학교\n")
    a=["abcd", "1234"]
    fp.write("\n".join(a))
    fp.writelines(a)
    b=["abcd\n", "1234\n"]
    fp.write("\n".join(b))
    fp.writelines(b)
    fp.close()
```

```
with open('hello.txt', 'r') as f:
    print(f.read())
```

```
Hello World!
Hello C!
99
99.90
서강대학교
abcd
1234abcd1234abcd
```

```
1234
abcd
1234
```

- **join** 함수는 리스트를 문자열 형태로 만들어 준다.
- 앞에 `\n`은 각 요소 사이에 삽입할 문자열이다.

- `writelines` 함수도 인수의 타입만 다르고 `write()`와 같으므로 필요하면 줄 단위로 `\n`을 직접 넣어야 한다.

4 텍스트 파일 읽기

- 텍스트 파일을 읽는 데는 세 가지 방법이 있다.
 - readline 메서드
 - readlines 메서드
 - read 메서드

파일_객체.readline() -> 문자열
한 줄을 읽어서 반환한다.

파일_객체.readlines() -> 리스트
끝까지 읽은 후 각 줄을 리스트 객체에 담아서 반환한다.

파일_객체.read() -> 문자열
끝까지 읽은 데이터를 문자열 형태로 반환한다.

◆ read()와 read(n)

```
f = open('newfile.txt', 'r')
```

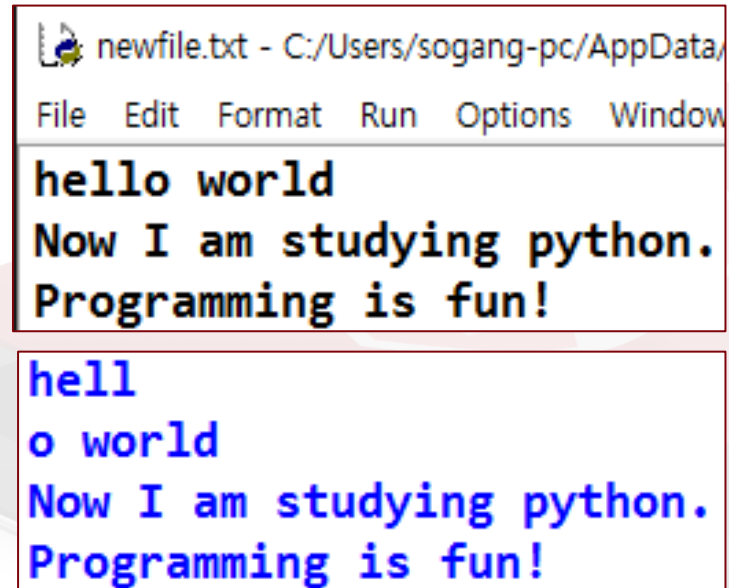
```
a = f.read(4)
```

```
print(a)
```

```
b = f.read()
```

```
print(b)
```

```
f.close()
```



```
%%writefile FileTest.txt
Building Machine Learning Systems with Python
Python Machine Learning
Machine Learning in Action
행복하세요
```

Overwriting FileTest.txt

```
f=open('FileTest.txt', 'rt', encoding='utf-8')
a=f.read(50)
print(a)
b=f.read(50)
print(b)
c=f.read(4)
print(c)
f.close()
```

```
Building Machine Learning Systems with Python
Pyth
on Machine Learning
Machine Learning in Action
행복하
세요
```

read()

```
if __name__ == "__main__":  
    fp = open("hello.txt", "rt")  
    contents = fp.read()  
    print(contents)  
    fp.close()
```

```
Hello World!  
Hello Sea  
Hello C!  
99  
99.90  
서강대학교
```

read()은 끝까지 읽어 **문자열** 객체로 반환한다.

```
contents
```

```
'Hello World!\nHello Sea \nHello C! \n99\n99.90\n서강대학교'
```

파일 입력

◆ 한 줄씩 읽어 오기 - readline()

```
f = open('about.txt', 'r')
```

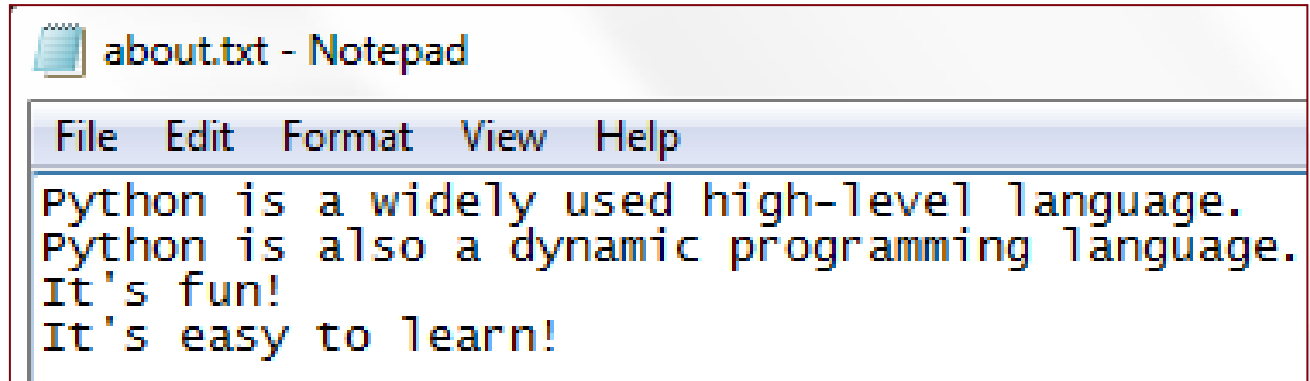
```
a = f.readline()
```

```
print(a)
```

```
b = f.readline()
```

```
print(b)
```

```
f.close()
```



```
Python is a widely used high-level language.  
  
Python is also a dynamic programming language.  
  
>>>
```



```
f=open('FileTest.txt', 'rt', encoding='utf-8')
a=f.readline() #한줄씩 읽어오기
print(a)
b=f.readline()
print(b)
for line in f:
    print(line)
f.close()
```

Building Machine Learning Systems with Python

Python Machine Learning

Machine Learning in Action

행복하세요

```
f=open('FileTest.txt', 'rt', encoding='utf-8')
a=f.readlines() #파일전체를 리스트로
print(a)
f.close()
```

```
['Building Machine Learning Systems with Python\n', 'Python Machine Learning\n', 'Machine Learning in Action\n', '행복하세요']
```

readline()

```
if __name__ == "__main__":  
    fp = open("hello.txt", "rt")  
    line = fp.readline()  
    print(line.strip())  
    line = fp.readline()  
    print(line.strip())  
    line = fp.readline()  
    print(line.strip())  
    line = fp.readline()  
    print(line.strip())  
    line = fp.readline()  
    print(line.strip())  
    line = fp.readline()  
    print(line.strip())  
    fp.close
```

```
Hello World!  
Hello Sea  
Hello C!  
99  
99.90  
서강대학교
```

strip()는 문자열의 시작과 끝의 공백 문자를 제거한다. 즉, \n을 제거한다.

readline()은 한 줄 읽어 문자열 객체로 반환한다.

```
if __name__ == "__main__":  
    fp = open("hello.txt", "rt")  
    for i in range(0,6):  
        line = fp.readline()  
        print(line)  
    fp.close
```

```
Hello World!  
  
Hello Sea  
  
Hello C!
```

```
99  
  
99.90
```

서강대학교

readlines()

```
if __name__ == "__main__":  
    fp = open("hello.txt", "rt")  
    lines = fp.readlines()  
    print(lines)  
    fp.close()
```

```
['Hello World!\n', 'Hello Sea \n', 'Hello C! \n', '99\n', '99.90\n', '서강대학교']
```

readlines()은 끝까지 읽어 **리스트** 객체로 반환한다.

실습 1

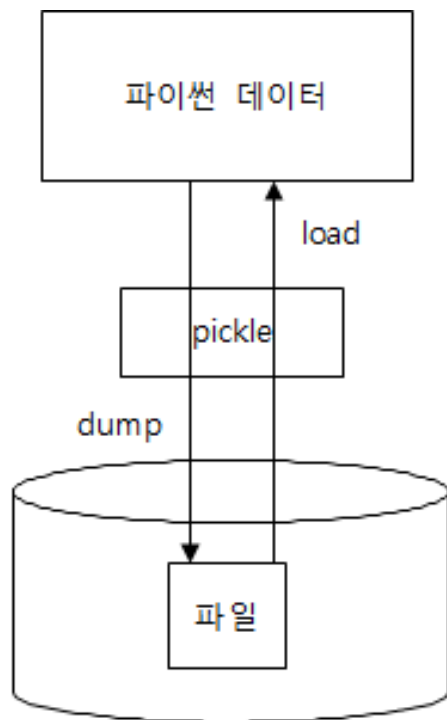
◆ 파일 내용을 통째로 읽어 들여 한 줄씩 사전에 저장 하시오

- ✓ 입력파일 "ban_stu.txt" 은 write()을 이용하여 만드시오.
- ✓ 입력 파일 각 줄의 첫 번째 자료는 키로, 나머지 자료는 값으로 하여 사전에 저장하시오.

```
2 Alice Paul David Bob
4 Cindy Stella Bill
1 Henry Jenny Jessica Erin Tim
3 John Joe Tom
```

5 직렬화/역직렬화

- Binary file을 다루는 방법
- **직렬화/역직렬화(serialization/Deserialization)**는 파이썬 데이터를 파일에 직접적으로 **쓰고 읽을 수** 있도록 해준다.
- 파이썬에서는 직렬화/역직렬화를 위해 **pickle** 모듈을 제공한다.



- **pickle.dump**(데이터_객체, 파일_객체)
 - 데이터 객체를 파일에 쓴다.
- **pickle.load**(파일_객체) -> 데이터 객체
 - 파일로부터 데이터 객체를 읽어 온다.

```
import pickle
if __name__ == "__main__":
    fp = open("binary.dat", "wb")
    pickle.dump(1, fp)
    pickle.dump(3.14, fp)
    pickle.dump("안녕 C!", fp)
    pickle.dump([11, 22, 33], fp)
    pickle.dump((11, 22, 33), fp)
    pickle.dump({"line": 0, "rectangle": 1, "triangle": 2}, fp)
    fp.close
```

이 binary.dat 파일은 텍스트 에디터로는 볼 수 없고 load()로 역직렬화를 해야 볼 수 있다.

```
import pickle
if __name__ == "__main__":
    fp = open("binary.dat", "rb")
    for i in range(0, 6):
        data = pickle.load(fp)
        print(data)
    fp.close()
```

```
1
3.14
안녕 C!
[11, 22, 33]
(11, 22, 33)
{'line': 0, 'rectangle': 1, 'triangle': 2}
```

```
a=['-3\n', '-3.14\n', '\\"-3.14\\"n']
with open("2019Info.txt", 'wt') as fp:
    fp.writelines(a)
```

```
with open('2019Info.txt', 'rt') as fp:
    for i in fp:
        print(i)
```

```
-3
-3.14
\\"-3.14"
```

```
with open('2019Info.txt', 'rt') as fp:
    test = fp.read()
    print(test)
```

```
-3
-3.14
\\"-3.14"
```

```
import pickle
with open('2019Info.dat', 'wb') as fp:
    pickle.dump(-3, fp)
    pickle.dump(-3.14, fp)
    pickle.dump("\\"-3.14\\"'", fp)
```

```
fp = open('2019Info.dat', 'rb')
while True:
    try:
        print(pickle.load(fp))
    except:
        break
fp.close()
```

```
-3
-3.14
\\"-3.14"
```

```
with open('2019Info.dat', 'rb') as fp:
    print(pickle.load(fp))
    print(pickle.load(fp))
    print(pickle.load(fp))
```

```
-3
-3.14
\\"-3.14"
```


pickle 모듈의 dump() & load()

```
colors=['red', 'green', 'black']
colors
```

```
['red', 'green', 'black']
```

colors를 file에 저장

```
import pickle
f=open('coloT', 'wb')
pickle.dump(colors, f)
f.close()
```

file에서 colors 읽어오기

```
del colors
```

```
colors
```

```
UsageError: %colors: you must specify a color
'%colors?'
```

```
f=open('coloT', 'rb')
colors = pickle.load(f)
f.close()
colors
```

```
['red', 'green', 'black']
```

Class의 객체를 파일에 저장하고 읽어오기

```
class Test:
    var =None
a=Test()
a.var="varTest"
f=open('claT', 'wb')
pickle.dump(a, f)
f.close()
f=open('claT', 'rb')
b=pickle.load(f)
f.close()
b
```

```
<__main__.Test at 0x1fa04187dd8>
```

```
b.var
```

```
'varTest'
```


Binary mode로 파일 복사하기

```
f= open('BDP_Mheri.mp4', 'wb')
```

```
f.write(open('BDP_다중상속.mp4', 'rb').read())
```

```
60814453
```

```
f.close()
```

text file 복사하기

```
f= open('test.py', 'w')
```

```
f.write(open('add.py', 'r').read())
```

```
46
```

```
f.close()
```

```
f.closed
```

```
True
```

text mode에서는 일반 문자열과 같은 인코딩이 적용되어, binary data를 다룰때는 오류발생함

```
f= open('BDP_Mheri2.mp4', 'w')
```

```
f.write(open('BDP_다중상속.mp4', 'r').read())
```

UnicodeDecodeError Traceback (most recent call last)

<ipython-input-8-a4a2c44eff47> in <module>()

```
----> 1 f.write(open('BDP_다중상속.mp4', 'r').read())
```

UnicodeDecodeError: 'cp949' codec can't decode byte 0xe8 in position 63: illegal multibyte sequence

6 with 문

- 파이썬에서의 with문은 작업 블록을 보호하고 **파일을 열고 닫는 것을 자동으로 완료해준다.**
- with 문을 사용하면 파일 조작 부분의 가독성이 높아지고 안전해진다.



```
# with_statement.py
```

```
if __name__ == "__main__":
```

```
    with open("text.txt", "rt", encoding="utf-8") as fp:
```

```
        lines = fp.readlines()
```

```
        for line in lines:
```

```
            print(line.strip())
```

```
print("Done!")
```

with block이 끝나면 파일이
닫히므로 fp를 더 이상 쓸 수
없다.

7 성적 처리 시스템 구현을 통한 파일 입출력 알아보기

- 파일 입출력을 통한 성적 처리는 대규모의 성적 처리를 가능하게 한다 .
- 학생들의 성적 정보를 포함하는 파일을 성적 처리 시스템에게 넘겨주면 시스템은 파일로부터 학생 정보를 이용하여 학생 객체들을 만들고 이를 시스템에 등록한다.
- 학생 객체들의 등록이 끝나면 성적 처리 작업을 수행하고 처리된 결과는 결과 파일에 출력한다.
- csv(Comma Separated Value): 각 요소들이 콤마로 분리된 텍스트 파일이다.

실습 2

1. information.txt라는 텍스트 파일을 생성하고 다음의 데이터를 그 파일에 출력하는 프로그램을 작성하시오 (write 메소드를 이용하시오).
3
3.14
"3.14"
2. 위의 information.txt를 읽어서 화면에 출력하는 프로그램을 작성하시오.
3. information.dat라는 파일을 생성하고 상기의 데이터를 직렬화하는 프로그램을 작성하시오.
4. 위의 information.dat를 역직렬화하고 화면에 출력하는 프로그램을 작성하시오.