Python 프로그래밍 기초

파일 입출력

파일 입출력 개요
: 파일의 생성과 파일 모드

```
파일객체 = open({파일명}, {파일모드}[, encoding='인코딩'])
```

파일 모드	설명
r (default)	읽기 모드 - 파일을 읽기만 할 때 사용
w	쓰기 모드 - 파일에 내용을 기록할 때 사용
a	추가 모드 - 파일의 마지막에 사로운 내용을 추가할 때 사용

파일 모드	설명
t (default)	텍스트 모드
b	바이너리 모드

파일 입출력 개요
: 파일 제어 기본 함수

함수명	설명
open	파일을 생성한다
write	파일에 내용을 기록한다
read	파일에서 내용을 읽어온다
close	파일 사용을 끝낸다. 파일을 열었으면(open) 반드시 사용후 닫아주도록 한다

```
>>> # File Write Sample
>>>
>>> f = open('text.txt', 'w', encoding='utf-8') # text.txt, 쓰기모드
>>> write_size = f.write("Life is too short, You need Python")
>>> print(write_size)
33
>>> f.close() # 반드시 닫아주자
```

파일 입출력 개요
: 텍스트 파일 예제

```
# File Write

f = open('test.txt', 'w', encoding='utf-8')
for i in range(1, 10):
    f.write("%d: Life is too short, You need Python\n" % i)
f.close()

# File Read

f = open('test.txt', 'r', encoding='utf-8')
text = f.read()
print(text)
f.close()
```

파일 입출력 개요
: 텍스트 파일 예제 - write and read

```
# File Write

f = open('multilines.txt', 'w', encoding='utf-8')
for i in range(1, 10):
    f.write("%d: Life is too short, You need Python\n" % i)
f.close()

# File Read

f = open('multilines.txt', 'r', encoding='utf-8')
text = f.read()
print(text)
f.close()
```

파일 입출력 개요
: readline 함수를 이용한 텍스트 파일 읽기

- readline 함수를 이용하면 텍스트 파일을 줄 단위로 읽어올 수 있다

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

while True:
    line = f.readline()
    if not line:
        break # 무한루프 탈출
    print(line)

f.close()
```

파일 입출력 개요
: readlines 함수를 이용한 텍스트 파일 읽기

- readlines 함수를 이용하면 모든 라인을 불러 리스트로 제공한다

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

lines = f.readlines()
# print(lines)

for line in lines:
    print(line)

f.close()
```

파일 입출력 개요
: 바이너리(Binary) 파일 다루기

- 바이너리 파일을 다루려면 모드를 바이너리로 지정해야 한다

```
>>> f = open('python.png')
>>> f.read()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/namsk/.pyenv/versions/3.4.3/lib/python3.4/codecs.py", line
319, in decode
    (result, consumed) = self._buffer_decode(data, self.errors, final)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x89 in position 0:
invalid start byte
```

파일 입출력 개요
: 바이너리(Binary) 파일 다루기

- 바이너리 파일을 다루려면 모드를 바이너리로 지정해야 한다

```
>>> # copy binary sample
>>>
>>> f_src = open('python.png', 'rb') # 바이너리 읽기 모드
>>> data = f_src.read()
>>> f_src.close()
>>>
>>> f_dest = open('python_copy.png', 'wb') # 바이너리 쓰기 모드
>>> f_dest.write(data)
11155
>>> f_dest.close()
```

파일 입출력 개요
: 그 외 파일 관련 함수

함수명	설명
seek	사용자가 원하는 위치로 파일 포인터 이동
tell	현재 파일에서 어디까지 읽고 썼는지 위치를 반환

```
f = open('multilines.txt', 'r', encoding='utf-8')

text = f.readline()
print(text)
pos = f.tell() # 현재의 파일 포인터를 얻어옴
print(pos)

f.seek(16)
text = f.read()
print(text)
```

파일 입출력 개요
: with ~ as – 자동 자원 정리

- with ~ as 를 이용, 파일 입출력을 수행하면 수동으로 파일을 close 하지 않아도 된다

```
with open('multilines.txt', 'r') as f_as:
    for line in f_as.readlines():
        print(line, end = "")

print(f_as.closed) # 파일이 close 되었는지 점검
```

Using Pickle

- 객체의 내용을 파일에 저장하거나 복원해야 할 경우에 Pickle 모듈을 사용하면 편리
- Pickle 모듈은 객체를 파일에 썼다가 나중에 복원할 수 있도록 객체를 바이트 스트림으로 직렬화
  - 모든 파이썬의 객체를 저장하고 읽을 수 있음
  - 원하는 객체를 형태 변환 없이 쉽게 쓰고 읽을 수 있다
- Pickle 모듈을 사용하려면 import pickle 을 이용, 모듈을 로드해야 한다
- Pickle 모듈 주요 메서드

메서드	설명
dump(data, file [, protocol])	data 객체를 [protocol 을 이용해] file에 저장
load(file)	File로부터 저장된 객체를 불러옴

Using Pickle
: 객체의 저장:Pickling - dump

- file에 객체를 저장하고자 할 때에는 dump 메서드를 이용한다

```
import pickle
f = open("players.bin", "wb")
data = {"baseball": 9}
pickle.dump(data, f)
f.close()
```

- dump 메서드에 프로토콜 버전을 정의해 줄 수 있다
  - 최신 프로토콜 버전을 확인하려면 pickle.HIGHEST_PROTOCOL로 확인

```
...
pickle.dump(data, f, pickle.HIGHEST_PROTOCOL)
...
print(pickle.HIGHEST_PROTOCOL)
```

Using Pickle
: 객체의 복원:Unpickling - load

- file에 객체로부터 객체를 불러올 때에는 load 메서드를 이용한다

```
import pickle
f = open("players.bin", "rb")
data = pickle.load(f)
f.close()
print(data)
```

- dump시에 PROTOCOL을 지정했다 하더라도 load할 때는 지정해주지 않아도 된다
  - pickle 파일에 PROTOCOL 버전이 저장되어 있음

Using Pickle
: 복수 객체의 저장

- 기본적으로 Pickle은 단일 객체를 저장하는 포맷이지만, dump 메서드를 중복하여 사용하면 복수 개의 객체를 저장할 수 있다

```
import pickle
with open("players.bin", "wb") as f:
    pickle.dump({"baseball": 9}, f)
    pickle.dump({"soccer": 11}, f)
    pickle.dump({"basketball": 5}, f)
```

Using Pickle
: 복수 객체의 복원

- 저장된 객체를 복원하려면 load 메서드를 이용
  - load가 수행될 때마다 한줄씩 불러들이며 더 이상 불러올 객체가 없을 때 EOFError 발생
  - 다음 코드를 수행해 보고 무엇이 문제인지 확인해 봅니다

```
>>> import pickle
>>> with open("players.bin", "rb") as f:
...     print(pickle.load(f))
...     print(pickle.load(f))
...     print(pickle.load(f))
...     print(pickle.load(f))
...
{'baseball': 9}
{'soccer': 11}
{'basketball': 5}
Traceback (most recent call last):
  File "<stdin>", line 5, in <module>
EOFError: Ran out of input
```

Using Pickle
: 복수 객체의 복원

- [Solution]

```
>>> import pickle
>>> with open("players.bin", "rb") as f:
...     data_list = []
...     while True:
...         try:
...             data = pickle.load(f)
...         except EOFError:
...             break
...         data_list.append(data)
...
>>> print(data_list)
[{'baseball': 9}, {'soccer': 11}, {'basketball': 5}]
```

Using Pickle

- Pickle 사용시 유의사항
  - Pickle은 단순 텍스트 저장이 아닌 바이트 스트림 직렬화를 이용한 것이므로 파일 모드는 반드시 "b" 모드 (wb / rb) 로 지정해야 한다
  - Pickle에 사용되는 데이터 포맷은 파이썬에 특화되어 있기 때문에 다른 언어로 작성된 응용프로그램과의 데이터 교환에는 사용하지 않는 것이 좋다
  - 저장된 데이터에 대한 보안을 제공하지 않는 점에 유의하여 사용