Lab.

전화번호부 파일입출력

Photo by Jordan Sanchez on Unsplash

준비

```
>>>a={'KIM':'111-1111', 'LEE':'222-2222'}
>>>f=open('a.db', 'w')
>>>for item in a.items():
>>>record = '{}, {}\n'.format(item[0], item[1])
>>>f.write(record)
>>>f.close()
```

```
C:\>type a.db
KIM, 111-1111
LEE, 222-2222
|
```

끝에 공백 한 줄 들어가 있음에 주의!

읽어온 데이터 (레코드) 정제

```
>>>b={}
>>>f=open('a.db', 'r')
>>>lines = f.readlines()
['KIM, 111-1111\n', 'LEE, 222-2222\n']
>>>f.close()
>>>b.clear()
>>>record = lines[0]
'KIM, 111-1111\n'
>>>record = record.split(',')
['KIM', ' 111-1111\n']
```

```
>>>name = record[0].strip()
'KIM'
>>>phone = record[1].strip()
'111-1111'


>>>b[name]=phone
{'KIM': '111-1111'}
```

읽은 것 모두 딕셔너리에 입력

```
b={}
f=open('a.db', 'r')
lines = f.readlines()
f.close()
b.clear()
for record in lines:
    record = record.split(',')
    name = record[0].strip()
    phone = record[1].strip()
    b[name]=phone
print(b)
```

```
{'KIM': '111-1111', 'LEE': '222-2222'}
```

딕셔너리 전화번호부 이전 코드 참조

```
import os

class PhoneBook(object):
    def __init__(self):
        self.phoneBook={}
        self.filename = 'phonebook.txt'

    def isEmpty(self):
        n = len(self.phoneBook)
        if n==0:
            return True
        else:
            return False
```

```
    def load(self):
        if os.path.isfile(self.filename)==False:
            print(self.filename,'Not Found')
            return

        f = open(self.filename, 'r')
        lines = f.readlines()
        f.close()

        self.phoneBook.clear()

        for record in lines:
            record = record.split(',')
            name = record[0].strip()
            phone = record[1].strip()

            self.phoneBook[name] = phone

        print(len(lines),'records loaded')
```

```
    def save(self):
        if self.isEmpty():
            print('Empty')
            return

        f = open(self.filename, 'w')

        for item in self.phoneBook.items():
            record = '{},{}\n'.format(item[0], item[1])
            f.write(record)

        f.close()

        print(len(self.phoneBook), 'record saved')
```

```
def Main():
    myPhoneBook= PhoneBook()
    print(‘\nstarting...')
    myPhoneBook.load()
 bFinish = False
 while not bFinish:
        print(‘\n-----------')
        print('[F] Find')
        print('[N] Insert')
        print('[U] Update')
        print('[D] Delete')
        print('[A] List All')
        print('[L] Load')
        print('[S] Save')
        print('[Q] Quit')
        print('-----------')
        menuList = ['F', 'N', 'U', 'D', 'A', 'Q', 'L', 'S']
```

```
        menu = ''
        while True:
            menu = input('Menu: ')
            menu = menu.strip().upper()
            print()
            if menu in menuList:
                break

        if menu == 'Q':
            print('Bye');
            bFinish = True
        elif menu =='F':
            myPhoneBook.search()
        elif menu == 'N':
            myPhoneBook.insert()
        elif menu == 'U':
            myPhoneBook.update()
```

```
            elif menu == 'U':
                myPhoneBook.update()
            elif menu == 'D':
                myPhoneBook.delete()
            elif menu == 'A':
                myPhoneBook.listAll()
            elif menu == 'L':
                myPhoneBook.load()
            elif menu == 'S':
                myPhoneBook.save()

        del myPhoneBook

Main()
```

Comma-Separated Values (CSV)

```
import csv
f = open('a.txt', 'w')
w=csv.writer(f)
w.writerow([123])
f.close()
```

iterable 객체만 허용

123은 '123'으로 간주

```
C:\>type a.txt
123

C:\>
```

```
>>>f = open('a.txt', 'r')
>>>lines = f.read()
>>>f.close()
>>>lines
'123\n\n'
```

이런 미세한 차이는 실행하고 확인해 봐야 알 수 있다.

CSV

import csv f=open('a.csv', 'w', newline='') w=csv.writer(f, delimiter=',') w.writerow([1,2,3]) w.writerow([4,5,6]) f.close()	import csv f=open('a.csv', 'r', encoding='utf-8') reader=csv.reader(f) for line in reader: print(line) f.close()
C:\>type a.csv 1,2,3 4,5,6	['1', '2', '3'] ['4', '5', '6']

숫자 데이터 편리하게 다루기 csv, np.loadtxt

```
import csv
data=[1,2,3]
with open('a.csv', 'w', newline='') as file:
    writer = csv.writer(file, delimiter=',') 숫자를 바로 저장할 수 있음
    writer.writerow(data)
```

```
f = open('a.csv', 'r')
lines = f.read()
print(lines)  '1,2,3\n'
f.close()
```

```
import numpy as np   행렬을 다룰 수 있는 패키지
data = np.loadtxt('a.csv', delimiter=',', dtype=np.float32)
print(data)  array([1., 2., 3.], dtype=float32)
```

C:₩>pip install numpy

```
1,2,3
```

```
[1., 2., 3.]
```

문자열 인코딩

```
>>>a=‘hello’
>>>type(a)
<class ‘str’>
>>>a.encode()
b'hello'
>>>type(a.encode())
<class ‘byte’>
>>>b = a.encode()
>>>c = b.decode(‘UFT-8’)
>>>type(c)
<class ‘str’>
```

text 파일, binary 파일, pickle

```
data = '1'
f = open('a.txt', 'w')
f.write(data)
f.close()
f = open('a.txt', 'r')
read_data=f.read()
print(read_data)
f.close()
```

```
C:\>type a.txt
1
```

```
import pickle
data = ‘1’
pickle.dump(data, open('a.dat','wb'))
read_data=pickle.load(open('a.dat','rb'))
print(read_data)
```

```
C:\>type d.dat
X   1q .
```

pickle은 이진 데이터

byte 데이터만 다룬다. 읽은 결과는 적절한 타입으로 변경

data = ‘1’ data = data.encode() f = open('a.dat', 'wb') f.write(data) f.close() f = open('a.dat', 'rb') read_data=f.read() print(read_data) f.close()	import pickle data = ‘1’ pickle.dump(data, open('a.dat','wb')) read_data=pickle.load(open('a.dat','rb')) print(read_data) 'str ' 로 변경해 주었음
b‘1’	1

리스트, 딕셔너리 데이터 다루기 편리 pickle

```
import pickle
data = {1:'apple',2:'banana',3:'graph'}
pickle.dump(data, open('a.dat','wb'))
read_data=pickle.load(open('a.dat','rb'))
print(read_data)
print(read_data[1])
```

저장할 때는 알아서 byte로 바꾸어 저장하고,
읽을 때는 알아서 dict로 바꾸어 준다.

```
{1: 'apple', 2: 'banana', 3: 'graph'}
apple
```

```
import pickle
data = [1,2,3]
pickle.dump(data, open('a.dat','wb'))
read_data=pickle.load(open('a.dat','rb'))
print(read_data)
print(read_data[1])
```

대신 몽땅 읽고 몽땅 쓰기만.

```
[1,2,3]
2
```

binary에 추가 모드

Pickle은 append 지원 안함

data = '1' data = data.encode() f = open('a.txt', 'ab') f.write(data) f.close() f = open('a.txt', 'rb') read_data=f.read() print(read_data) f.close()	import pickle data = '1' pickle.dump(data, open('a.dat','ab')) read_data=pickle.load(open('a.dat','rb')) print(read_data)
b'1' b'11'	1 1

예외처리

Photo by freestocks on Unsplash

예외들

```
>>>a = a + 1
NameError: name 'a' is not defined


>>>a=0
>>>b=1
>>>c=b/a
ZeroDivisionError: division by zero


>>a=[1,2]
>>a[2]
IndexError: list index out of range
```

```
>>>f=open('a.txt', 'w')
>>>f.write(123)
TypeError: write() argument must be str,
not int
```

예외들

```
try:
    a = a + 1
    a = 0
    c = 1 / a
    a = [1,2]
    a[2]
    f = open(‘a.txt’, ‘w’)
    f.write(123)
    f.close()
except:
    print(‘Exception’)
```

```
try:
    #code block
except TypeError:
    print(‘Type Error’)
except ZeroDivisionError:
    print('zero-division exception')
except FileNotFoundError:
    print('file not found exception')
```

예외 메시지

Type Error가 두 군데서 발생했고 누군지 알고 싶다면

```
try:
    f=open('a.txt', 'w')
    f.write(123)
    f.close()
except TypeError as e:
    print(e)
except:
    print('Exception')
```

```
write() argument must be str, not int
```

이렇게 하면 편하다.

```
try:
    f=open('a.txt', 'w')
    f.write(123)
    f.close()
except Exception as e:
    print(e)
```

```
write() argument must be str, not int
```

범용적인 예외처리

예외가 일어나지 않는다면, 예외 여부에 상관없이 꼭 처리해야 할 것이 있다면...

```
import sys
try:
    a = 1
    a = a + 1
except Exception as e:
    print(e)
    sys.exit()
else:
    print('no error')
finally:
    print('mandatory execution code')
```

내 코드에서 예외 객체 발생

내가 만든 함수에서 예외를 발생 시킬 수 도 있다.

```
def f():
    n = int(input('input a number: '))
    if n<1 or n>10:
        raise Exception('out of my range')
try:
    f()
except Exception as e:
    print(e)
```

```
Input a number: 11
out of my range
```