10장 파일과 예외처리

학습 목표

- 텍스트 파일 읽고 쓰기를 살펴본다.
- 이진 파일 읽고 쓰기를 살펴본다.
- 정규식을 사용하는 방법을 살펴본다.
- CSV 파일 읽고 쓰기를 살펴본다.
- 예외를 처리하는 방법을 살펴본다.

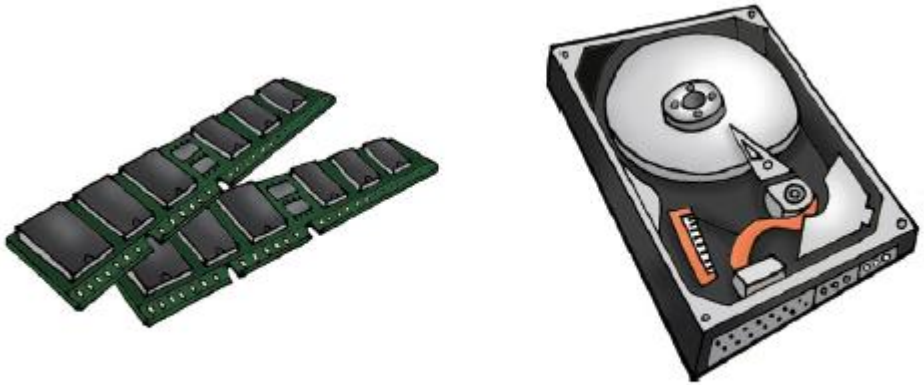
이번 장에서 만들 프로그램

```
_____
단어를 추측하시오: a
틀렸음!
9기회가 남았음!

_____
단어를 추측하시오: e
```

```
['날짜', '지점', '평균기온(℃)', '최저기온(℃)', '최고기온(℃)']
['1980-04-01', '108', '6.5', '3.2', '11.7']
['1980-04-02', '108', '6.5', '1.4', '12.9']
['1980-04-03', '108', '11.1', '4.1', '18.4']
['1980-04-04', '108', '15.5', '8.6', '21']
…
가장 추웠던 날은 –19.2 입니다.
```

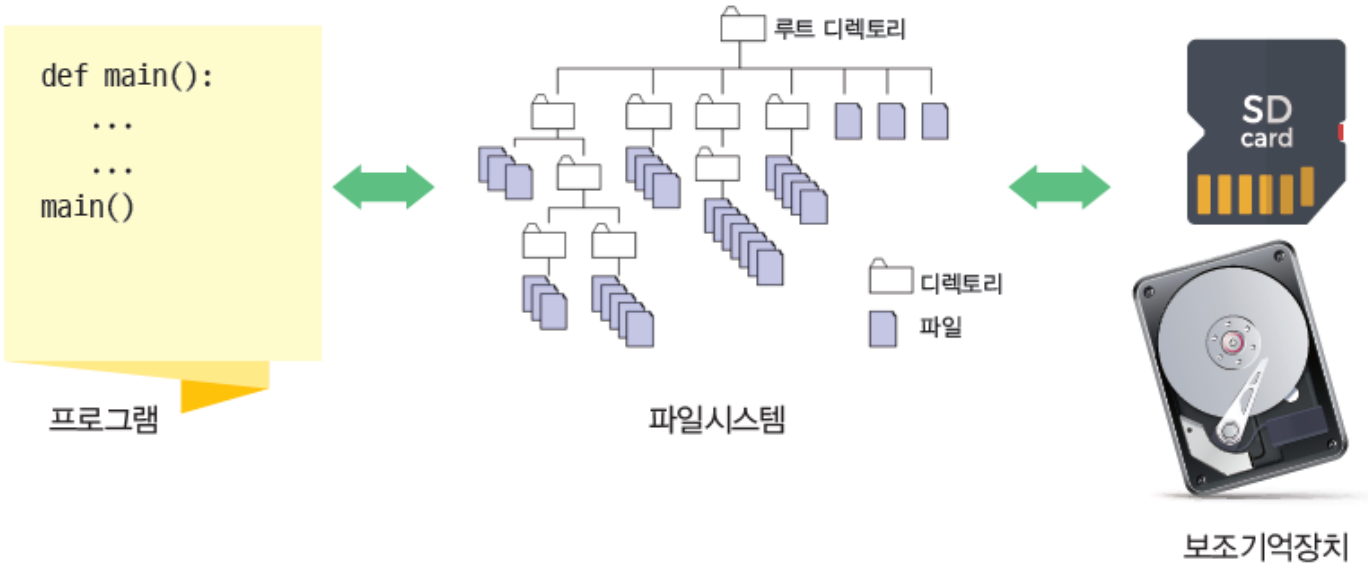
파일의 기초

- 프로그램에서 만든 데이터를 영구히 저장하고자 한다면 하드 디스크에 파일 형태로 저장하여야 한다.
- 큰 데이터 세트는 파일로 저장되는 것이 일반적. 공공 데이터 세트들도 CSV 파일 형태로 많이 제공된다.



파일의 개념

- 파일은 보조기억장치 상에서 논리적인 정보 단위이다.

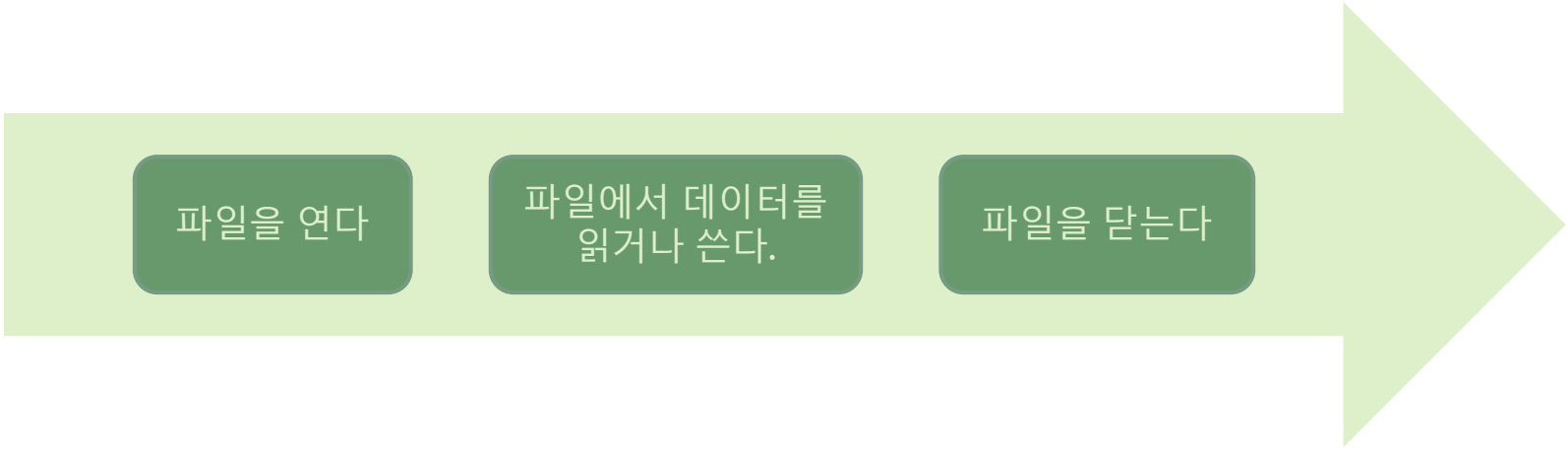


파일의 논리적인 구조

- 파일 안에는 바이트들이 순차적으로 저장되어 있고 맨 끝에는 EOF(end-of-file) 마커가 있다.
- 파일을 처음으로 열면, 파일 포인터는 파일의 첫번째 바이트를 가르킨다. 파일의 내용을 읽거나 쓰면 파일 포인터는 자동적으로 업데이트된다.

파일의 논리적인 구성

파일 열고 닫기



Syntax: 함수 정의

형식
```
파일객체 = open(파일이름, 파일모드)
파일객체.close()
```

예
```
infile = open("input.txt", "r")
...
infile.close()
```

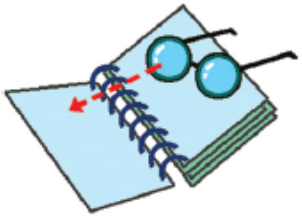
파일 객체

파일의 이름(name)

파일을 여는 모드(mode)

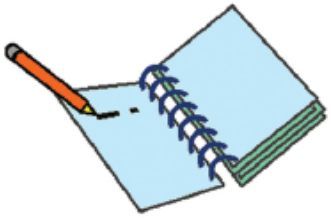
파일 모드

파일 모드	모드 이름	설명
"r"	읽기 모드(read mode)	파일의 처음부터 읽다.
"w"	쓰기 모드(write ode)	파일의 처음부터 쓴다. 파일이 없으면 생성된다. 만약 파일이 존재하면 기존의 내용은 지원진다.
"a"	추가 모드(append mode)	파일의 끝에 쓴다. 파일이 없으면 생성된다.
"r+"	읽기와 쓰기 모드	파일에 읽고 쓸 수 있는 모드이다. 모드를 변경하려면 seek()가 호출되어야 한다.



"r"

파일의 처음 부터 읽는다.



"w"

파일의 처음 부터 쓴다.
만약 파일이 존재하면 기존의
내용이 지워진다.

"a"

파일의 끝에 쓴다.
파일이 없으면 생성 된다.

파일에서 읽기

input.txt

```
홍길동
김철수
```

```
infile = open("input.txt", "r", encoding="utf-8")                    p462.py
line = infile.readline()
while line != "" :
        print(line)
        line = infile.readline()
        #line = infile.readline().rstrip()      #줄바꿈 문자(\n)를 삭제하여 빈 줄 출력을 제거
```

```
홍길동

김철수
```

```
홍길동
김철수
```

파일에 쓰기

- 다양한 형식의 파일 쓰기 코드를 사용할 수 있다.

p463.py

```
outfile = open("output.txt", "w", encoding="utf-8")

outfile.write("김영희\n")
print("나하나", file=outfile)

age = 20
outfile.write(f"이쓸이 나이 ={age}\n")
outfile.write("한바읻 나이=%s\n" % age)

outfile.close()
```

output.txt

```
김영희
나하나
이쓸이 나이=20
한바읻 나이=20
```

파일 닫기

- 파일과 연결된 자원을 해제

```
f = open("test.txt", "w")       # 파일을 연다.

# 여기서 여러 가지 작업을 한다.
f.close()                       # 파일을 닫는다.
```

완전히 안전한 방법은 아니다.

```
try:                            # 예외가 발생할 가능성이 있는 작업들을 여기에 둔다.
   f = open("test.txt", "w")
   # 여기서 여러 가지 작업을 한다.
finally:                        # 예외가 발생하더라도 반드시 실행된다.
   f.close()
```

예외가 발생하더라도 닫는 것을 보장

```
with open("test.txt", "w") as f:
        f.write("김영희\n")
        f.write("최자영\n")
# 블록을 빠져나오면 자동으로 파일이 닫혀진다.
```

with블록이 종료될 때 자동으로 닫힌다

Lab: 매출 파일 처리

- 입력 파일에 상점의 일별 매출이 저장되어 있다고 하자. 이것을 읽어서 일별 평균 매출과 총 매출을 계산한 후에 다른 파일에 출력하는 프로그램을 작성해보자.

sales.txt

```
1000000
1000000
1000000
500000
1500000
```

summary.txt

```
총매출 = 5000000
평균 일매출 = 1000000.0
```

Lab: 매칠 파일 처리

p465.py

```
infilename = input("입력 파일 이름: ");
outfilename = input("칠력 파일 이름: ");

infile = open(infilename, "r")
outfile = open(outfilename, "w")

sum = 0
count = 0

line = infile.readline()
while line != "" :
        s = int(line)
        sum += s
        count += 1
        line = infile.readline()


…
…
```

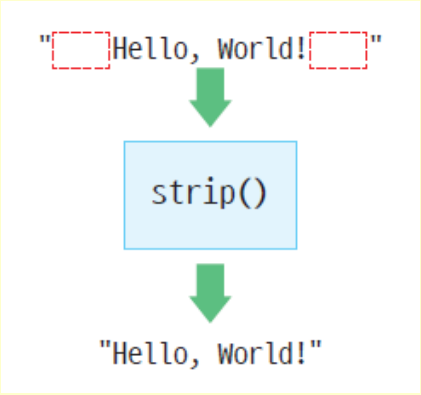
다양한 텍스트 입출력 방법

- for – each 구문을 이용하는 방법

```
infile = open("scores.txt", "r")
for line in infile :
        print(line)
```

- 줄바꿈 문자 제거

```
>>> s = " Hello, World!\n"
>>> s.strip()
"Hello, World!"
>>> s = "#########this is example#####"
>>> s.strip('#')
'this is example’
>>> s = "#########this is example#####"
>>> s.lstrip('#')
'this is example#####’
>>> s.rstrip('#')
 '#########this is example'
```

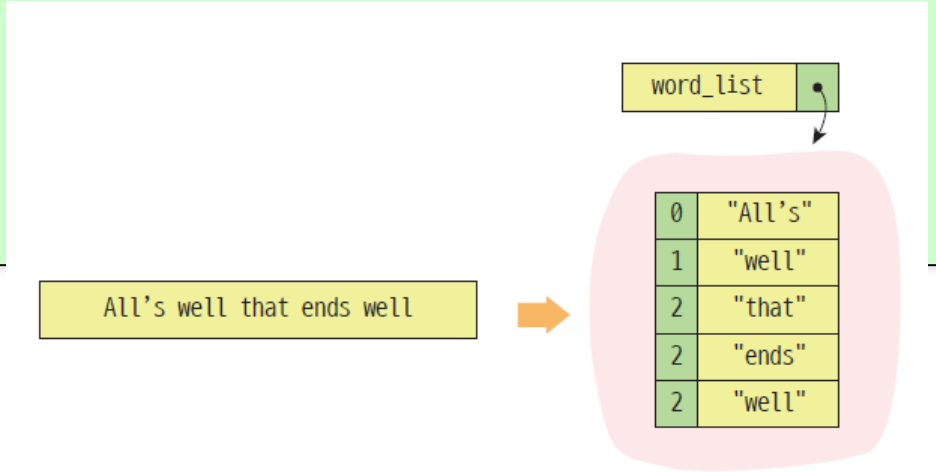


단어로 분리하기

p468_spilit.py

```
infile = open("proverbs.txt", "r")

for line in infile:
        line = line.rstrip()          # 오른쪽 공백 문자를 없앤다.
        word_list = line.split()      # 단어들로 분리한다.
        for word in word_list:        # 리스트에 들어 있는 단어들을 출력한다.
                print(word);
infile.close()
```

```
All's
well
...
flock
together.
```



파일 전체 읽기

1) read() 사용 - 파일의 모든 문자가 하나의 문자열로 반환. 메모리 많이 필요

p468_read.py

```
infile = open("input.txt", "r")
s = infile.read()
print(s)
infile.close()
```

2) readline() 사용 - 각 줄이 저장된 리스트를 반환

```
infile = open("input.txt", "r")
lines = infile.readlines()
for line in lines :
        print(line)
infile.close()
```

```
홍길동
김철수
```

문자 단위로 읽기

- read()를 이용하여 원하는 개수만큼의 글자를 읽어올 수 있다.

p469.py

```
infile = open("input.txt", "r")
ch = infile.read(1)
while ch != "" :
        print(ch)
        ch = infile.read(1)
infile.close()
```

```
홍
길
동
…
```

문자 단위로 읽기

- 예: read()를 이용하여 문자들의 출현 횟수를 계산하는 코드

p470.py

```
counter = [0] *26
infile = open("mobydick.txt", "r")
ch = infile.read(1)
while ch != "" :
    ch = ch.upper()              # 대문자->소문자
    if ch >= "A" and ch <= "Z" :
        i = ord(ch) - ord("A")
        counter[i] += 1
    ch = infile.read(1)
print(counter)
```

```
[79235, 17211, 23318, 38853, 119338, 21260, 21285, 63764, 66701, 1176,
8223, 43368, 23696, 66779, 70790, 17886, 1581, 53585, 65145, 89895,
27203, 8730, 22540, 1064, 17230, 638]
```

문자 인코딩

- 최근에는 세계의 모든 문자를 나타낼 수 있는 유니코드가 사용된다.
- 유니코드 중에서 가장 많이 사용되는 인코딩은 UTF-8이다. UTF-8에서는 각 문자를 1개에서 4개의 바이트로 인코딩한다.

- infile = open("input.txt", "r", encoding="utf-8")

Lab: 행맨

- 사용자는 한 번에 하나의 글자만을 입력할 수 있으며 맞으면 글자가 보이고 아니면 시도 횟수만 하나 증가한다.

p471.py

```
import random

guesses = ''
turns = 10
infile = open("words.txt", "r")
lines = infile.readlines()
word = random.choice(lines)
while turns > 0:
    failed = 0
    for char in word:
        if char in guesses:
            print(char, end="")
        else:
            print("_", end="")
            failed += 1
…
…
```

```
_____
단어를 추측하시오: a
틀렸음!
9기회가 남았음!

_____
단어를 추측하시오: e
틀렸음!
8기회가 남았음!
```

Lab: 각 문자 횟수 세기

- 파일 안의 각 문자들이 몇 번이나 나타나는지를 세는 프로그램

p472.py

```
filename = input("파일명을 입력하세요: ").strip()
infile = open(filename, "r") # 파일을 연다.

freqs = {}

# 파일의 각 줄에 대하여 문자를 추출한다. 각 문자를 사전에 추가한다.
for line in infile:
    for char in line.strip():                # 양쪽 끝의 공백 문자를 제거한다.
        if char in freqs:                    # 문자열 안의 각 문자에 대하여
            freqs[char] += 1                 # 딕셔너리의 횟수를 증가한다.
        else:                                            # 처음 나온 문자이면
            freqs[char] = 1                  # 딕셔너리의 횟수를 1로 초기화한다.

print(freqs)
infile.close()
```

```
{' ': 16, 'e': 12, 'o': 4, 'a': 7, 'u': 1, 'n': 4, 'k': 1, 'A': 1, 'r': 4, 'g': 2, 's': 7, 'b': 1, 'd': 4, 'v': 1, 'f': 5, 'w': 3, 'B': 2, 'h': 4, 'i': 2, 't': 7, 'l': 11, 'W': 1, '.': 4, '"': 1, 'c': 1}
```

Lab: CVS 파일 처리

- CSV : 테이블 형식의 데이터를 저장하고 이동하는 데 사용되는 구조화된 텍스트 파일 형식. Microsoft Excel와 같은 스프레드시트에 적합한 형식이다.
- 파이썬 모듈 csv는 CSV reader와 CSV writer를 제공한다.

p473.py

```
import csv

f = open('weather.csv')        # CSV 파일을 열어서 f에 저장한다.
data = csv.reader(f)
header = next(data)            # 헤더를 제거한다.
for row in data:               # 반복 루프를 사용하여 데이터를 읽는다.
    print(row)
f.close()
```

Lab: CVS 파일 처리

- 서울이 언제 가장 추웠는지를 조사해보자.

p474.py

```
import csv

f = open('d://weather.csv')                    # CSV 파일을 열어서 f에 저장한다.
data = csv.reader(f)
header = next(data)
temp = 1000
for row in data:
    if temp > float(row[3]):
        temp = float(row[3])
print('가장 추웠던 날은', temp, '입니다')
f.close()
```

```
가장 추웠던 날은 –19.2 입니다.
```

Lab: 파일 암호화

- 시저 암호를 구현하여 보자. 줄리어스 시저는 친지들에게 비밀리에 편지를 보내고자 할 때 다른 사람들이 알아보지 못하도록 문자들을 다른 문자들로 치환하였다.

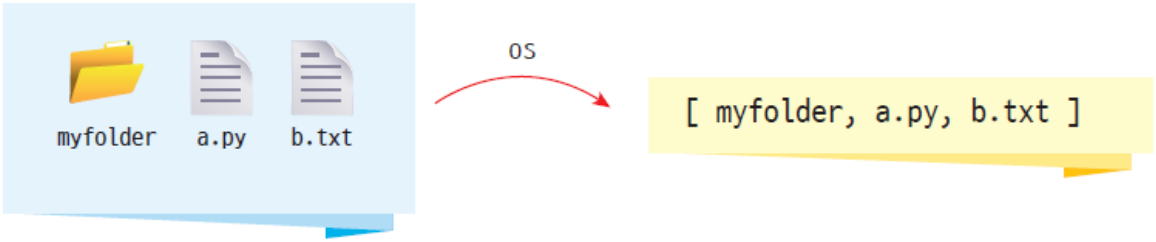
평문	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
암호문	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

원문: the language of truth is simple.
암호문: wkh odqjxdjh ri wuxwk lv vlpsoh.
복호문: the language of truth is simple.

p475.py

```
key = 'abcdefghijklmnopqrstuvwxyz'

def encrypt(n, plaintext):
    result = ''
    for l in plaintext.lower():
        try:
…
…
```

디렉토리 작업




```
>>> dir = os.getcwd()          # 작업 디렉토리를 얻기

>>> subdir = "data"
>>> os.chdir(subdir)           # 작업 디렉토리를 변경

>>> for filename in os.listdir() :          #파일들의 리스트를 얻기
        print(filename)

>>> if os.path.isfile(filename) :          # 파일만 처리하려면
        print("파일입니다.")
```

디렉토리 작업

- 파일 확장자 검사하려면 endswith()를 사용
- 예. 작업 디렉토리에서 확장자가 “.jpg ” 인 파일을 전부 찾아서 파일 이름을 출력

p477.py

```
import os

cwd = os.getcwd()
files = os.listdir()
for name in files :
        if os.path.isfile(name) :
                if name.endswith(".jpg") :
                        print(name)
```

```
DSC04886_11.jpg
DSC04886_12.jpg
DSC04886_13.jpg
```

Lab: 디렉토리 안의 파일 처리

- 예. 디렉토리 안의 모든 파일 중에서 ”Python “ 을 포함하고 있는 줄이 있으면 파일의 이름과 해당 줄을 출력한다.

p478.py

```
import os
arr = os.listdir()

for f in arr:
    infile = open(f, "r", encoding="utf-8")
    for line in infile:
        e = line.rstrip()                    # 오른쪽 줄바꿈 문자를 없앤다.
        if "Python" in e:
            print(f, ":", e)
    infile.close()
```

```
file.py :        if "Python" in e:
summary.txt : The joy of coding Python should be in seeing short
summary.txt : Python is executable pseudocode.
```

이진 파일

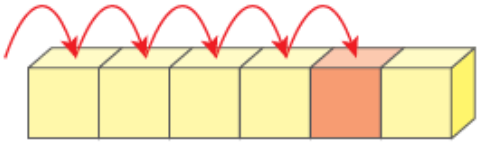
- 데이터가 직접 저장되어 있는 파일. 이진수 형태로 저장.
- 장점 – 효율성. 적은 기억공간 차지
- 단점 – 읽을 수 없다.

이진 파일

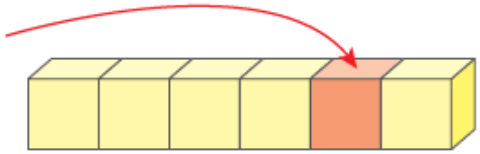
```
이진 파일에서 데이터를 읽기 목적으로 열기
>>> infile = open(filename, "rb")

입력 파일에서 8 바이트 읽기
>>> bytesArray = infile.read(8)

첫 번째 바이트를 꺼내기
>>> byte1 = bytesArray[0]

이진 파일에 바이트들을 저장
>>> outfile = open(filename, "wb")
>>> bytesArray = bytes([255, 128, 0, 1])
>>> outfile.write(bytesArray)
```

순차접근과 임의접근

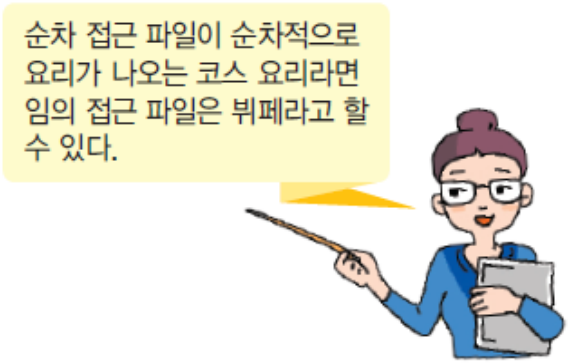
- 순차 접근(sequential access) : 파일의 처음부터 순차적으로 읽거나 기록
- 임의 접근(random access) : 파일의 어느 위치에서든지 읽기와 쓰기가 가능



순차 접근 파일

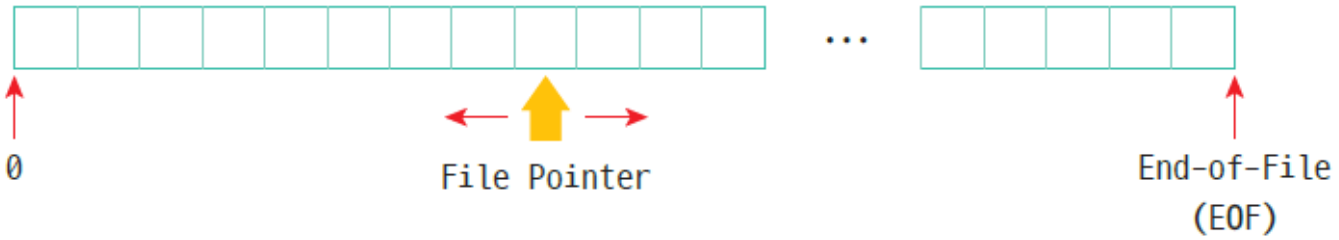


임의 접근 파일



임의접근의 원리

- 파일 포인터는 읽기와 쓰기 동작이 현재 어떤 위치에서 이루어지는 지를 나타낸다. 새 파일이 만들어 지면 파일 포인트는 값이 0이고 이것은 파일의 시작 부분을 가르킨다.
- 보통 순차적으로 데이터를 읽게 되면 파일 포인트는 파일의 시작위치에서 순차적으로 증가하여 파일의 끝으로 이동한다. 데이터를 임의의 위치에서 읽는 기능이 필요한 경우에는 위치 표시자를 조작하여 원하는 위에서 읽을 수 있다.



- seek() : 위치 표시자 바꾸기
- tell() : 현재의 위치 알기

임의접근의 원리

- 예. 텍스트 파일에서 몇 개의 문자를 읽은 후에 seek()를 이용하여 다시 파일의 처음으로 돌아가기.

p481.py

```
infile = open("test.txt", "r+")
str = infile.read(10);
print("읽은 문자열 : ", str)
position = infile.tell();
print("현재 위치:  ", position)

position = infile.seek(0);
str = infile.read(10);
print("읽은 문자열 : ", str)
infile.close()
```

```
읽은 문자열 :  abcdefghij
현재 위치:   10
읽은 문자열 :  abcdefghij
```

Lab: 이미지 파일 복사하기

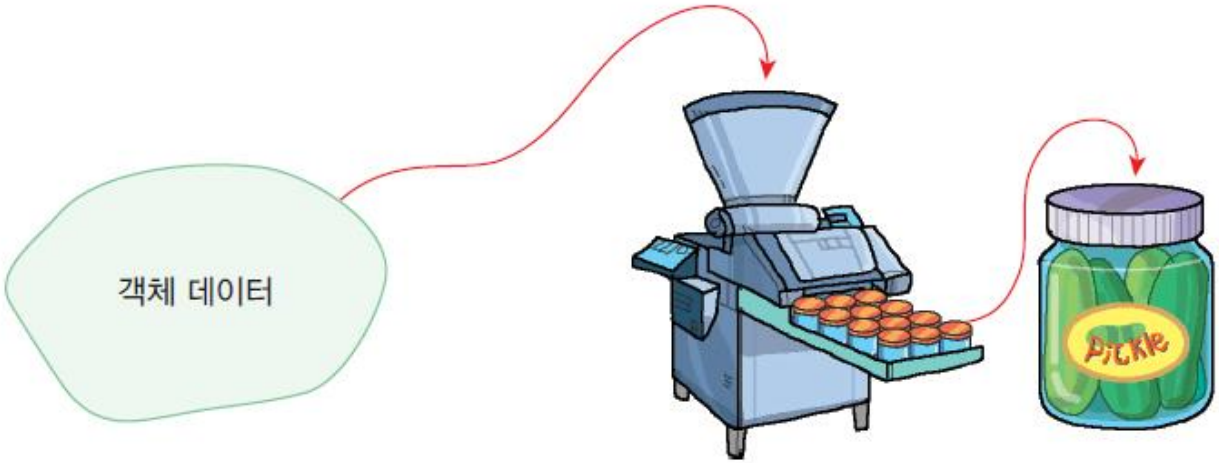
- 하나의 이미지 파일을 다른 이미지 파일로 복사하는 프로그램

p482.py

```
infile = open("123.png", "rb")
outfile = open("kkk.png", "wb")

# 입력 파일에서 1024 바이트씩 읽어서 출력 파일에 쓴다.
while True:
    copy_buffer = infile.read(1024)
    if not copy_buffer:
        break
    outfile.write(copy_buffer)

infile.close()
outfile.close()
print(filename1+"를 " +filename2+"로
```

객체 출력

- 딕셔너리, 리스트와 같은 객체들을 이진 파일로 저장하거나 읽을 수 있는 방법
- 프로그램 내에서 빠르게 동작
- pickle 모듈의 dump()와 load() 메소드 사용



객체 출력

p484.py

- 파일로 저장

```
import pickle

gameOption = {
                                        "Sound": 8,
                                        "VideoQuality": "HIGH",
                                        "Money": 100000,
                                        "WeaponList": ["gun", "missile", "knife" ]
}
file = open( "d:\\save.p", "wb" )# 이진 파일 오픈
pickle.dump( gameOption, file )          # 딕셔너리를 피클 파일에 저장
file.close()                                                    # 파일을 닫는다.
```

- 파일 읽기

```
import pickle

file = open( "d:\\save.p", "rb" )                                # 이진 파일 오픈
obj = pickle.load( open( "save.p", "rb" ) )                      # 피클 파일에 딕셔너리를 로딩
print(obj)
```

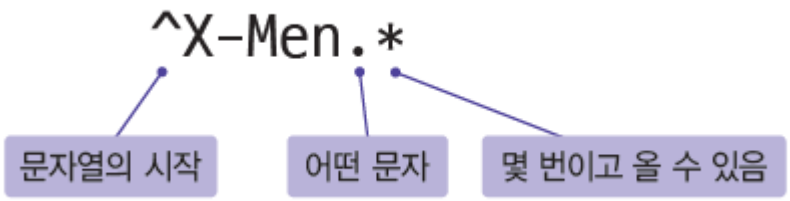
정규식

- 정규식(regular expression) : 특정한 규칙을 가지고 있는 문자열들을 메타 문자를 이용하여 표현하는 수식

식	기능	설명
^	시작	문자열의 시작을 표시
$	끝	문자열의 끝을 표시
.	문자	한 개의 문자와 일치
\d	숫자	한 개의 숫자와 일치
\w	문자와 숫자	한 개의 문자나 숫자와 일치
\s	공백문자	공백, 탭, 줄바꿈, 캐리지리턴 문자와 일치
\S	공백문자제외	공백 문자를 제외한 모든 문자
*	반복	0번 이상 반복
+	반복	1번 이상 반복
[abc]	문자 범위	[abc]는 a 또는 b 또는 c를 나타낸다.
[^abc]	문자 범위	[^abc]는 a,b,c가 아닌 어떤 문자

정규식

- 예. X-Men으로 시작하여야 되고 그 이후에는 어떤 문자가 반복되어도 된다.



"X-Men: First Class", "X-Men: Days of Future Past", "X-Men Origins: Wolverine"

- re 모듈이 필요

re.search() : 정규식에 매치되는 문자열을 찾는다.

re.findall() : 정규식을 만족하는 모든 문자열을 추출한다.

정규식

- 예. 미국 헌법에서 숫자로 시작되는 줄만을 출력하는 프로그램

p486.py

```
import re
f = open('uscons.txt')
for line in f:
        line = line.rstrip()
        if re.search('^[0-9]+', line) :
                print(line)
```

[0-9]는 0에서 9까지의 문자 중에서 하나를 의미하고 +는 1회 이상 반복을 의미

```
10th Amendment
11th Amendment
12th Amendment
13th Amendment
…
…
```

Lab: 정규식 이용하기

- "[수강 번호][수강 코드][과목 이름]" 형식으로 되어 있는 텍스트에서 수강 번호만을 추출해보자.

```
101 COM   PythonProgramming
102 MAT   LinearAlgebra
103 ENG   ComputerEnglish
```

```
['101', '102', '103']
```

p487.py

```
text="""101 COM   PythonProgramming
102 MAT   LinearAlgebra
103 ENG   ComputerEnglish"""

import re
s = re.findall("\d+", text)
print(s)
```

Lab: 패스워드 검사 프로그램

- 사용자가 입력한 패스워드를 검증하는 프로그램을 작성해보자.
  - 최소 8글자
  - 적어도 하나의 대문자
  - 적어도 하나의 숫자
  - 적어도 하나의 특수문자[_, @, $]

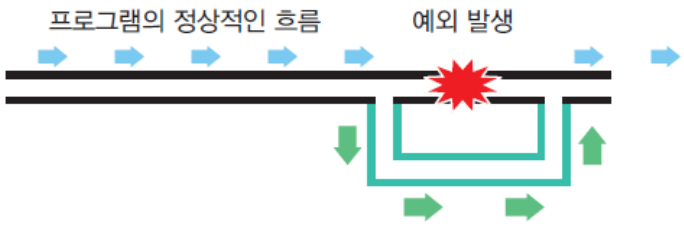
p488.py

```
import re

password = input("패스워드를 입력하세요");
flag = 0
while True:
    if (len(password)<8):
        flag = -1
        break
    elif not re.search("[a-z]", password):
        …
        …
```

예외처리

```
>>> (x, y)=(2, 0)
>>> z=x/y
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    z=x/y
ZeroDivisionError: division by zero
>>>
```

- 예외(exception) : 프로그램 실행 도중에 발생하는 오류
- 예외처리 : 오류를 처리한 후에 계속 실행할 수 있도록 처리하는 기능



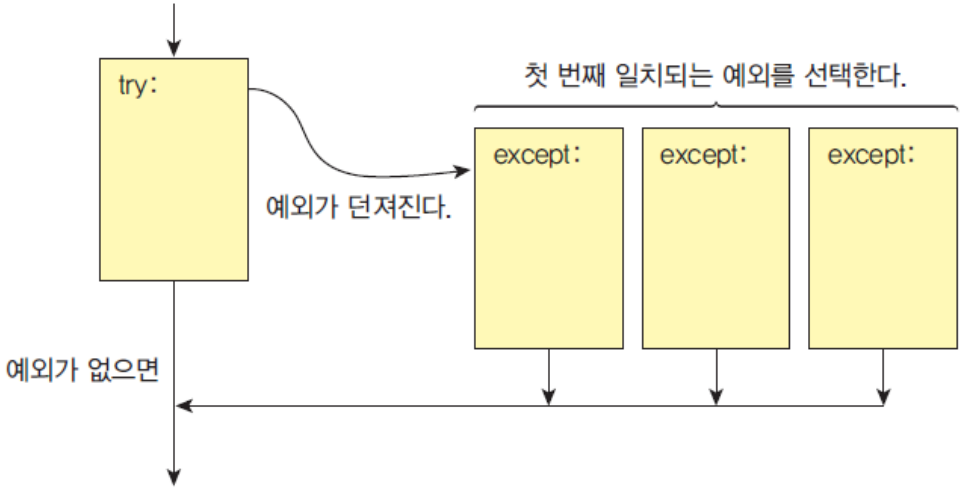
예외 처리는 프로그램의 실행을
계속할 수 있는 다른 경로를 제공한다.

오류의 종류

- 사용자 입력 오류: 사용자가 정수를 입력하여야 하는데 실수를 입력할 수 있다.
- 장치 오류: 네트워크가 안 된다거나 하드 디스크 작동이 실패할 수 있다.
- 코드 오류: 잘못된 인덱스를 사용하여서 배열에 접근할 수 있다.

  - IOError: 파일을 열 수 없으면 발생한다.
  - importError: 파이썬이 모듈을 찾을 수 없으면 발생한다.
  - ValueError: 연산이나 내장 함수에서 인수가 적절치 않은 값을 가지고 으면 발생한다.
  - KeyboardInterrupt: 사용자가 인터럽트 키를 누르면 발생한다. (Control-C나 Delete)
  - EOFError: 내장 함수가 파일의 끝을 만나면 발생한다.

try-except 블록



Syntax: 예외 처리

형식

```
try:
    예외가 발생할 수 있는 문장
except(오류):
    예외를 처리하는 문장
```

예

```
try:
    z = x/y
except ZeroDivisionError:
    print ("0으로 나누는 예외")
```

예외가 발생할 수 있는 문장

예외

try-except 블록

p491.py

```
(x,y) = (2,0)
try:
    z = x/y
except ZeroDivisionError:
```

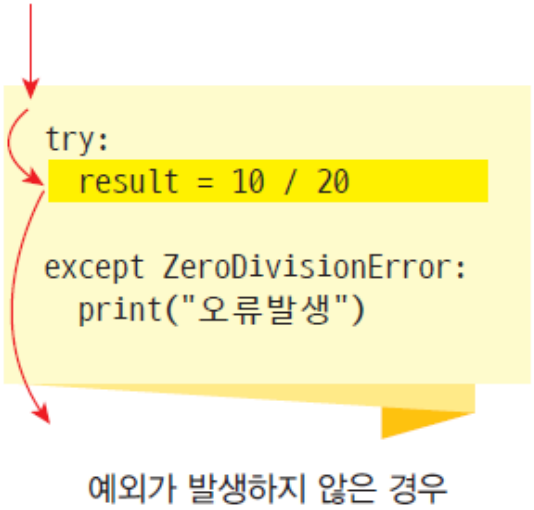
```
(x,y) = (2,0)
try:
    z = x/y
```

```
while True:
    try:
        n = input("정수를 입력하시오 :  ")
        n = int(n)
        break
    except ValueError:
prin
```

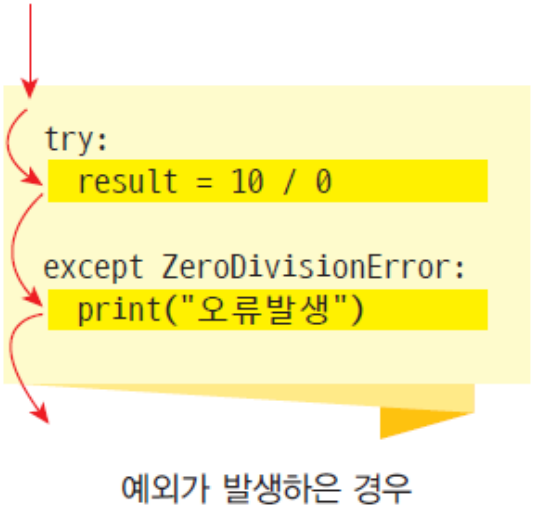
```
try:
    fname = input("파일 이름을 입력하세요: ")
    infile = open(fname, "r")
except IOError:
    print("파일 " + fname + "을 발견할 수 없습니다.")
```

try/execpt 블록에서의 실행 흐름

- 예외가 발생하지 않는 경우에는 except 블록의 코드는 실행되지 않는다. 예외가 발생한 경우에만 except 블록의 코드가 실행된다.



예외가 발생하지 않은 경우



예외가 발생하은 경우

다중 예외 처리 구조

- 하나의 문장에서 많은 예외가 발생할 수 있는 경우

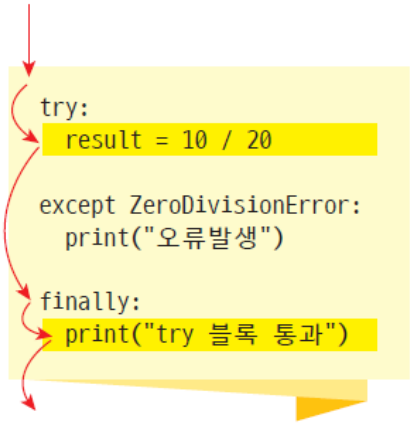
p493.py

```
try:
   fh = open("testfile", "w")
   fh.write("테스트 데이터를 파일에 씁니다!!")
except IOError:
   print("Error: 파일을 찾을 수 없거나 데이터를 쓸 수 없습니다. ")
else:
   print("파일에 성공적으로 기록하였습니다. ")
   fh.close()
```

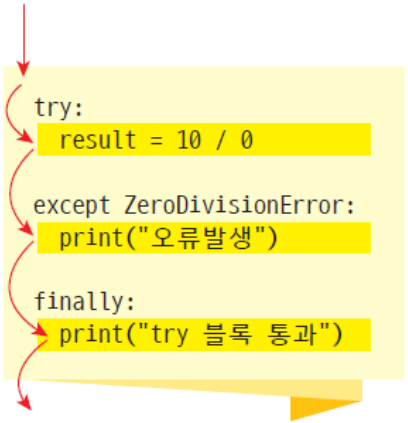
```
파일에 성공적으로 기록하였습니다.
```

try-finally 블록

- finally : 오류가 발생한 경우나 아니면 발생하지 않은 경우에도 항상 실행되어야 하는 문장을 두는 블록



예외가 발생하지 않은 경우



예외가 발생하은 경우

p495.py

```
try:
   f = open("test.txt", "w" )
   f.write("테스트 데이터를 파일에 씁니다!!")
   ...                                          # 파일 연산을 수행한다.
except IOError:
   print("Error: 파일을 찾을 수 없거나 데이터를 쓸 수 없습니다. ")
finally:
   f.close()
```

예외 발생하기

- 파이썬에서는 오류가 감지되면 raise 문을 사용하여 예외를 생성한다.

  raise exception

  exception : 예외의 종류. 문자열이나 객체로 정의할 수 있다.

```
>>> raise NameError('Hello')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: Hello
```

이번 장에서 배운 것

- 파일을 읽을 때는 파일을 열고, 데이터를 읽은 후에, 파일을 닫는 절차가 필요하다.
- 파일 모드에서 “r”, “w”, “a”가 있다. 각각 읽기모드, 쓰기모드, 추가모드를 의미한다.
- 파일은 텍스트 파일과 이진 파일로 나누어진다.
- 파일에서 데이터를 읽거나 쓰는 함수는 read()와 write() 함수이다. 텍스트 파일에서 한 줄을 읽으려면 for 루프를 사용한다.
- 예외 처리는 오류가 발생했을 때 프로그램을 우아하기 종료하는 방법이다. try 블록과 except 블록으로 이루어진다.