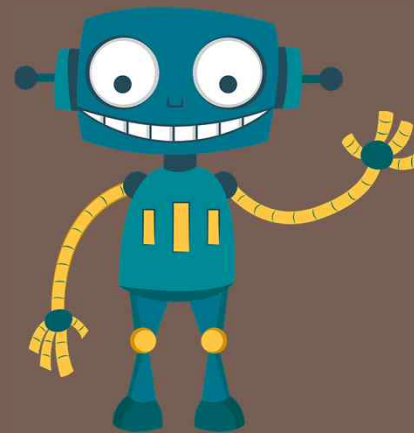


파이썬 익스프레스



10장 파일과 예외처리



Q & A

- 언제라도 질문하세요
 1. 강의시간의 채팅창 (수신인: 모두, 수강생모두에게 답변하기에)
 2. 네이버카페 질의응답게시판

- 강사의 1번 선생님은 여러분의 질문



아으로 나는 할 수 있다

1. 나는 텍스트 파일 읽고 쓰기를 살펴본다.
2. 나는 이진 파일 읽고 쓰기를 살펴본다.
3. 나는 정규식을 사용하는 방법을 살펴본다.
4. 나는 **CSV** 파일 읽고 쓰기를 살펴본다.
5. 나는 예외를 처리하는 방법을 살펴본다.



파일의 기초

- 프로그램에서 만든 데이터를 영구히 저장하고자 한다면 하드 디스크에 파일 형태로 저장하여야 한다.



메모리

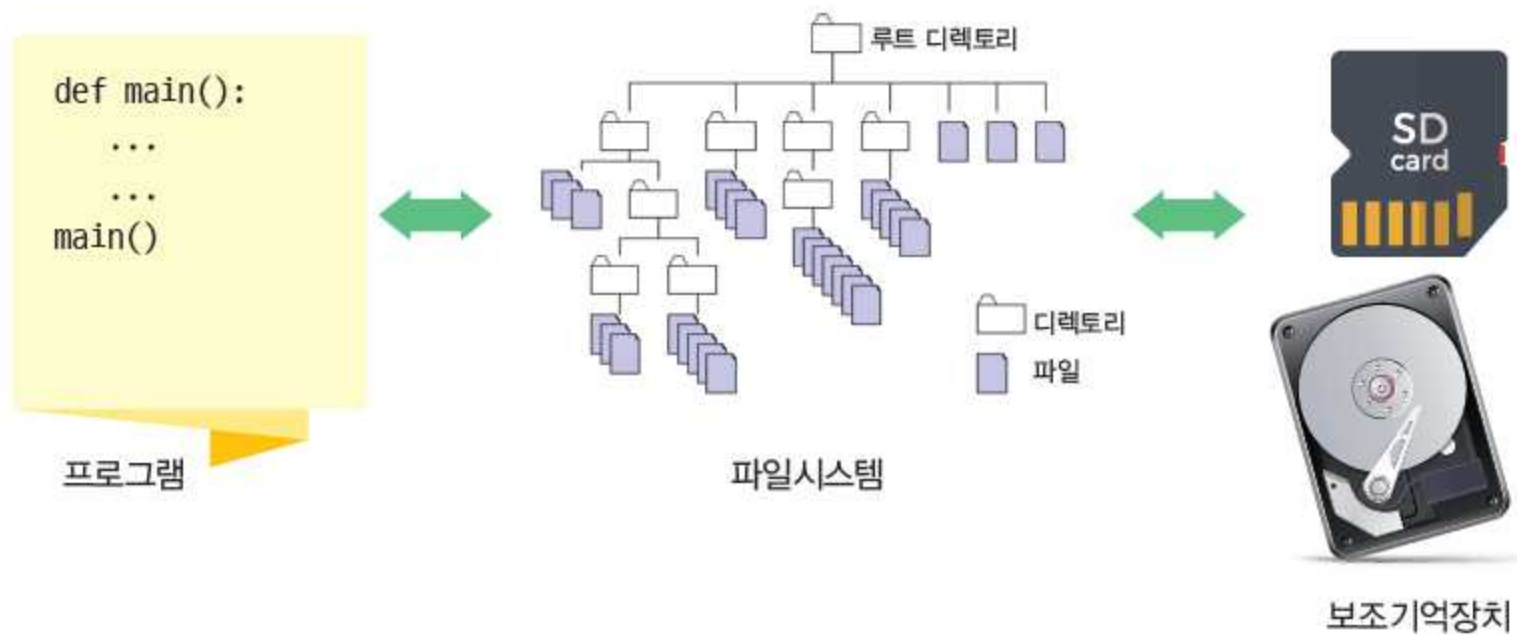
VS



SSD, 하드 디스크

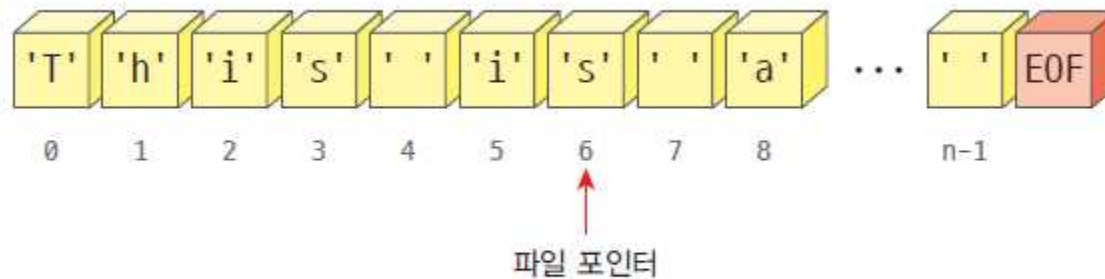
파일의 개념

- 파일은 보조기억장치 상에서 논리적인 정보 단위이다.



파일의 논리적인 구조

- 파일 안에는 바이트들이 순차적으로 저장되어 있고 맨 끝에는 EOF(end-of-file) 마커가 있다.



파일의 논리적인 구성

파일 열고 닫기

Syntax: 함수 정의

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

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

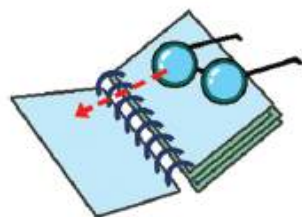
파일 객체

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 mode)	파일의 처음부터 쓴다. 파일이 없으면 생성된다. 만약 파일이 존재하면 기존의 내용은 지워진다.
"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 (readline(p462).py)

input.txt

호기도
오르도
기침스
미르

```
infile = open("input.txt", "r")  
line = infile.readline()  
while line != "" :  
    print(line)  
    line = infile.readline()
```

호기도
오르도

기침스
미르

파일에서 읽기 (readline(p462).py)

학생> 2가지 문제가 있습니다

1. 출력에서 빈줄이 생김
2. Spyder 편집창에서 input.txt 한글이 깨짐

교수> 1번 문제는 다다음 slide에서 해결

2번 문제는 메모장에서는 input.txt 한글이 깨지지 않았고 콘솔창에서도 한글이 깨지지 않음

문자 인코딩 변경하여 다음 slide에서 해결

파일에서 읽기 (readline_change.py)

utf-8 , utf_8, utf8 모두 동일

encoding 매개변수가 없으면 운영체제에서 가져옴

Input_utf8.txt

호기
도
를
기
칠
수
없
다

```
infile = open("input_utf8.txt", "r", encoding="utf-8")
line = infile.readline()
while line != "" :
    print(line)
    line = infile.readline()
```

파일에서 읽기

Line변수에 줄바꿈 문자인 '\n'이 저장되어 빈 줄이 출력됨

input.txt

호기도
○ 리
기척스
□ 리

```
infile = open("input.txt", "r")  
line = infile.readline()  
while line != "" :  
    print(line)  
    line = infile.readline().rstrip()
```

호기도
○ 리
기척스
□ 리

파일에 쓰기

```
outfile = open("output.txt", "w")  
outfile.write("김영희\n")
```

output.txt

김영희

파일 닫기

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

```
# 여기서 여러 가지 작업을 한다.
```

```
f.close()                    # 파일을 닫는다.
```

```
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()
```

```
with open("test.txt", "w") as f:
```

```
    f.write("김영희\n")
```

```
    f.write("최자영\n")
```

```
# 블록을 빠져나오며 자동으로 파일이 닫혀진다.
```

Lab: 매출 파일 처리 (sales.py)

- 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
```

5.1 매치 파일 처리

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

outfile.write("총매출 = "+ str(sum)+"\n")
outfile.write("평균 일매출 = "+ str(sum/count) )

infile.close()
outfile.close()
```

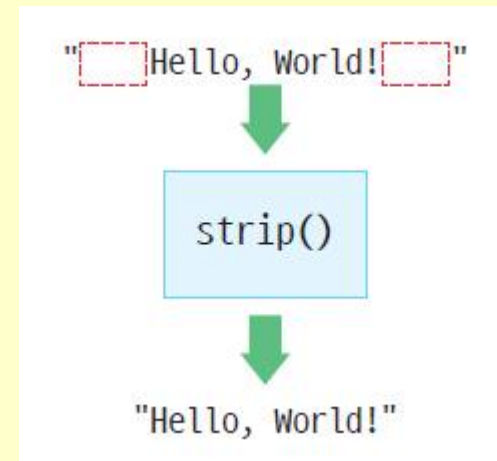

다양한 텍스트 입력 방법

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

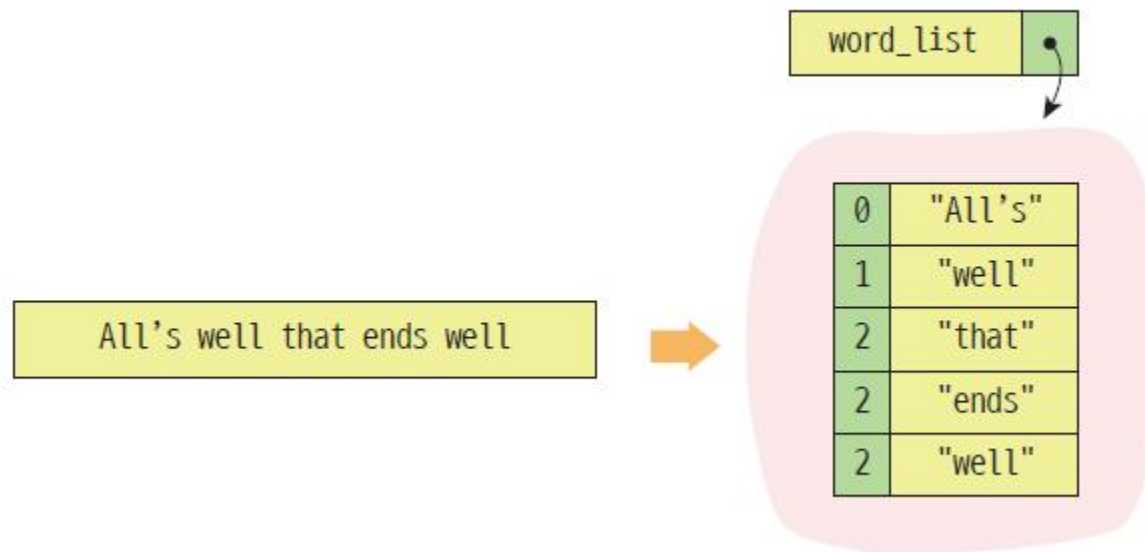
```
>>> s = " Hello, World!\n"
>>> s.strip()
"Hello, World!"
```

```
>>> 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



단어로 분리하기 (split(p468).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 단어들 출력한다.
        print(word)

infile.close()
```

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

파일 전체 읽기 (readlines(p469).py)

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

호기도
○ 20
기척[△]
□ 2⁺

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

문자 단위로 읽기 (read1(p469).py)

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

```
○
○
○
○
○
...
```

문자 출현 횟수 계산 (char_count(p470).py)

Mobydick.txt에서 영어 26개 문자 출현 횟수 계산
e 문자가 가장 많음

```
counter = [0] * 26 // list 자료형
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-8**이다. **UTF-8**에서는 각 문자를 1개에서 4개의 바이트로 인코딩한다.
- `infile = open("input.txt", "r", encoding="utf-8")`

Lab: 행맨 (hangman.py)

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

단어를 추측하시오: a

틀림!

9 기회가 남아있음!

단어를 추측하시오: e

틀림!

8 기회가 남아있음!

Sol: 행맨

```
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
```

```
    if failed == 0:
```

```
        print("사양자 승리")
```

```
        break
```

Sol: 행맨

```
print("")
guess = input("단어를 추측하시오: ")
guesses += guess
if guess not in word:
    turns -= 1
    print ("틀렸습니다!")
    print (str(turns)+ " 기회가 남아있음!")
    if turns == 0:
        print("사용자 패배 정답은 "+word)
```

```
infile.close()
```

Lab: 각 문자 횟수 세기 (count_letters.py)

- 파일 안의 각 문자들이 몇 번이나 나타나는지를 세는 프로그램을 작성하자.

```
{ ' ': 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 }
```

Sol:

- Spyder의 variable explore창에서 freqs의 자료형(data type) 확인하세요

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

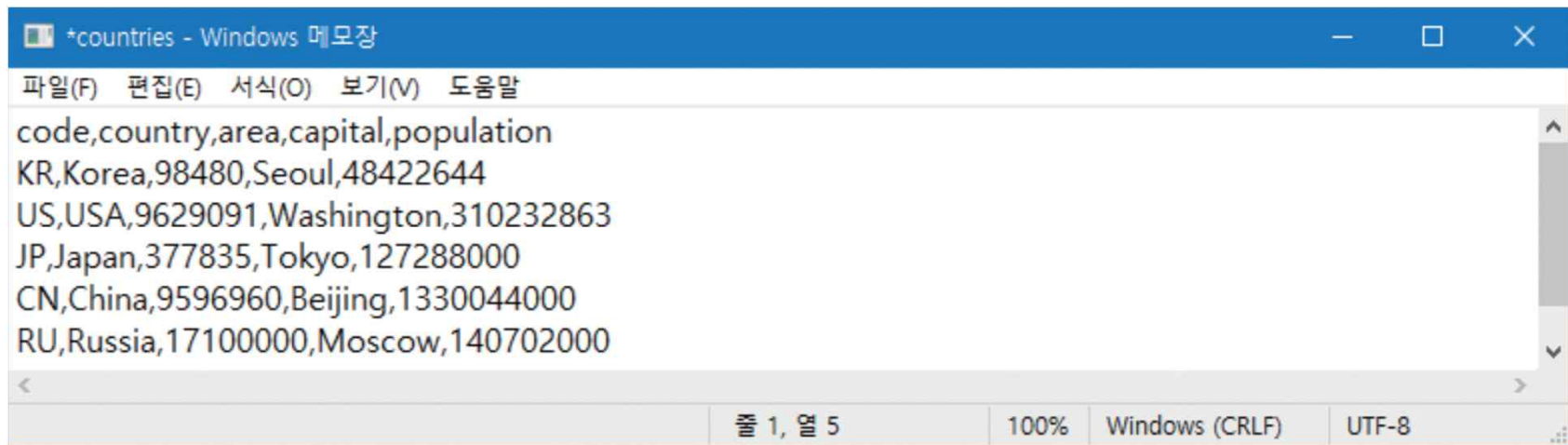
freqs = {} # 딕셔너리 자료형

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

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

Lab: CSV 파일 처리 (csv.py)

- CSV는 테이블 형식의 데이터를 저장하고 이동하는 데 사용되는 구조화된 텍스트 파일 형식이다. CSV는 Microsoft Excel과 같은 스프레드시트에 적합한 형식이다.



```
*countries - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말
code,country,area,capital,population
KR,Korea,98480,Seoul,48422644
US,USA,9629091,Washington,310232863
JP,Japan,377835,Tokyo,127288000
CN,China,9596960,Beijing,1330044000
RU,Russia,17100000,Moscow,140702000
<
줄 1, 열 5    100%    Windows (CRLF)    UTF-8
```

Lab: CVS 파일 처리

- 날씨 정보를 읽어서 서울이 언제 가장 추웠는지를 조사해보자.

weather.csv

날짜, 지점, 평균기온(°C), 최저기온(°C), 최고기온(°C)

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

1980-04-05,108,15.4,12.5,18.2

1980-04-06,108,7.1,4.3,12.5

1980-04-07,108,8.5,4.7,13.3

1980-04-08,108,10.8,8.4,15.2

...

Sol:

- Variable explorer창에서 header 변수, row 변수, temp변수 보기

```
import csv

f = open('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 (cipher.py)

- 시저 암호를 구현하여 보자.

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.

Sol:

```
key = 'abcdefghijklmnopqrstuvwxyz'
```

```
# 평문을 받아서 암호화하고 암호문을 반환한다.
```

```
def encrypt(n, plaintext):
```

```
    result = ""
```

```
    for l in plaintext.lower():
```

```
        try:
```

```
            i = (key.index(l) + n) % 26
```

```
            result += key[i]
```

```
        except ValueError:
```

```
            result += l
```

```
    return result.lower()
```

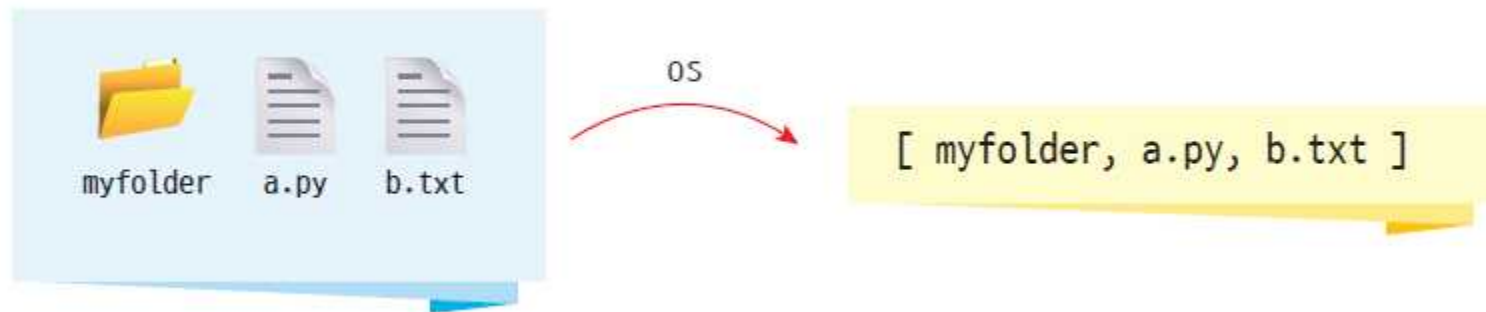
Sol:

암호문을 받아서 복호화하고 평문을 반환한다.

```
def decrypt(n, ciphertext):  
    result = "  
  
    for l in ciphertext:  
        try:  
            i = (key.index(l) - n) % 26  
            result += key[i]  
        except ValueError:  
            result += l  
  
    return result
```

```
n = 3  
text = 'The language of truth is simple.'  
encrypted = encrypt(n, text)  
decrypted = decrypt(n, encrypted)  
print ('평문: ', text)  
print ('암호문: ', encrypted)  
print ('복호문: ', decrypted)
```

디렉토리 작업



디렉토리 작업

```
>>> import os
```

작업 디렉토리를 얻으려면 다음과 같은 함수 호출을 사용한다.

```
>>> dir = os.getcwd()
```

작업 디렉토리를 변경할 수 있다.

```
>>> subdir = "data"
```

```
>>> os.chdir(subdir)
```

작업 디렉토리 안에 있는 파일들의 리스트를 얻으려면 `listdir()` 함수를 사용한다.

```
>>> for filename in os.listdir() :  
    print(filename)
```

파일만 처리하려면 다음과 같이 `isfile()` 함수를 사용한다.

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

작업 디렉토리에서 확장자가 ".jpg"인 파일을 전부
찾아서 파일 이름을 출력하는 프로그램

```
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

(listdir2(p478).py)

- 파일 중에서 "Python"을 포함하고 있는 줄이 있으면 파일의 이름과 해당 줄을 출력한다.

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

Sol:

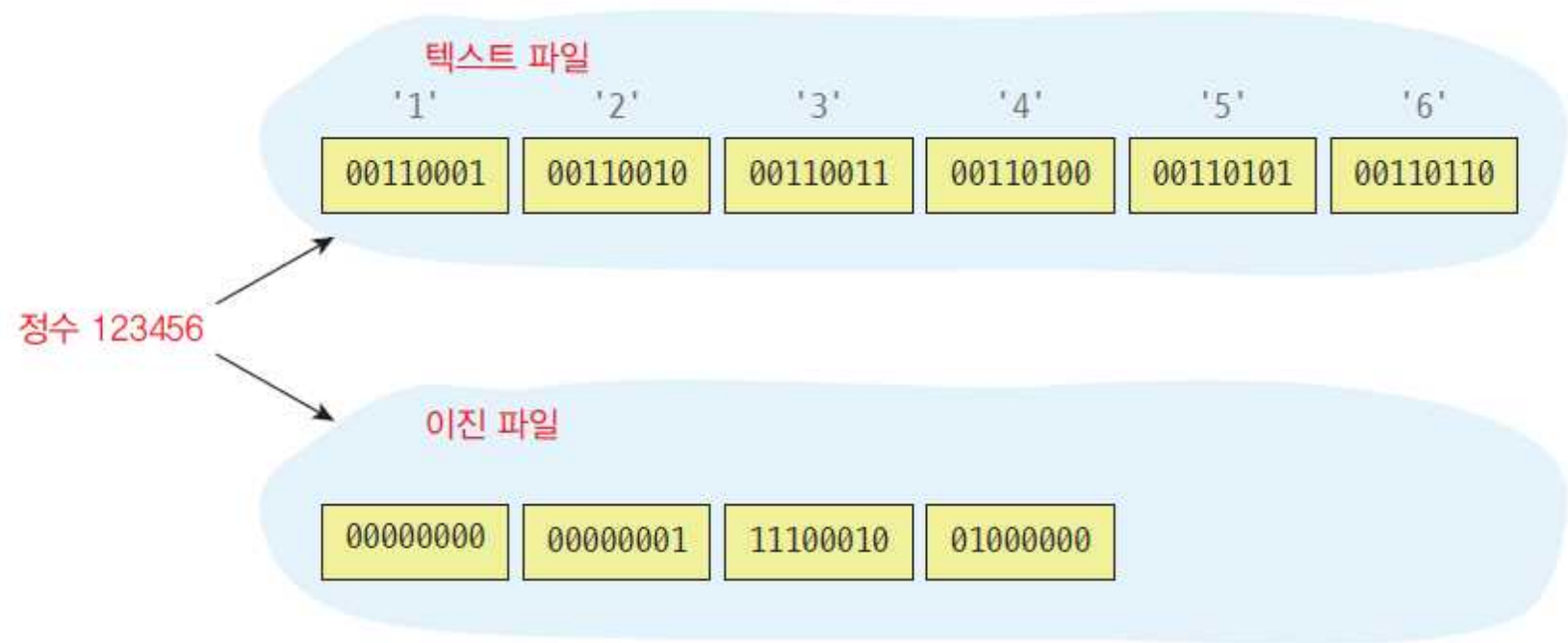
학생> error나는데요

교수> 변수창에서 123.png 파일에서 error나므로 2slide앞처럼 대상파일들을 .py와 .txt로 변경

```
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()
```

이진 파일



이진 파일에서 읽기 (예제는 4slide 다음에)

이진 파일에서 데이터를 읽으려면 다음과 같이 파일을 열어야 한다.

```
>>> infile = open(filename, "rb")
```

입력 파일에서 8 바이트를 읽으려면 다음과 같은 문장을 사용한다.

```
>>> byteArray = infile.read(8)
```

첫 번째 바이트를 꺼내려면 다음과 같은 문장을 사용하면 된다.

```
>>> byte1 = byteArray[0]
```

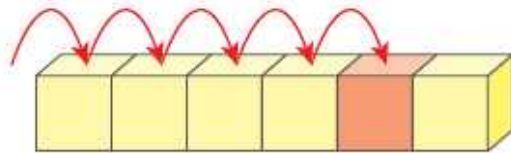
이진 파일에 바이트들을 저장하려면 다음과 같이 한다.

```
>>> outfile = open(filename, "wb")
```

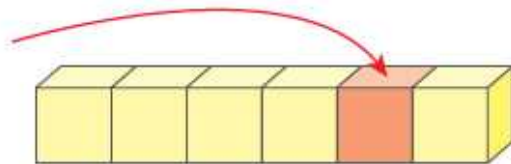
```
>>> byteArray = bytes([255, 128, 0, 1])
```

```
>>> outfile.write(byteArray)
```

순차 접근과 임의 접근



순차 접근 파일



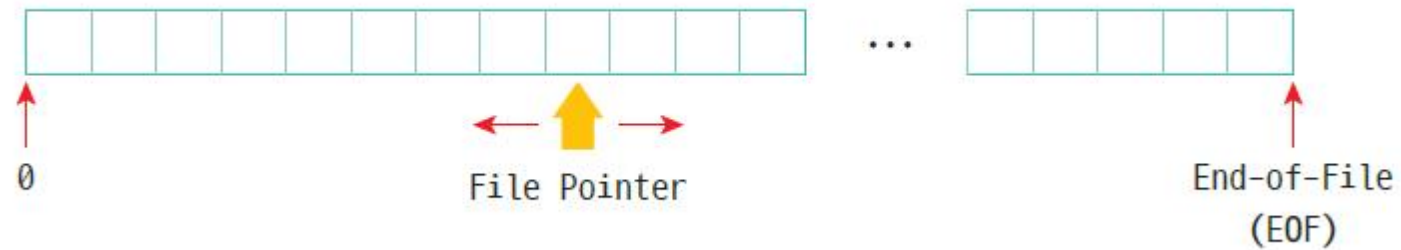
임의 접근 파일

순차 접근 파일이 순차적으로
요리가 나오는 코스 요리라면
임의 접근 파일은 뷔페라고 할
수 있다.



임의 접근의 원리

- 파일 포인터는 읽기와 쓰기 동작이 현재 어떤 위치에서 이루어지는지를 나타낸다.



예제 (seek.py)

- 텍스트 파일에서 몇 개의 문자를 읽은 후에 **seek()**를 이용하여 다시 파일의 처음으로 돌아가 보자.

```
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

(binary_filecopy.py)

- 하나의 이미지 파일을 다른 이미지 파일로 복사하는 프로그램을 작성하여 보자.



Sol:

학생> 교수님~ 에러나요.

```
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 복사하였습니다. ")
```

Sol: (binary_filecopy_change.py)

- Error 메시지보고 변경했습니다.
- >>> print(infile)
결과 보고 infile.name 사용

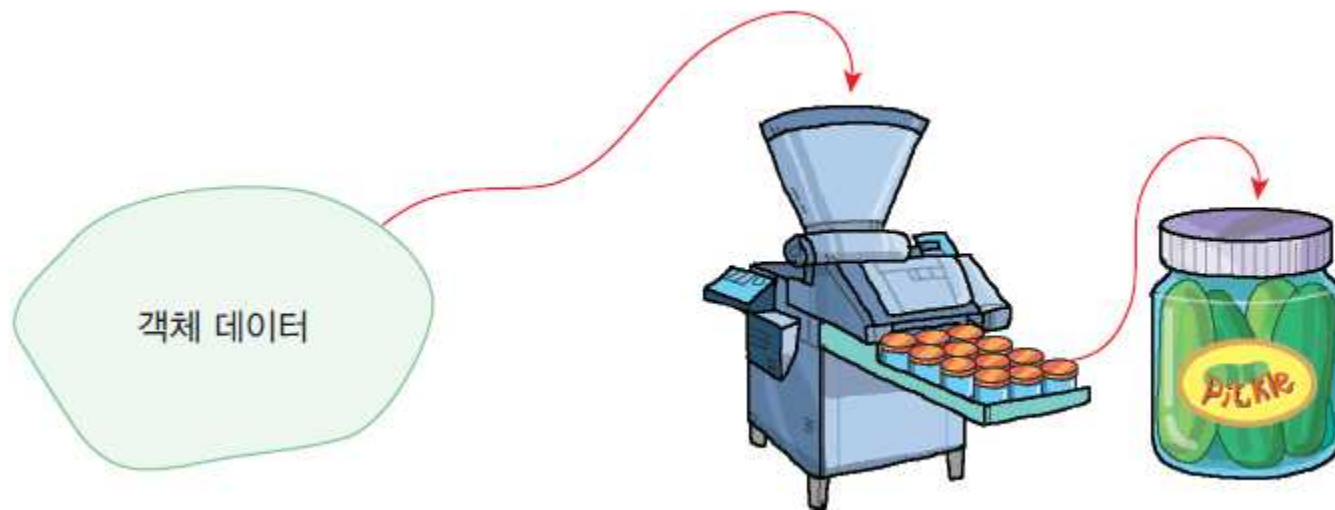
```
infile = open("123.png", "rb")
outfile = open("kkk.png", "wb")

# 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(infile.name+"를 " +outfile.name+"로 복사하였습니다. ")
```

객체 입출력

- 이제까지 문자열 데이터와 이진데이터 입출력해봄
- **pickle** 모듈의 **dump()**와 **load()** 메소드를 사용하면 객체(예: 딕셔너리나 리스트)를 쓰고 읽을 수 있다.



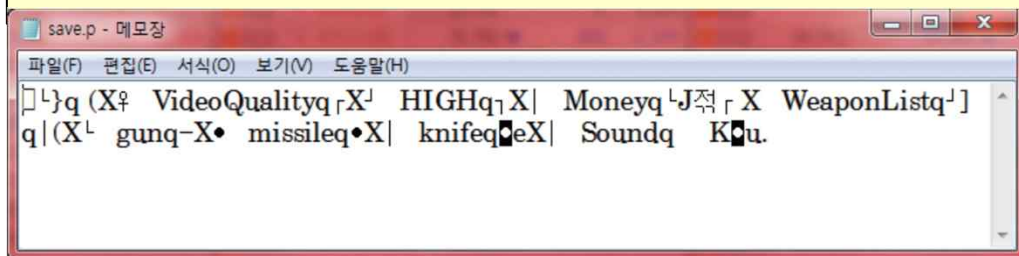
예제 (pickle.py)

학생> save.p가 안 열려요.

```
import pickle

gameOption = {
    "Sound": 8,
    "VideoQuality": "HIGH",
    "Money": 100000,
    "WeaponList": ["gun", "missile", "knife" ]
}

file = open( "save.p", "wb" ) # 이진 파일 오픈
pickle.dump( gameOption, file ) # 딕셔너리를 피클 파일에 저장
file.close() # 파일을 닫는다.
```



예제 (pickle2.py)

학생> 객체를 파일에 입출력하는 기능이 신기하네요

교수> 자바에서는 이러한 기능을 **serialization**이라고 합니다

```
import pickle
```

```
obj = pickle.load( open( "save.p", "rb" ) )    # 피클 파일에 디렉너리를 로딩  
print(obj)
```

```
In [17]: runfile('C:/idec/sources/chap10/pickle2(p484).py',  
wdir='C:/idec/sources/chap10')  
{'Sound': 8, 'VideoQuality': 'HIGH', 'Money': 100000,  
'WeaponList': ['gun', 'missile', 'knife']}
```

정규식

- 정규식(**regular expression**)이란 특정한 규칙을 가지고 있는 문자열들을 메타 문자를 이용하여 표현하는 수식이다.

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

정규식에서 점과 별표의 의미



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

예제(re(p486).py)

- 미국 헌법에서 숫자로 시작되는 줄 만을 출력하는 프로그램은 다음과 같다.

학생> regular expression 사용하지 않고 어떻게 이런 프로그램 만들 수 있나요?

교수> 만들어 보고 있습니다 ^^

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

```
In [19]: runfile('
wdir='C:/idec/sour
10th Amendment
11th Amendment
12th Amendment
13th Amendment
14th Amendment
15th Amendment
16th Amendment
17th Amendment
18th Amendment
19th Amendment
20th Amendment
21st Amendment
```

Lab: 정규식 이용하기 (re(p487).py)

- 위의 텍스트는 “[수강 번호][수강 코드][과목 이름]” 형식으로 되어 있다. 위의 텍스트에서 코스 번호만을 추출해보자.

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

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

Sol:

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

import re
s = re.findall("\d+", text) # 한 개의 숫자, 1번이상반복
print(s)
```

Lab: 패스워드 검사 프로그램 (password.py)

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

Sol:

학생> regular expression 사용하지 않고 어떻게 이런 프로그램 만들 수 있나요?

교수> 만들어 보고 있습니다 ^^

```
import re

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

Sol:

```
elif not re.search("[0-9]", password): # 적어도 하나의 숫자
    flag = -1
    break
elif not re.search("[_@$]", password): # 적어도 _,@,$
    flag = -1
    break
else:
    flag = 0
    print("유효한 패스워드")
    break

if flag == -1:
    print("유효한 패스워드가 아닙니다.")
```

예외처리

- 프로그램사용자들은 잘못된 데이터를 입력할 수도 있고, 사용자가 오픈하고자 하는 파일이 컴퓨터에 존재하지 않을 수도 있으며 인터넷이 다운될 수도 있다.

```
>>> (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
>>>
```



예외처리

- 오류가 발생했을 때 오류를 프로그램사용자에게 알려주고 모든 데이터를 저장하게 한 후에 프로그램개발자가 우아하게(**gracefully**) 프로그램을 종료할 수 있도록 하는 것이 바람직
- C언어는 오류가 발생하면 프로그램 종료
- C++, Java, Python은 예외처리기능을 프로그램개발자에게 제공

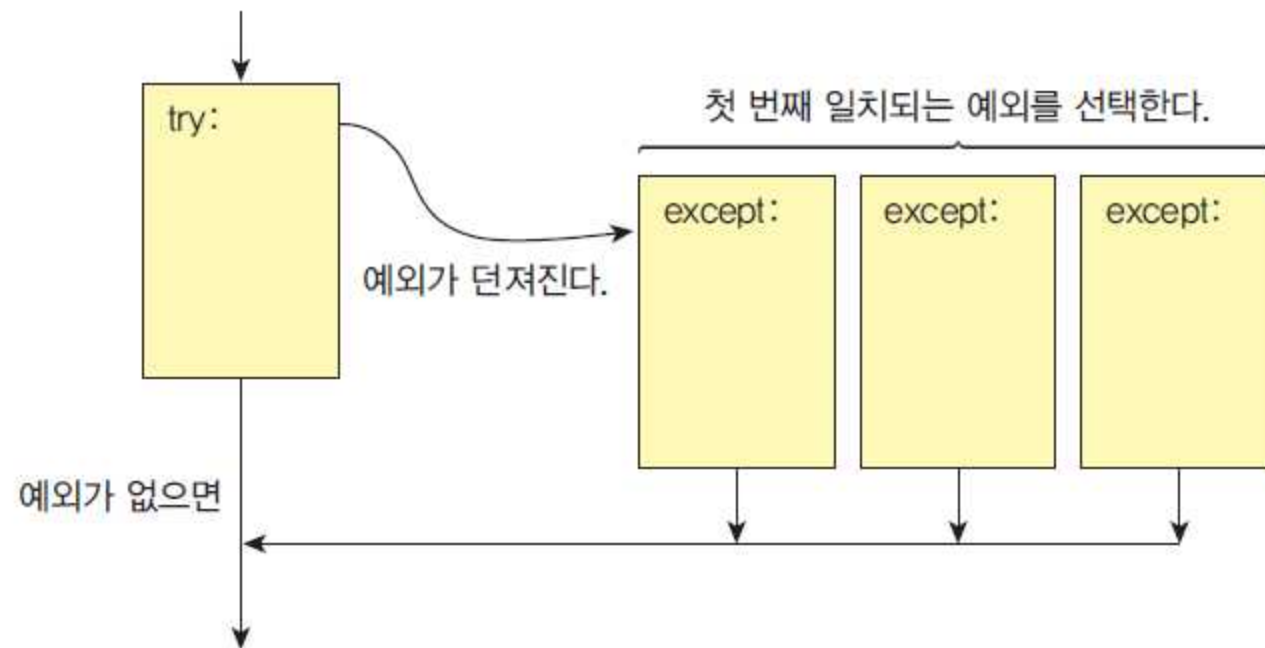


예외 처리는 프로그램의 실행을
계속할 수 있는 다른 경로를 제공한다.

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 적절치않은 값을 가지고 있으면 발생한다.
 - **KeyboardInterrupt**: 사용자가 인터럽트 키를 누르면 발생한다. (Control-C나 Delete)
 - **EOFError**: 내장 함수가 파일의 끝을 만나면 발생한다.

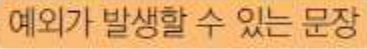

Try-except 구조



Try-except 구조

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")`

예제 (코드파일 없음)

```
(x,y) = (2,0)
try:
    z = x/y
except ZeroDivisionError:
    print ("0으로 나누는 예외")
```

0으로 나누는 예외

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

division by zero

예제 (try.py)

```
while True:
    try:
        n = input("숫자를 입력하시오 : ")
        n = int(n)
        break
    except ValueError:
        print("정수가 아닙니다. 다시 입력하시오. ")
print("정수 입력이 성공하였습니다!")
```

```
숫자를 입력하시오 : 23.5
정수가 아닙니다. 다시 입력하시오.
숫자를 입력하시오 : 10
정수 입력이 성공하였습니다!
```

예제 (try_change.py)

학생> 3.5 입력했다고 프로그램이 중단되는 것은 황당합니다

교수> 그래서 예외처리를 C언어를 제외한 대부분 언어에서 제공하고 있습니다

```
In [2]: runfile('C:/idec/sources/chap10/try_change.py', wdir='C:/idec/
sources/chap10')
```

숫자를 입력하시오 : 3.5

Traceback (most recent call last):

```
File "<ipython-input-2-876f983a929e>", line 1, in <module>
    runfile('C:/idec/sources/chap10/try_change.py', wdir='C:/idec/
sources/chap10')
```

```
File "C:\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 827, in runfile
    execfile(filename, namespace)
```

```
File "C:\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)
```

```
File "C:/idec/sources/chap10/try_change.py", line 13, in <module>
    n = int(n)
```

ValueError: invalid literal for int() with base 10: '3.5'

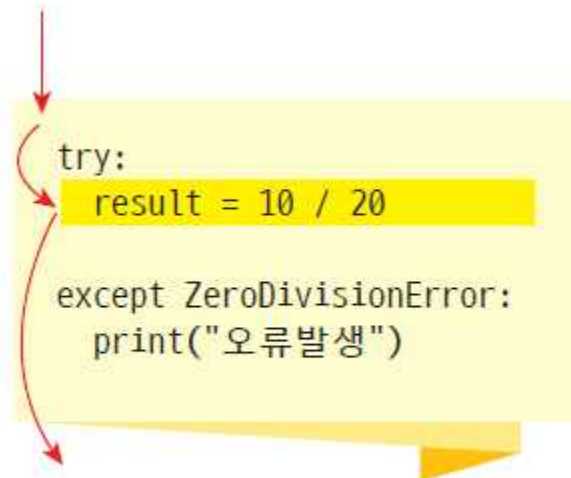
```
In [3]:
```

예제 (try2.py)

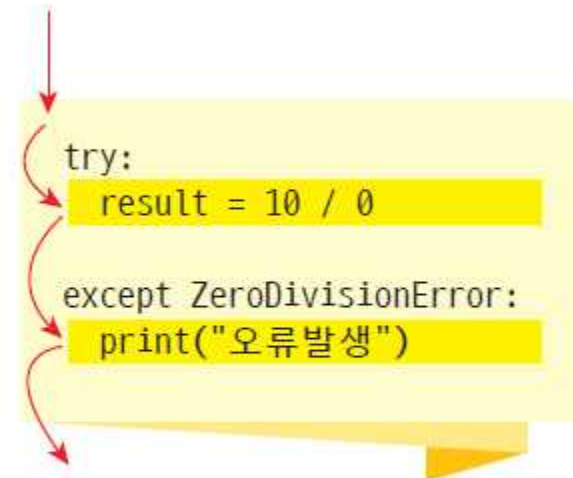
```
try:
    fname = input("파일 이름을 입력하세요: ")
    infile = open(fname, "r")
except IOError:
    print("파일 " + fname + "을 발견할 수 없습니다.")
```

```
파일 이름을 입력하세요: kkk.py
파일 kkk.py을 발견할 수 없습니다.
```

try/except 블록에서의 실행 흐름



예외가 발생하지 않은 경우



예외가 발생하는 경우

다중 예외 처리 구조 (try3.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.

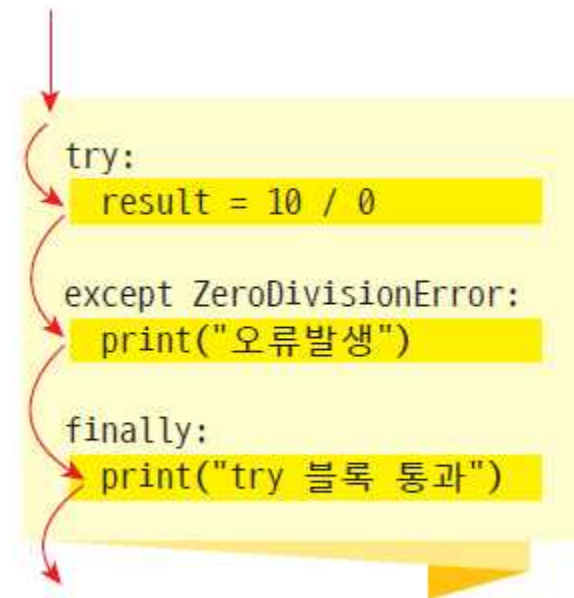
다중 예외 처리 구조 (try3_change.py)

학생> **else**는 예외가 없을 때 실행되네요.

finally 블록



예외가 발생하지 않은 경우



예외가 발생하는 경우

finally 블록의 사용예 (try4.py)

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


try4_change.py

교수> 아래 코드에서 `f.close()`는 예외 유무에 관계없이 실행되는데 `finally`를 왜 사용하는지 모르겠음

```
try:
    f = open("test.txt", "r" )
    f.write("테스트 데이터를 파일에 씁니다!!")
    ...      # pass와 같은 효과
except IOError:
    print("Error: 파일을 찾을 수 없거나 데이터를 쓸 수 없습니다. ")
print("end")
f.close()
```

예외 발생하기

- **raise** 문을 사용하여 프로그램개발자가 예외를 생성할 수 있다

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

이제 나는 할 수 있다

1. 나는 텍스트 파일 읽고 쓰기를 할 수 있다
2. 나는 이진 파일 읽고 쓰기를 할 수 있다
3. 나는 정규식을 사용하는 방법을 할 수 있다
4. 나는 **CSV** 파일 읽고 쓰기를 할 수 있다
5. 나는 예외를 처리하는 방법을 할 수 있다



HW10장-1,2

- HW10장-1,2 게시판에 업로드

프로그래밍 문제 - 1

파이썬 EXPRESS

Programming



01 파일 "input.txt"의 처음 3줄을 읽는 프로그램을 작성하라.

실행결과

```
All's well that ends well.  
Bad news travels fast.  
Well begun is half done.
```

02 파일 "input.txt" 안에 저장된 단어 중에서 가장 긴 단어를 찾는 프로그램을 작성하라.

실행결과

```
가장 긴 단어는 Python입니다.
```

03 파일에서 임의의 행을 읽는 Python 프로그램을 작성하시오.

실행결과

```
파일 이름을 입력하시오: test.txt  
행 번호를 입력하시오: 3  
3번 행은 Well begun is half done.입니다.
```

04 현재 작업 디렉토리에 A.txt, B.txt, ..., Z.txt 까지 총 26개의 텍스트 파일을 생성해보자.

05 왼쪽 파일을 오른쪽 파일로 변환하는 프로그램을 작성하라.

```
line 1  
line 2  
line 3
```



```
line 1  
line 2  
line 2-1  
line 3
```

06 현재의 작업 디렉토리에 "numbers.txt" 파일을 열어서 10개의 정수 값을 문자열 형태로 추가하고 파일을 닫는 프로그램을 작성하라.



```
13  
2  
99  
...
```

프로그래밍문제 - 2

- 07 파일에서 데이터를 읽을 때, 파일이 없으면 IOError가 발생한다. 이것을 try와 except로 처리해보자. 파일이 없으면 "파일이 없습니다. 다시 입력하십시오."를 화면에 출력하고 실행을 계속한다.

실행결과

```
입력 파일 이름: ppp.txt
파일 ppp.txt 이 없습니다. 다시 입력하십시오.
입력 파일 이름: proverbs.txt
파일이 성공적으로 열렸습니다.
```

- 08 학생들의 성적이 부동소수점 수로 파일 scores.txt에 저장되어 있다고 하자(예로장에서 ANS, 엔도링으로 저장한다). 이 성적을 읽어서 파일의 끝에 평균값을 추가하라.

```
99.1
88.2
67.7
96.9
```

```
99.1
88.2
67.7
96.9
평균값: 88.0
```

- 09 사용자로부터 파일 이름과 삭제할 문자열을 입력받는다. 파일을 열어서 사용자가 원하는 문자열을 삭제한 후에 다시 파일에 쓴다.

실행결과

```
파일 이름을 입력하십시오: d:\\words.txt
삭제할 문자열을 입력하십시오: black
변경된 파일이 저장되었습니다.
```

HINT 문자열을 파일에 쓰거나 읽을 방법이 있지만 다음과 같이 print()를 사용하기도 된다.

```
print(modified_s, file = outfile, end = "")
```

- 10 텍스트 파일을 열어서 파일 위의 스페이스 문자의 개수와 탭의 개수를 세는 프로그램을 작성하여 보자.

실행결과

```
파일 이름을 입력하십시오: proverbs.txt
스페이스 수 = 20, 탭의 수 = 0
```

- 11 텍스트 파일을 열어서 각 줄의 앞에 번호를 매겨서 다시 파일에 쓰는 프로그램을 작성해보자.

```
All's well that ends well.
Bad news travels fast.
Well begun is half done.
Birds of a feather flock together.
```

```
1: All's well that ends well.
2: Bad news travels fast.
3: Well begun is half done.
4: Birds of a feather flock together.
```

- 12 트윗 메시지에서 사용자 메시지를 추출해보자. 즉 특수 문자인 URL, 해쉬태그, 이메일 주소, RT, CC는 삭제한다.

실행결과

```
트윗 문자열: 'Good Morning! RT @PythonUser I like Python #Python
정제된 문자열: 'Good Morning! I like Python'
```