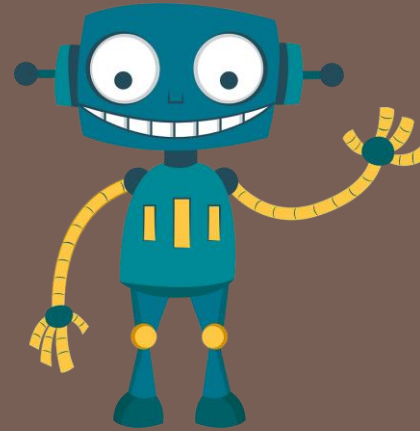
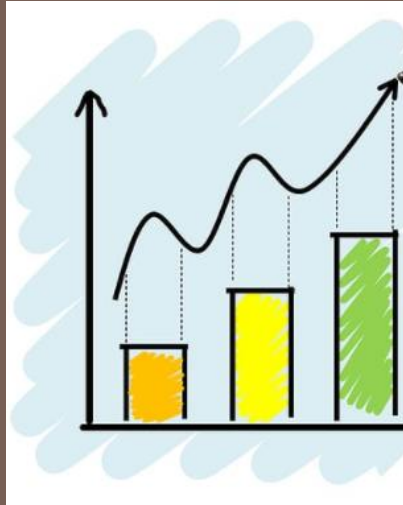


파이썬 익스프레스



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.



이번장에서 만들 프로그램

단어를 추측하시오: a

틀림!

9 기회가 남아있음!

단어를 추측하시오: e

['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']

...

가장 추웠던 날은 -19.2 입니다.

파일의 기초

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



메모리

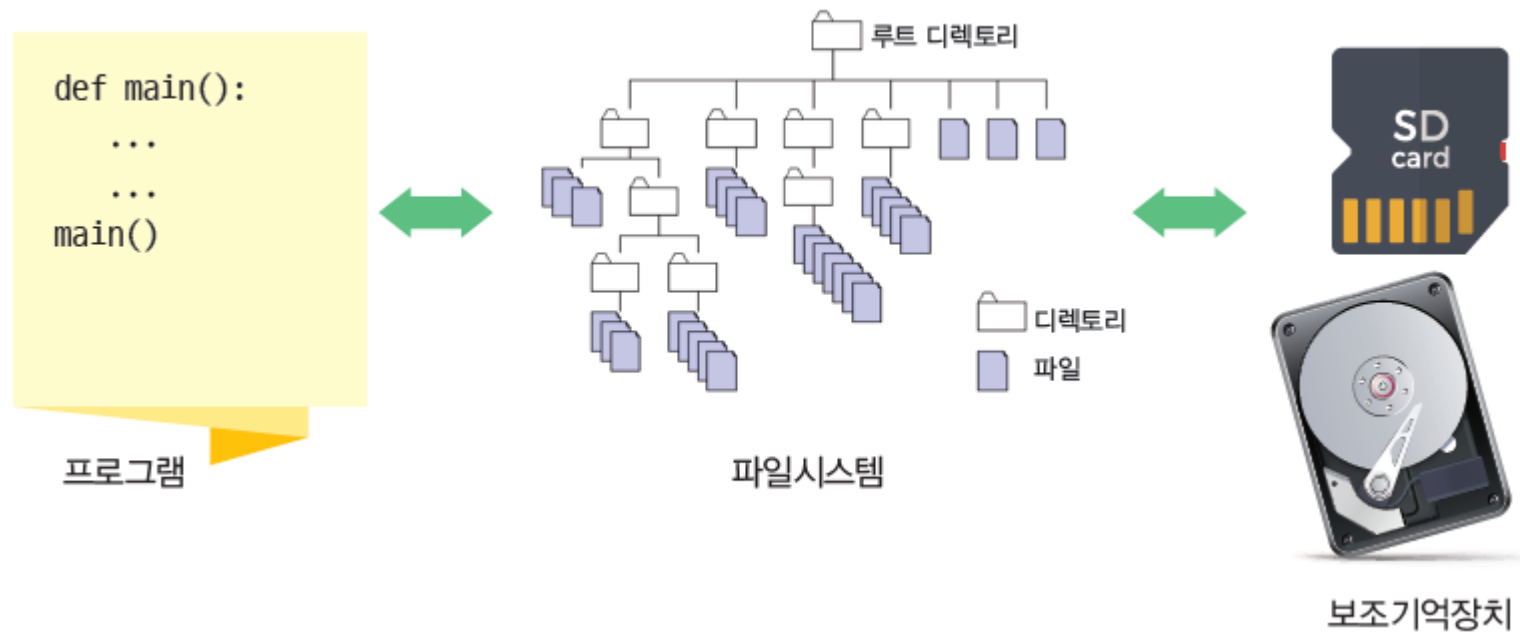
VS



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()`

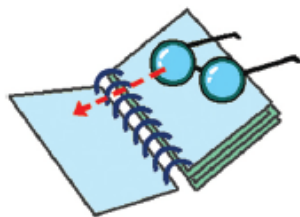
파일 객체

파일의 이름(name)

파일을 여는 모드(mode)

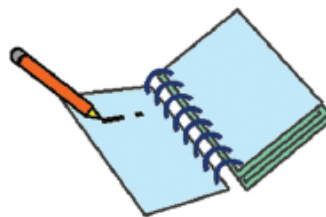
파일 모드

| 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

input.txt

호기도
○ 리
기철스
□ 리

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

호기도
○ 리

기철스
□ 리

파일에서 읽기

input.txt

호기도
○ 리
기척
□ 리

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

호기도
○ 리
기척
□ 리

파일에 쓰기

```
infile = open("d:\\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 빠져나오면 자동으로 파일이 닫혀진다.
```

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
```

5.1 매츠 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()
```

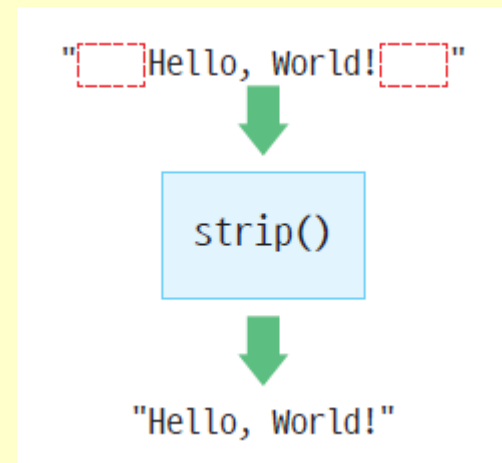
다양한 텍스트 입력 방법

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

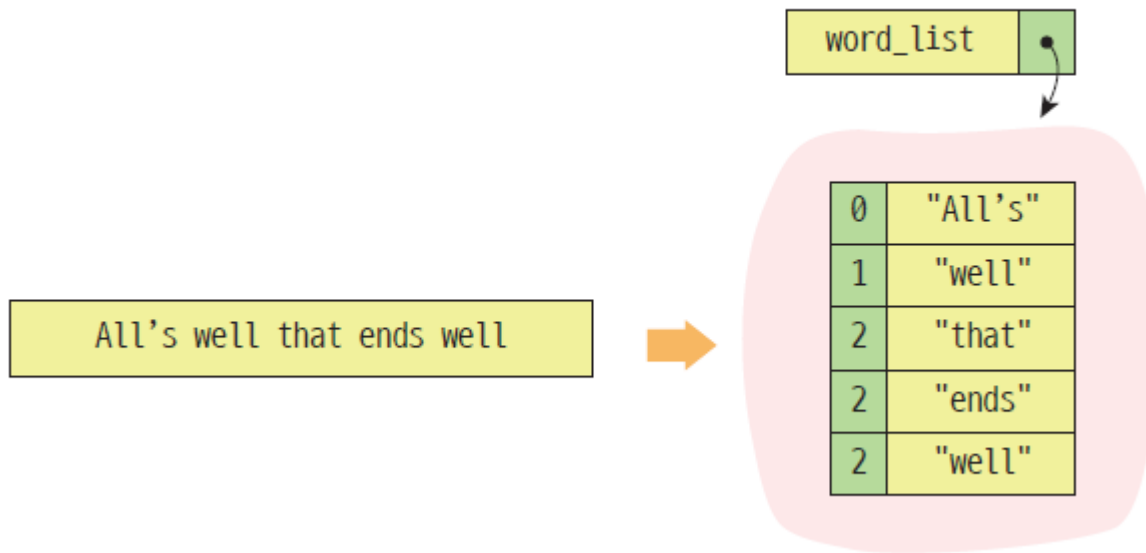
```
>>> s = " Hello, World!\n"
>>> s.strip()
"Hello, World!"
```

```
>>> 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

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

for line in infile:
    line = line.rstrip()
    word_list = line.split()
    for word in word_list:
        print(word);
infile.close()
```

오른쪽 공백 문자를 없앤다.
단어들로 분리한다.
리스트에 들어 있는 단어들 출력한다.

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

파일 전체 읽기

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

호기
심
기
미

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

문자 단위로 읽기

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

```
호
기
동
...
```

문자 출현 횟수 계산

```
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-8**이다. **UTF-8**에서는 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.

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("사옹자 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

- 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:

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

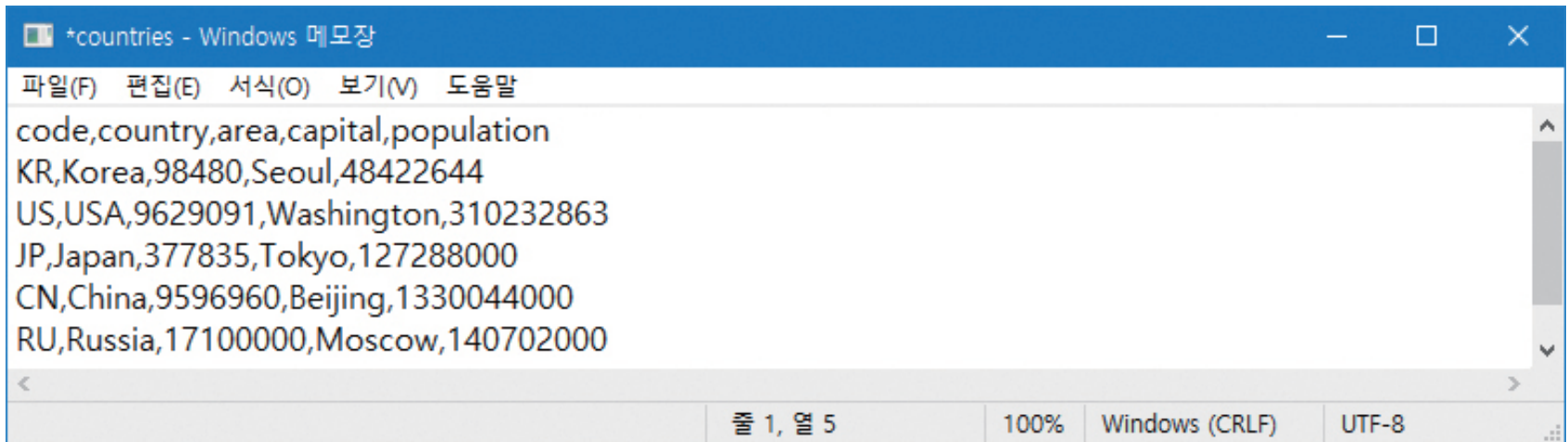
freqs = {}

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

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

Lab: CVS 파일 처리

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



The screenshot shows a Windows Notepad window titled '*countries - Windows 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', '보기(V)', and '도움말'. The text content is a CSV file with the following data:

| 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 |

The status bar at the bottom indicates '줄 1, 열 5', '100%', 'Windows (CRLF)', and '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:

```
import csv
```

```
f = open('d://weather.csv')
```

```
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()
```

CSV 파일을 열어서 f에 저장한다.

가장 추웠던 날은 -19.2 입니다.

Lab: 파일 암호화

- 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)
```

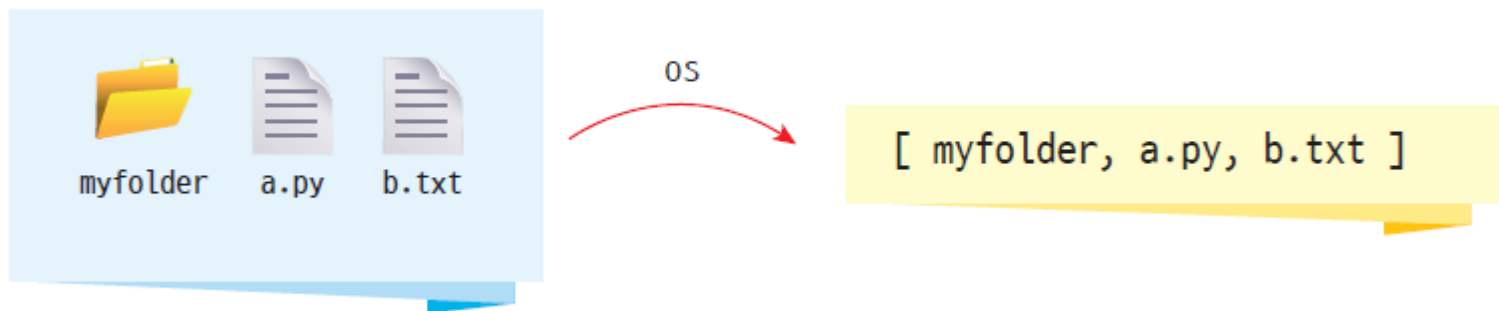
```
decrypted = decrypt(n, encrypted)
```

```
print ('평문: ', text)
```

```
print ('암호문: ', encrypted)
```

```
print ('복호문: ', decrypted)
```


디렉토리 작업



디렉토리 작업

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 디렉토리어서 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

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

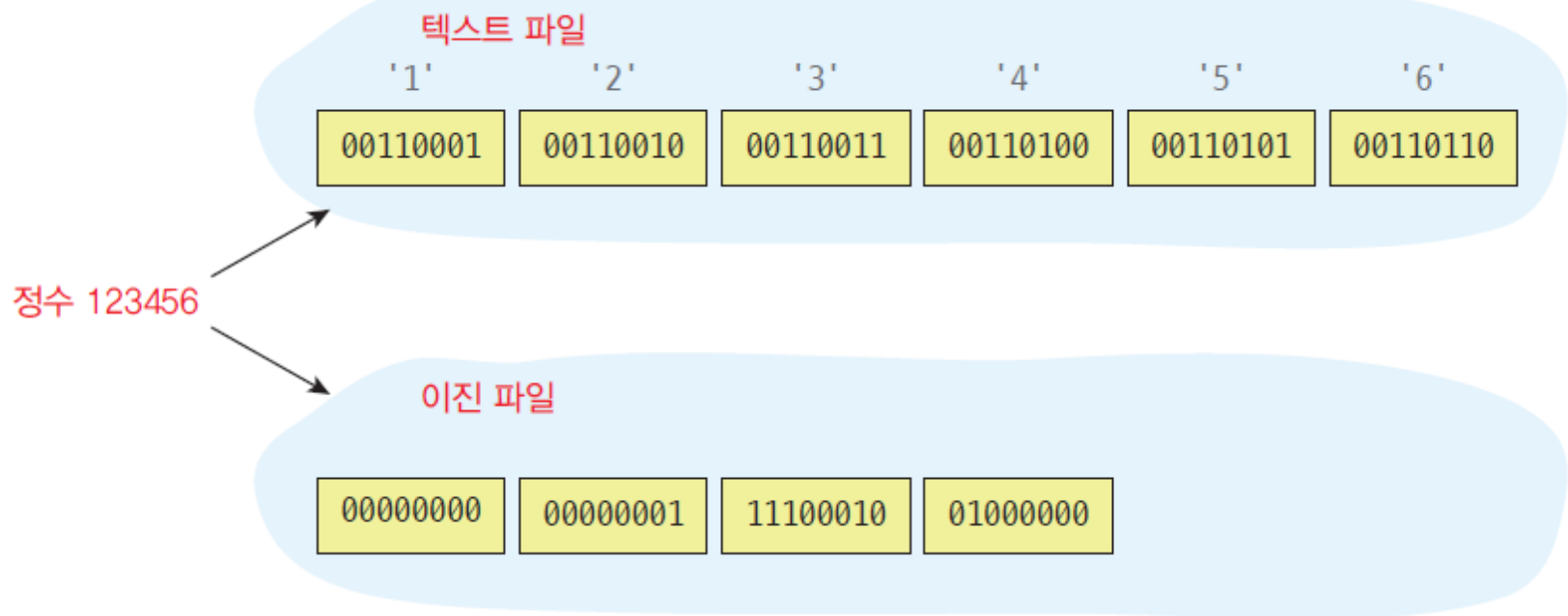
```
file.py :      if "Python" in e:  
summary.txt : The joy of coding Python should be in seeing short  
summary.txt : Python is executable pseudocode.
```

Sol:

```
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

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]
```

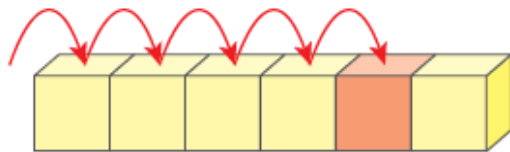
이진 파일에 바이트들을 저장하려면 다음과 같이 한다.

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

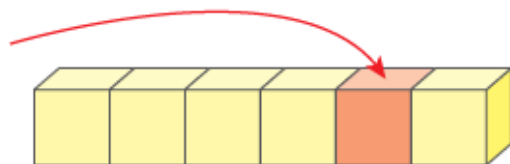
```
>>> byteArray = bytes([255, 128, 0, 1])
```

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

순차 접근과 임의 접근



순차 접근 파일



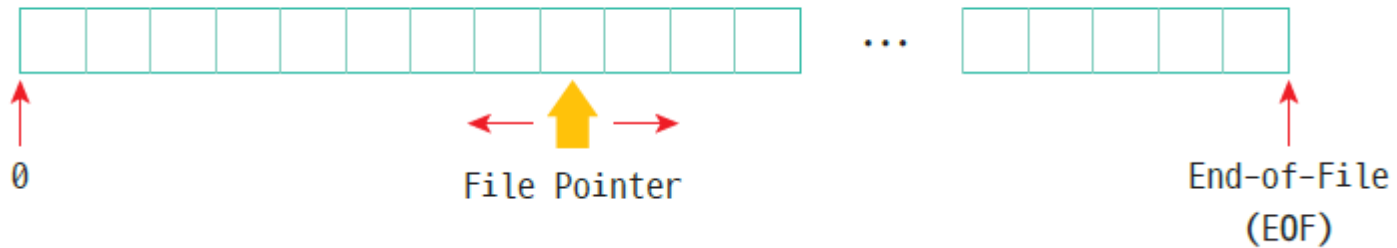
임의 접근 파일

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



임의적그의 위치

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



예제

- 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, 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 프로그램을 작성하여 보자.



Sol:

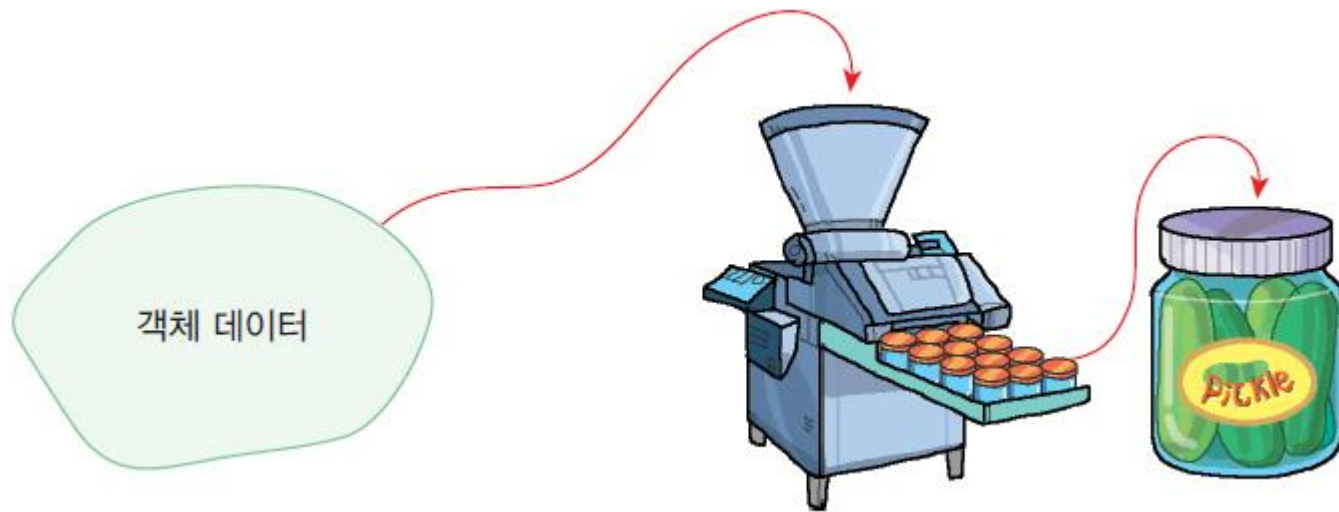
```
infile = open("123.png", "rb")
outfile = open("kkk.png", "wb")

# 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. ")
```

객체 입출력

- pickle 모듈의 `dump()`와 `load()` 메소드를 사용하면 객체를 쓰고 읽을 수 있다.

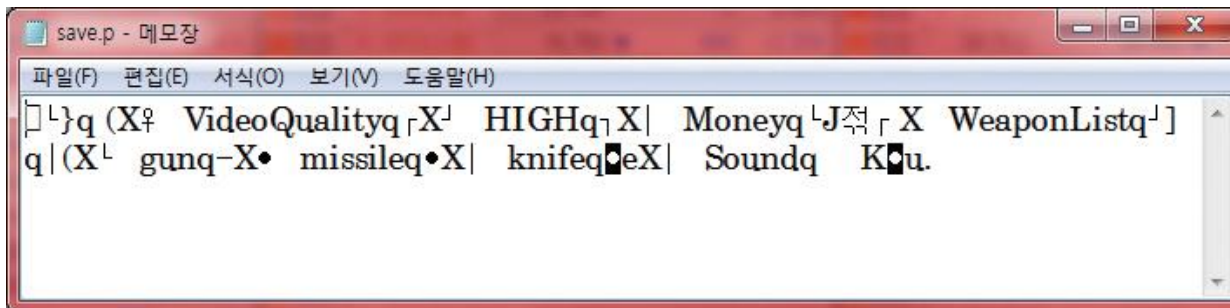


예제

```
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.
```



예제

```
import pickle
```

```
file = open( "d:\\save.p", "rb" )
```

```
# 이진 파일 오픈
```

```
obj = pickle.load( open( "save.p", "rb" ) )
```

```
# 피클 파일에 디렉터리를 로딩
```

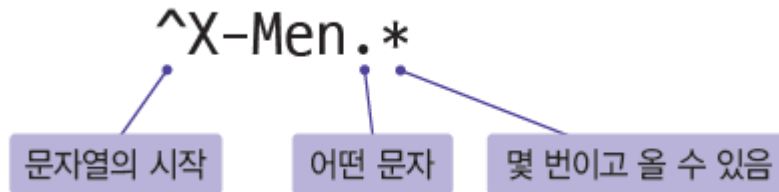
```
print(obj)
```

```
{'WeaponList': ['gun', 'missile', 'knife'], 'Money': 100000, 'VideoQuality':  
'HIGH', 'Sound': 8}
```

- 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

- 미국 헌법에서 숫자로 시작되는 줄만을 출력하는 프로그램은 다음과 같다.

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

```
1. Neither slavery nor involuntary servitude, except as a punishment for
crime
2. Congress shall have power to enforce this article by appropriate
...
```

Lab: 정규식 이용하기

- 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"""
```

```
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[_ , @ , \$]

Sol:

```
import re

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

Sol:

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

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

예외처리

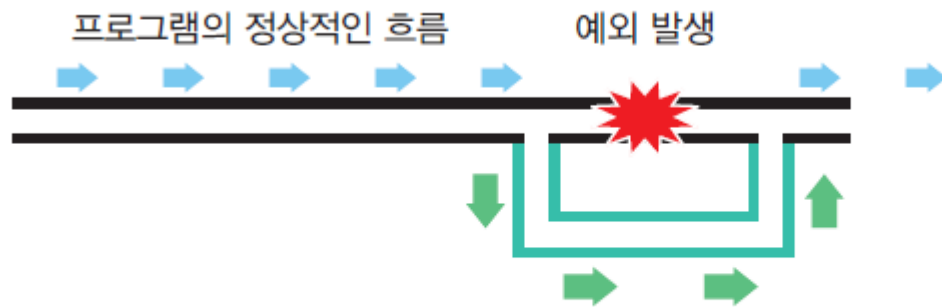
- 사용자들은 잘못된 데이터를 입력할 수도 있고, 우리가 오픈하고자 하는 파일이 컴퓨터에 존재하지 않을 수도 있으며 인터넷이 다운될 수도 있다.

```
>>> (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 사용자에게 알려주고 모든 데이터를 저장하게 한 후에 사용자가 **우아하게(gracefully)** 프로그램을 종료할 수 있도록 하는 것이 바람직

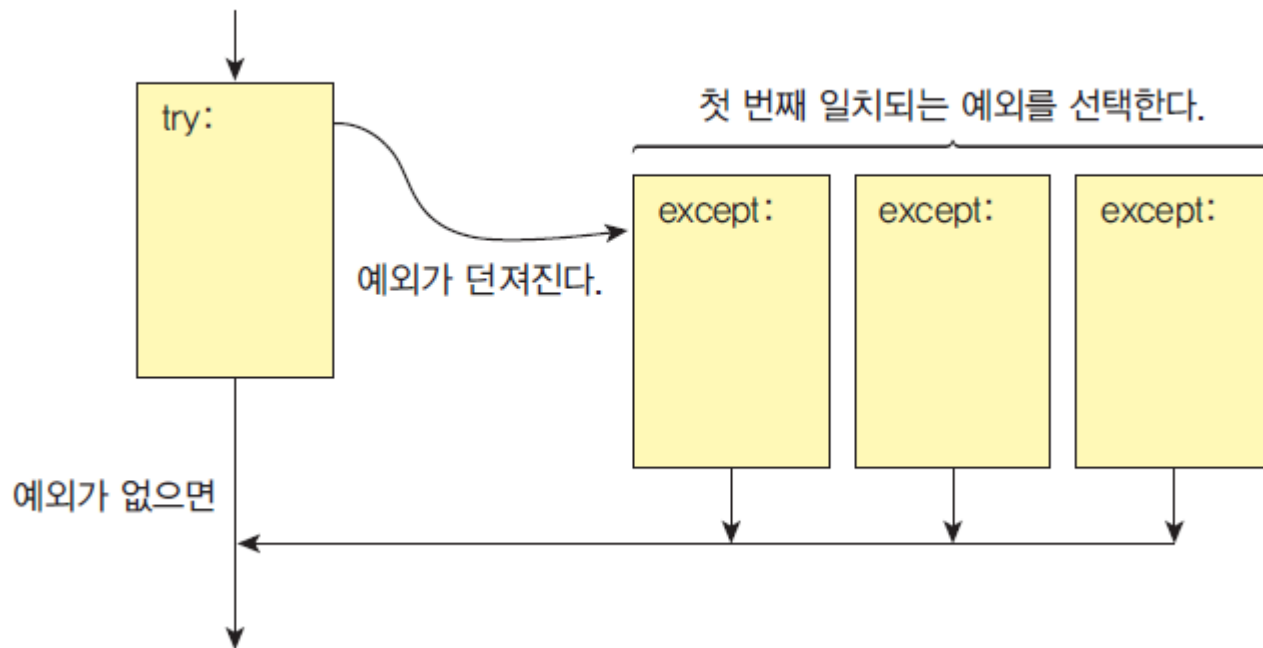


예외 처리는 프로그램의 실행을
계속할 수 있는 다른 경로를 제공한다.

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 으면 발생한다.
 - `KeyboardInterrupt`: 사용자가 인터럽트 키를 누르면 발생한다. (Control-C나 Delete)
 - `EOFError`: 내장 함수가 파일의 끝을 만나면 발생한다.

Try-catch 구조



Try-catch 구조

Syntax: 예외 처리

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

예 `try:`
 `z = x/y` 예외가 발생할 수 있는 문장
`except ZeroDivisionError:`
 `print ("0으로 나누는 예외")` 예외

예제

```
(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

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

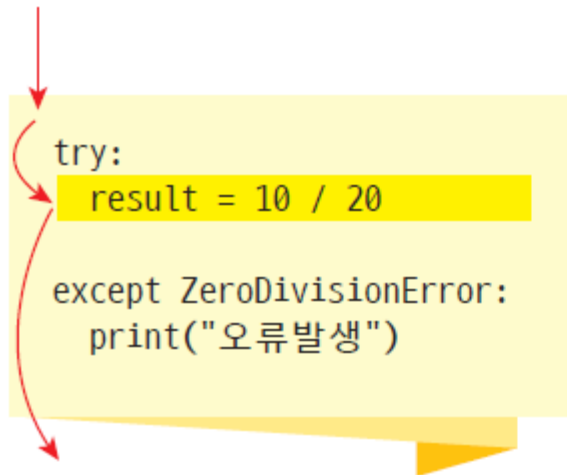
```
숫자를 입력하시오 : 23.5
정수가 아닙니다. 다시 입력하시오.
숫자를 입력하시오 : 10
정수 입력이 성공하였습니다!
```

예제

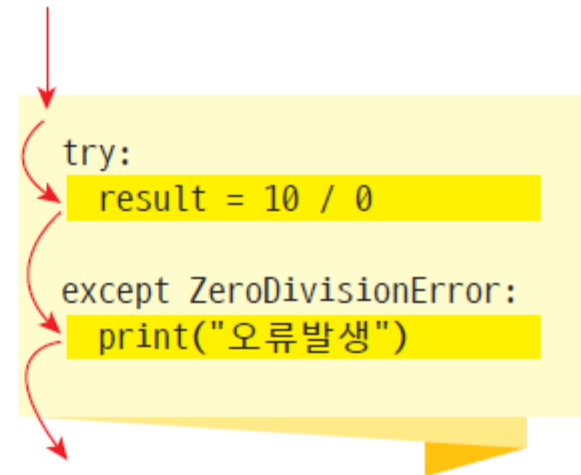
```
try:
    fname = input("파일 이름을 입력하세요: ")
    infile = open(fname, "r")
except IOError:
    print("파일 " + fname + "을 바꿀 수 없습니다.")
```

파일 이름을 입력하세요: kkk.py
파일 kkk.py를 바꿀 수 없습니다.

try/except 블록에서의 실행 흐름



예외가 발생하지 않은 경우



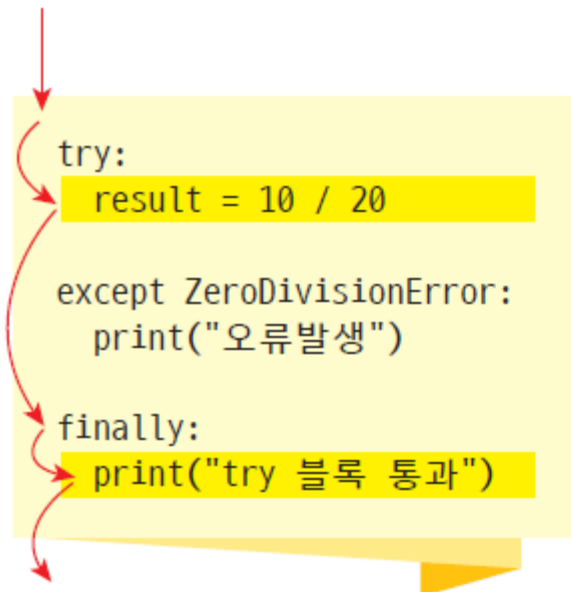
예외가 발생한 경우

다중 예외 처리 구조

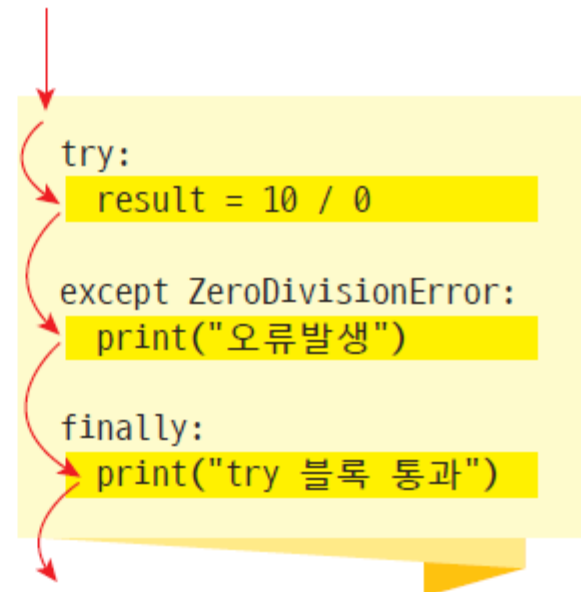
```
try:
    fh = open("testfile", "w")
    fh.write("테스트 데이터를 파일에 씁니다!!")
except IOError:
    print("Error: 파일을 찾을 수 없거나 데이터를 쓸 수 없습니다. ")
else:
    print("파일에 성공적으로 기록하였습니다. ")
    fh.close()
```

파일에 성공적으로 기록하였습니다.

finally 블록



예외가 발생하지 않은 경우



예외가 발생하는 경우

finally 블록의 사용예

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

파일 연산을 수행한다.

예외 발생하기

- 파이썬에서는 오류가 감지되면 **raise** 문을 사용하여 예외를 생성한다.

```
>>> 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 나뉘어진다.
- 파일에서 데이터를 읽거나 쓰는 함수는 read()와 write() 함수이다. 텍스트 파일에서 한 줄을 읽으려면 for 루프를 사용한다.
- 예외 처리는 오류가 발생했을 때 프로그램을 우아하게 종료하는 방법이다. try 블록과 except 블록으로 이루어진다.



Q & A

