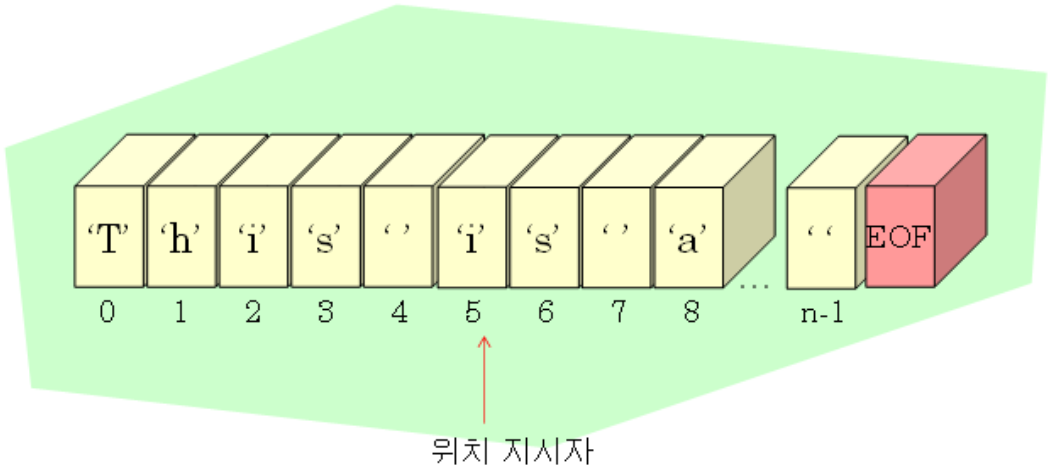
12장 파일과 예외처리

파일의 필요성


객체는 모두 메모리에 만들어지고 이것들은 모두 전원이 꺼지면 사라진다.
메모리
하드 디스크에 파일 형태로 저장하면 전원이 꺼지더라도 데이터가 보존된다.
프로그램
하드 디스크
데이터를 영구히 보관하려면 파일에 저장해야 합니다. 이때 스트림을 사용하면 됩니다.

논리적인 파일 구조

- 파일 안에는 바이트들이 순차적으로 저장되어 있고 맨 끝에는 **EOF(end-of-file)** 마커가 있다.
- 모든 파일은 입출력 동작이 발생하는 위치를 나타내는 위치 표시자**(position indicator)**를 가지고 있다.

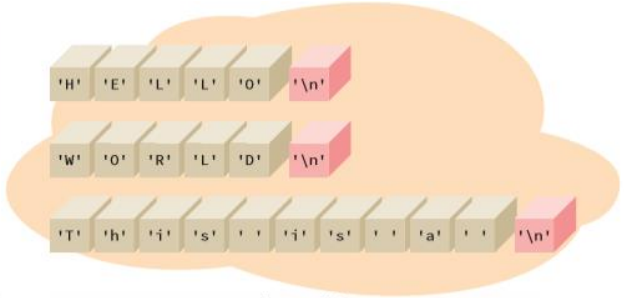


파일의 논리적인 구성

파일의 종류

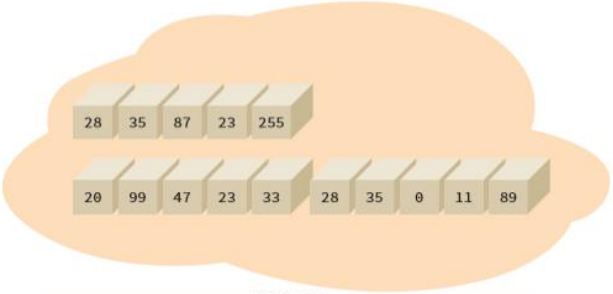
- 텍스트 파일(text file)
- 이진 파일(binary file)

텍스트 파일: 문자로 구성된 파일



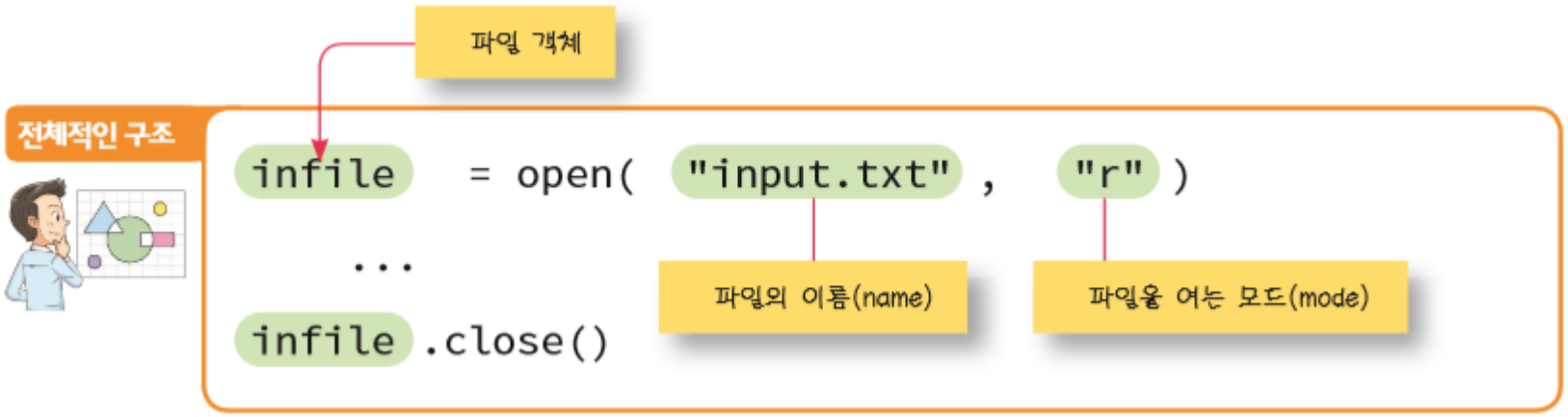
텍스트 파일

이진파일: 데이터로 구성된 파일



이진 파일

텍스트 파일 읽고 쓰기


파일 객체
전체적인 구조
infile = open("input.txt" , "r")
...
infile .close()
파일의 이름(name)
파일을 여는 모드(mode)

파일 모드

파일 모드	모드 이름	설명
"r"	읽기 모드(read mode)	파일의 처음부터 읽는다.
"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"
파일을 처음부터 읽는다.

"w"
파일의 처음부터 쓴다.
만약 파일이 존재하면
기존의 내용이 지워진다.

"a"
파일의 끝에 쓴다.
파일이 없으면 생성된다.

예제

```
infile = open("phones.txt", "r")
s = infile.read(10)
print(s);
infile.close()
```

홍길동 010-12

예제

```
infile = open("phones.txt", "r")
s = infile.readline()
print(s);
s = infile.readline()
print(s);
s = infile.readline()
print(s);
infile.close()
```

홍길동 010-1234-5678
김철수 010-1234-5679
김영희 010-1234-5680

예제

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

홍길동 010-1234-5678
김철수 010-1234-5679
김영희 010-1234-5680

예제

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

홍길동 010-1234-5678
김철수 010-1234-5679
김영희 010-1234-5680

파일에 데이터 쓰기

```
import os.path

outfile = open("phones.txt", "w")

if os.path.isfile("phones.txt"):
        print("동일한 이름의 파일이 이미 존재합니다. ")
else :
        outfile.write("홍길동 010-1234-5678")
        outfile.write("김철수 010-1234-5679")
        outfile.write("김영희 010-1234-5680")

outfile.close()
```

Lab: 매출 파일 처리

입력 파일에는 상점의 하루 매출이 한 줄에 정수로 기록되어 있다. 예를 들면 다음과 같다.

sales.txt

```
1000000
1000000
1000000
500000
1500000
```

출력 파일은 다음과 같아야 한다.

summary.txt

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

Sc

```
# 입력 파일 이름과 출력 파일 이름을 받는다.
infilename = input("입력 파일 이름: ");
outfilename = input("출력 파일 이름: ");

# 입력과 출력을 위한 파일을 연다.
infile = open(infilename, "r")
outfile = open(outfilename, "w")

# 합계와 횟수를 위한 변수를 정의한다.
sum = 0
count = 0

# 입력 파일에서 한 줄을 읽어서 합계를 계산한다.
for line  in infile:
    dailySale = int(line)
    sum = sum + dailySale
    count = count + 1

# 총매출과 일평균 매출을 출력 파일에 기록한다.
outfile.write("총매출 = "+ str(sum)+"\n")
outfile.write("평균 일매출 = "+ str(sum/count ))

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

텍스트 입출력 기법

- 데이터 추가하기

```
outfile = open("phones.txt", "a")

outfile.write("최무선  010-1111-2222")
outfile.write("정중부  010-2222-3333")

outfile.close()
```

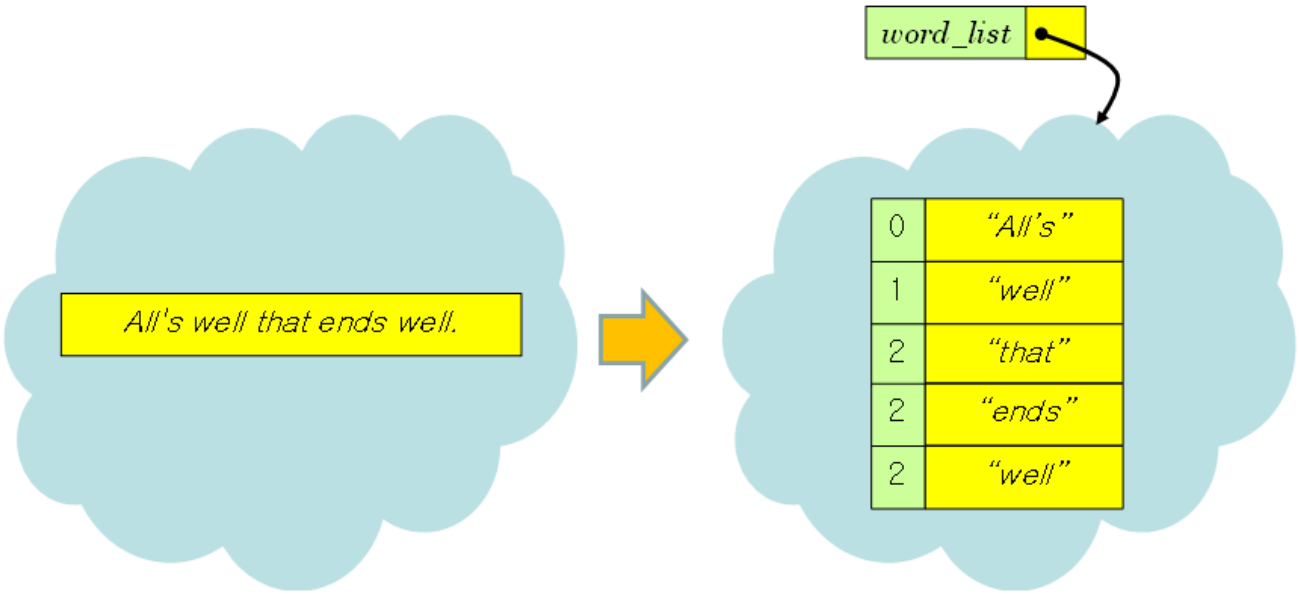
텍스트 입출력 기법

- *줄바꿈 기호 삭제하기*

```
infile = open("proverbs.txt", "r")
for line  in infile:
        line = line.rstrip()
        print(line);
infile.close()
```

텍스트 입출력 기법

- *파일에서 단어 읽기*

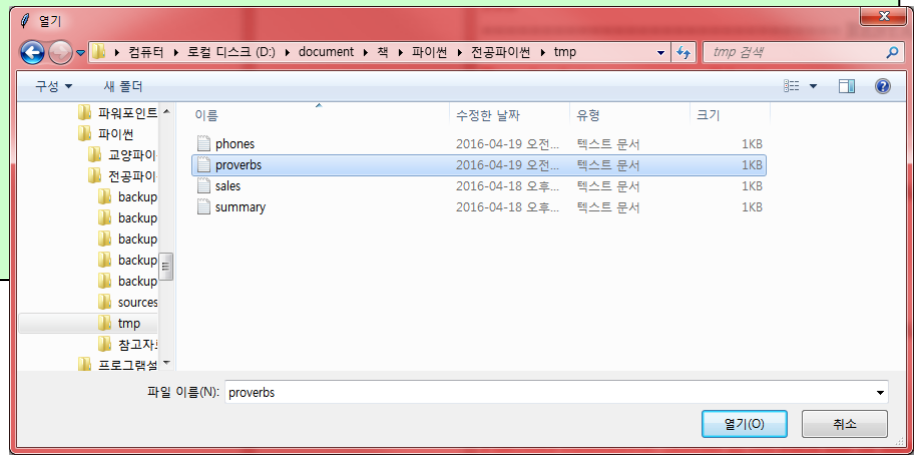


```
infile = open("proverbs.txt", "r")
for line  in infile:
            line = line.rstrip()
            word_list = line.split()
            for word in word_list:
                        print(word);
infile.close()
```

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

파일 대화 상자

```
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter.filedialog import asksaveasfilename

readFile = askopenfilename()
if( readFile != None):
    infile = open(readFile, "r")

for line in infile.readlines():
    line = line.strip()
    print(line)

infile.close()
```



Lab: 스페이스 세기

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

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

Solution

```
def parse_file(path):
    infile = open(path)
    spaces = 0
    tabs = 0
    for line in infile:
        spaces += line.count(' ')
        tabs += line.count('\t')
    infile.close()

    return spaces, tabs

filename = input("파일 이름을 입력하시오: ");
spaces, tabs = parse_file(filename)
print("스페이스 수 = %d, 탬의 수 = %d" % (spaces, tabs))
```

Lab: 줄앞에 번호붙이기

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

```
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.
```

Solution

```
infile  = open("proverbs.txt")
outfile = open("output.txt","w")
i = 1
for line in infile:
    outfile.write(str(i) + ": " + line)
    i = i + 1
infile.close()
outfile.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}

Solution

```
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():
        if char in freqs:
            freqs[char] += 1
        else:
            freqs[char] = 1
print(freqs)
infile.close()
```

Lab: CSV 파일 읽기

- **CSV(Comma Separated Values)** 형식은 엑셀과 같은 스프레드 쉬트나 데이터베이스에서 가장 널리 사용되는 입출력 형식이다. 파이썬은 **CSV** 형식을 읽기 위해서 **csv**라고 하는 모듈을 제공한다. 이 모듈을 이용하면 **CSV** 파일을 쉽게 읽을 수 있다. 우리는 연습 삼아서 **CSV** 형식의 파일을 읽는 코드를 작성하여 보자.

```
1/2/2014,5,8,red
    1/2/2014
    5
    8
    red
1/3/2014,5,2,green
    1/3/2014
    5
    2
    green
1/4/2014,9,1,blue
    1/4/2014
    9
    1
    blue
```

Solution

```
# 파일을 연다.
f = open("C:\test.csv", "r")

# 파일 안의 각 줄을 처리한다.
for line in f.readlines():

    # 공백 문자를 없앤다.
    line = line.strip()

    # 줄을 출력한다.
    print(line)

    # 줄을 쉼표로 분리한다.
    parts = line.split(",")

    # 각 줄의 필드를 출력한다.
    for part in parts:
        print("   ", part)
```

Lab: 파일 암호화

- 예를 들어 평문 **"come to me"**은 **"FRPH WR PH"**으로 바뀐다. 시저 암호 방식을 이용하여서 파일을 암호화하고 복호화하는 프로그램을 작성하라.

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

Solution

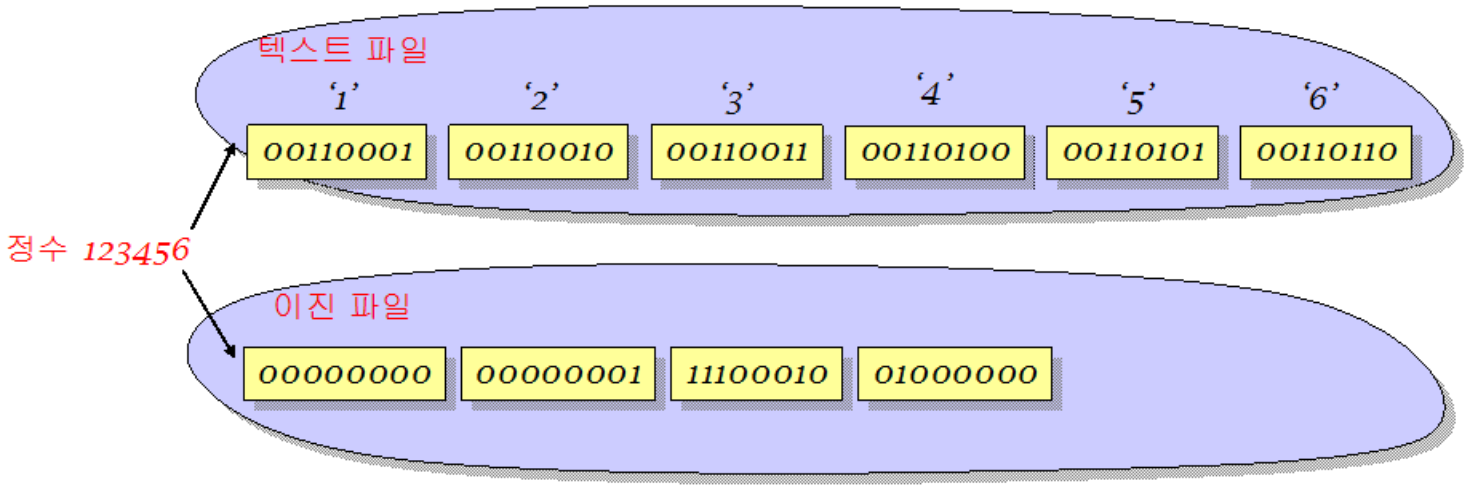
```
key = 'abcdefghijklmnopqrstuvwxyz'

# 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()
```

Solution

```
# 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)
```

이진 파일

- 이진 파일(**binary file**)은 데이터가 직접 저장되어 있는 파일이다.



Lab: 이미지 파일 복사하기

- 예를 들어 평문 **"come to me"**은 **"FRPH WR PH"**으로 바뀐다. 시저 암호 방식을 이용하여서 파일을 암호화하고 복호화하는 프로그램을 작성하라.

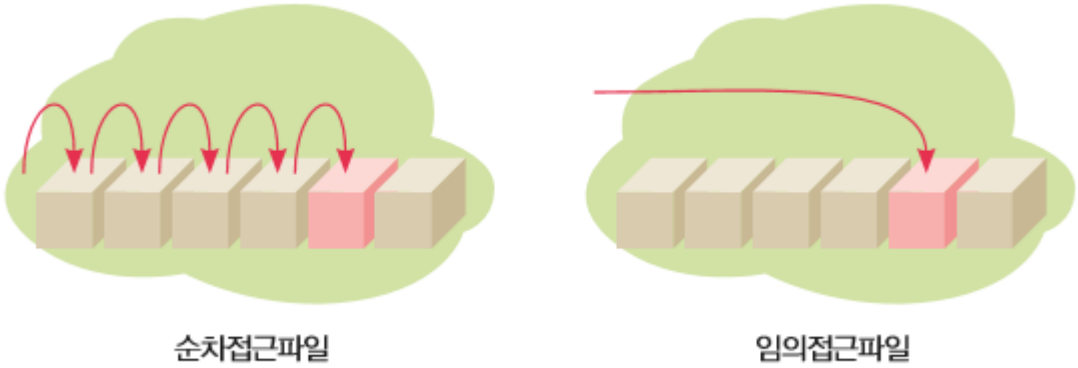
```
원본 파일 이름을 입력하시오: 123.png
복사 파일 이름을 입력하시오: kkk.png
123.png를 kkk.png로 복사하였습니다.
```

Solution

```
filename1 = input("원본 파일 이름을 입력하시오: ");
filename2 = input("복사 파일 이름을 입력하시오: ");

infile = open(filename1, "rb")
outfile = open(filename2, "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. ")
```

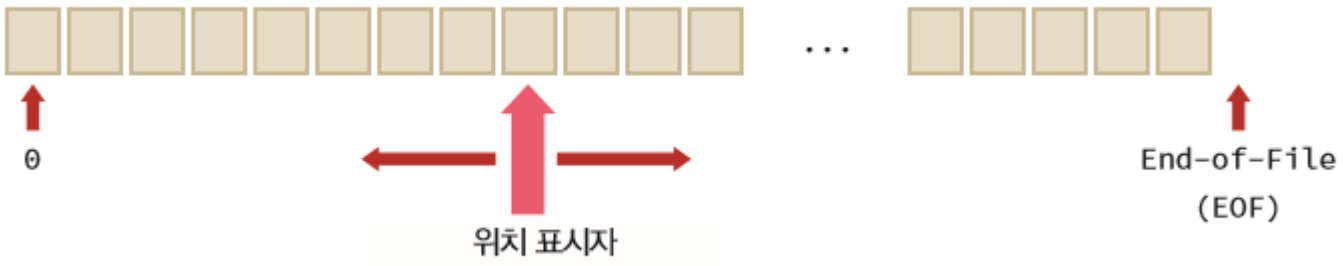
임의 접근 파일

- 파일 포인터를 이동시켜서 랜덤하게 읽는다.



임의 접근의 원리

- 위치 표시자의 이동



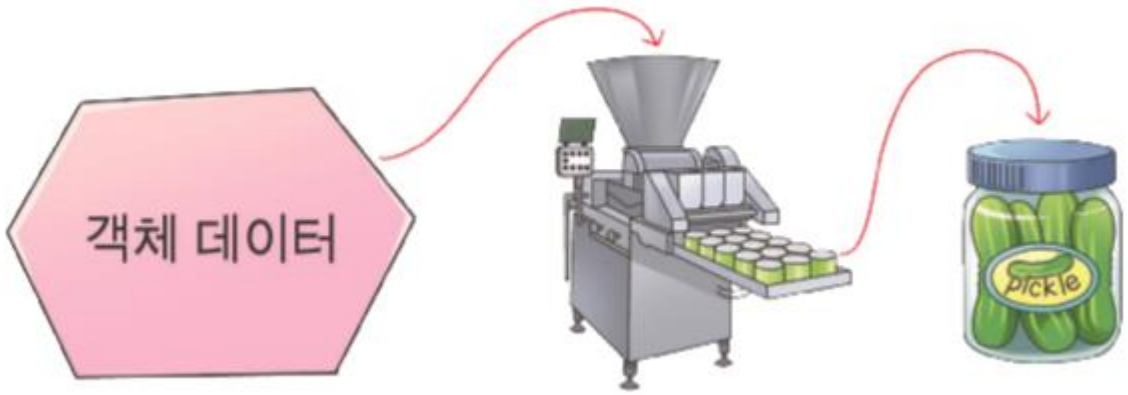
예제

```
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("다시 읽은 문자열 : ", str)
infile.close()
```

```
읽은 문자열 :  All's well
현재 위치:   10
다시 읽은 문자열 :  All's wel
```

객체 입출력

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



예제

```
import pickle

myMovie = { "Superman vs Batman ": 9.8, "Ironman": "9.6" }

# 딕셔너리를 피클 파일에 저장
pickle.dump( myMovie, open( "save.p", "wb" ) )

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

```
{'Superman vs Batman ': 9.8, 'Ironman': '9.6'}
```

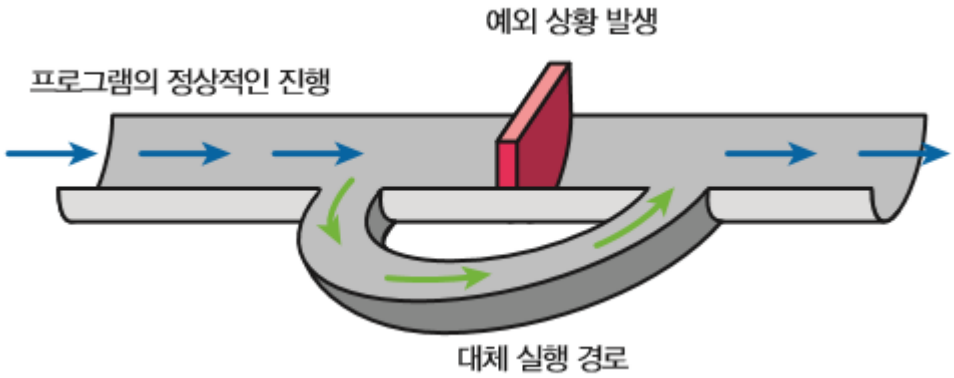
에외 처리

- 오류가 발생할 수 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
>>>
```

예외 처리의 개념

- 오류가 발생 했을 때 오류를 사용자에게 알려주고 모든 데이터를 저장하게 한 후에 사용자가 우아 하게 **(gracefully)** 프로그램을 종료할 수 있도록 하는 것이 바람직하다.



파이썬에서의 예외 처리

전체적인 구조

```
try :
    예외가 발생할 수 있는 문장
except 오류내용 :
    예외를 처리하는 문장
```

예제

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

```
0으로 나누는 예외
```

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 발견할 수 없습니다.")
```

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

다중 예외 처리 구조

전체적인 구조

```
try :
    예외가 발생할 수 있는 문장
except ExceptionI :
    ExceptionI이면 이 블록이 실행된다.
except ExceptionII :
    ExceptionII이면 이 블록이 실행된다.
else :
    예외가 없는 경우에 실행된다.
```

예제

```
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.

핵심 정리

- 파일은 텍스트 파일과 이진 파일로 나누어진다. 파일은 연 후에 입출력이 끝나면 반드시 닫아야 한다.
- 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.

Q & A