스크립트 프로그래밍

13 파일과 예외처리

2016 2학기 (02분반)

강승우

학습 목표

- 데이터 읽기및 쓰기를 위해 open 함수를 사용하여 파일을 열 수 있다(13.2.1절).
- file 객체의 write 메소드를 사용하여 파일에 데이터를 기록할 수 있다(13.2.2절).
- os.path.isfile 함수를 사용하여 파일이 존재하는지 검사할 수 있다(13.2.3절).
- 파일 객체의 read, readline, readlines 메소드를 사용하여 파일에서데이터를 읽을 수 있다 (13.2.4-13.2.5절).
- append 모드로 파일을 열어 파일에 데이터를 추가할 수 있다(13.2.6절).
- 숫자 데이터를 읽고 기록할 수 있다(13.2.7절).
- 데이터 읽기및 쓰기에 사용할 파일 이름을 얻기 위한 열기및 쓰기파일 다이얼로그를 표시할 수 있다(13.3절).
- 파일을 이용한 애플리케이션을 개발할 수 있다(13.4절).
- 웹자원으로부터 데이터를 읽을 수 있다(13.5절).
- try, except 및 finally 절을 사용하여 예외를 처리할 수 있다(13.6절).
- raise 명령문을 사용하여 예외를 발생시킬 수 있다(13.7절).
- 파이썬의 내장 예외 클래스를 자유롭게 사용할 수 있다(13.8절).
- 핸들러의 예외 객체에 접근할 수 있다(13.8절).
- 사용자 정의 예외 클래스를 생성할 수 있다(13.9절).
- pickle 모듈의 load와 dump 함수를 사용하여 이진 IO를 수행할 수 있다(13.10절).
- 이진 IO를 사용하여 주소록을 생성할 수 있다(13.11절).

파일 입출력

- 파일 사용
  - 프로그램에서 생성된 데이터를 영구적으로 저장하기 위함
    - 프로그램에서 사용되는 저장되지 않는 이상 프로그램이 종료되면 소멸됨
  - 추후 해당 프로그램 혹은 다른 프로그램에서 읽을 수 있음

- 파일
  - 텍스트 파일
    - 윈도우의 메모장이나 리눅스의 vi 같은 텍스트 편집기를 사용하여 처리 (생성, 읽기, 쓰기 등)
    - 텍스트 파일의 문자는 ASCII나 유니코드 같은 문자 인코딩 체계를 사용하여 인코딩
    - 파이썬 소스 프로그램은 텍스트 파일로 저장
  - 바이너리 파일
    - 텍스트 파일이 아닌 모든 파일
    - MS 워드 파일은 바이너리 파일로 저장, MS 워드 프로그램에 의해 처리

파일 열기

- open() 함수 이용

➢파일 변수 = open(파일명, 모드)

  - “파일명”에 해당하는 디스크 상에 존재하는 파일에 대한 객체를 생성
  - 모드
    - 파일을 어떤 용도로 열 것인지 명시

모드	설명
‘r’	파일을 읽기 용도로 연다
‘w’	새로운 파일을 쓰기 용도로 연다. 이미 파일이 존재하면 이전 내용은 모두 파괴된다.
‘a’	파일의 끝에 데이터를 덧붙이기 용도로 연다.
‘rb’	바이너리 데이터를 읽기 용도로 파일을 연다.
‘wb’	바이너리 데이터를 쓰기 용도로 파일을 연다.

파일 열기 예제

- input = open(“Scores.txt”, “r”)
  - 현재 디렉토리(폴더)에 있는 Scores.txt라는 파일을 읽기 용으로 연다
- input = open(“c:\\pybook\\Scores.txt”, “r”)
  - C 드라이브의 pybook 디렉토리에 있는 Scores.txt 파일을 읽기 용으로 연다
  - \\ : 문자열에서 역슬래시(\)를 나타내는 이스케이프 시퀀스
  - → open(r”c:\pybook\Scores.txt”, “r”)
    - 파일명을 나타내는 문자열 앞에 r을 붙여주면 문자열이 raw string(미가공 문자열)임을 표시하여 \가 이스케이프 시퀀스가 아닌 문자 그대로 역슬래시로 취급하도록 함

파일 열기

- open 함수 호출 결과
  - _io.TextIOWrapper 클래스의 인스턴스인 파일 객체를 생성

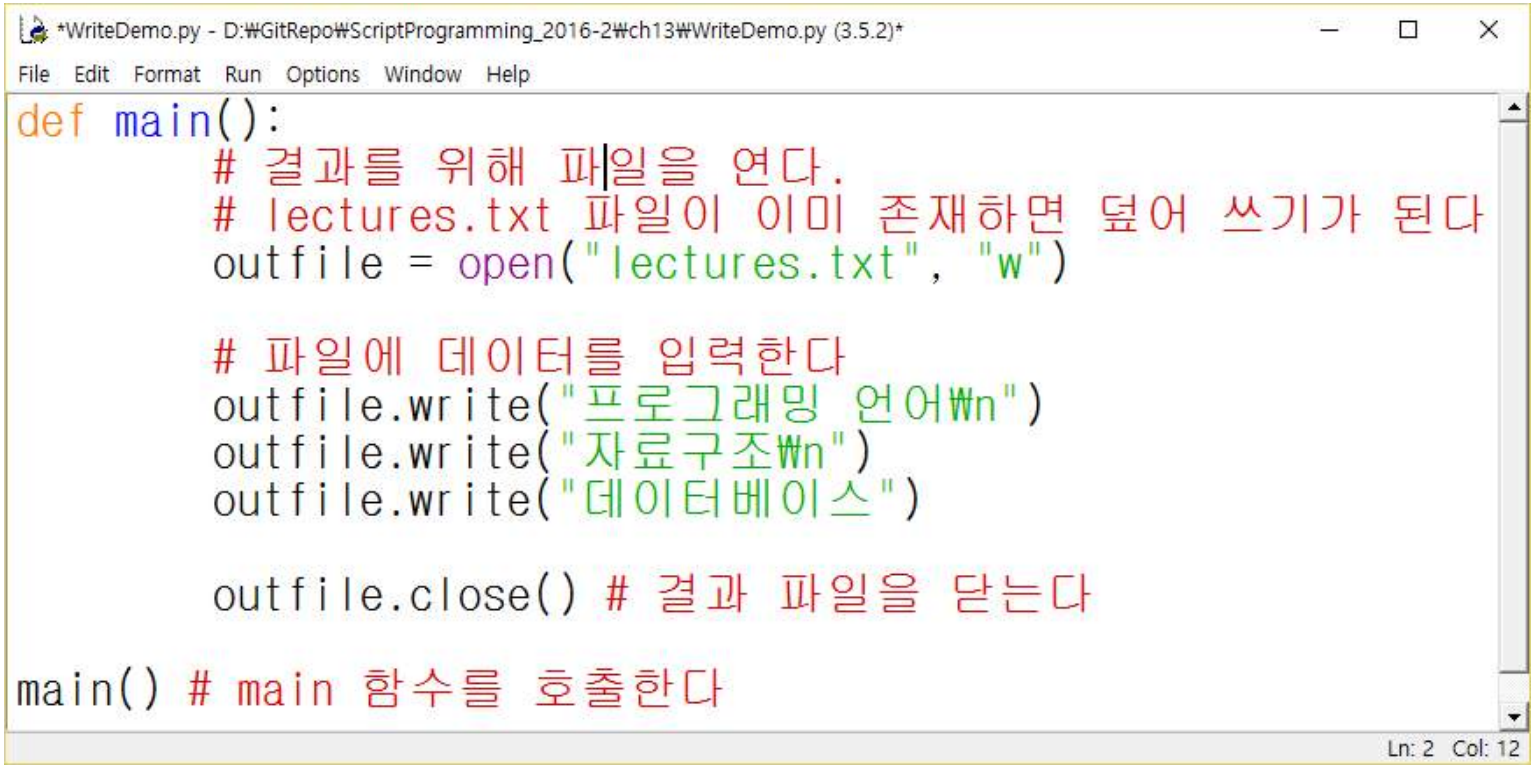
```
>>> file = open("temp.txt", "w")
>>> type(file)
<class '_io.TextIOWrapper'>
```

- _io.TextIOWrapper 클래스 제공 메소드
  - read([number: int]): str
  - readline(): str
  - readlines(): list
  - write(s: str): int
  - close(): None

데이터 쓰기

```
>>> file = open("temp.txt", "w")
>>> file.write("Welcome to Python")
17
>>> file.close()
```

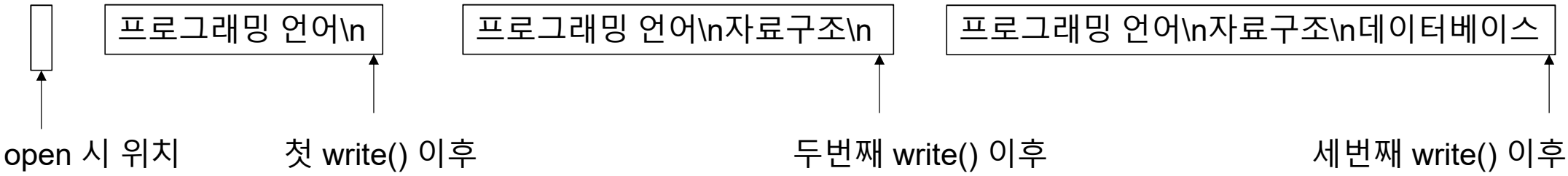
데이터 쓰기


WriteDemo.py - D:₩GitRepo₩ScriptProgramming_2016-2₩ch13₩WriteDemo.py (3.5.2)
File Edit Format Run Options Window Help
def main():
# 결과를 위해 파일을 연다.
# lectures.txt 파일이 이미 존재하면 덮어 쓰기가 된다
outfile = open("lectures.txt", "w")
# 파일에 데이터를 입력한다
outfile.write("프로그래밍 언어₩n")
outfile.write("자료구조₩n")
outfile.write("데이터베이스")
outfile.close() # 결과 파일을 닫는다
main() # main 함수를 호출한다
Ln: 2 Col: 12

파일 포인터

- 파일이 열리면 파일 포인터(file pointer)라는 특수 표시자가 파일 내부에 위치
- 이 포인터가 가리키는 위치에서 읽기와 쓰기 작업이 수행
- 파일이 열리면 파일 포인터는 파일의 시작 부분을 가리킴

- 앞의 예제



파일 존재 여부 검사하기

- 파일 쓰기 모드로 열 때
  이미 존재하는 파일의 데이터가 실수로 삭제되는 것을 방지하기 위해
  해당 파일이 존재하는지 확인한다
- os.path 모듈 사용
  - os.path 모듈의 isfile() 함수

```
>>> import os.path
>>> if os.path.isfile("temp.txt"):
        print("temp.txt 파일이 존재합니다.")


temp.txt 파일이 존재합니다.
```

데이터 읽기

- 읽기 모드로 파일을 열고 read(), readline(), readlines() 메소드 이용
  - read(): 특정 개수의 문자를 읽거나 파일의 모든 문자를 읽고 문자열로 반환
  - readline(): 한 줄을 읽고 문자열로 반환
  - readlines(): 모든 줄을 읽고 리스트 형태 반환

데이터 읽기

```
# 입력을 위한 파일을 연다
infile = open("Lectures.txt", "r")
print("(1) read() 사용하기: ")
print(infile.read() )
infile.close() # 입력 파일을 닫는다
```

실행 결과는?

데이터 읽기

```
# 입력을 위한 파일을 연다
infile = open("Lectures.txt", "r")
print("\n(2) read(number) 사용하기: ")
s1 = infile.read(5)
print(s1)
s2 = infile.read(7)
print(repr(s2) )
infile.close() # 입력 파일을 닫는다
```

repr() 함수:
미가공 문자열을 반환

```
>>> s = "프로그래밍 언어\n자료구조\n데이터베이스"
>>> s
'프로그래밍 언어\n자료구조\n데이터베이스'
>>> print(s)
프로그래밍 언어
자료구조
데이터베이스
>>> print(repr(s))
'프로그래밍 언어\n자료구조\n데이터베이스'
```

실행 결과는?

데이터 읽기

```
# 입력을 위한 파일을 연다
infile = open("Lectures.txt", "r")
print("\n(3) readline() 사용하기: ")
line1 = infile.readline()
line2 = infile.readline()
line3 = infile.readline()
line4 = infile.readline()
print(repr(line1))
print(repr(line2))
print(repr(line3))
print(repr(line4))
infile.close() # 입력 파일을 닫는다
```

실행 결과는?

데이터 읽기

```
# 입력을 위한 파일을 연다
infile = open("Lectures.txt", "r")
print("\n(4) readlines() 사용하기: ")
print(infile.readlines() )
infile.close() # 입력 파일을 닫는다
```

실행 결과는?

파일에서 모든 데이터 읽기

- read() 메소드 이용
  - 파일에서 모든 데이터를 읽고 하나의 문자열로 반환
- readlines() 메소드 이용
  - 모든 데이터를 읽고 문자열의 리스트로 반환

- 만약 파일의 크기가 매우 커서 모든 내용이 메모리에 저장될 수 없다면?
  - 루프를 이용하여 파일의 끝에 다다를 때까지 한 번에 한 줄 씩 읽고 처리하는 방법

파일에서 모든 데이터 읽기

```
# 파일의 줄 수 세기
countLines = 0
infile = open(“Lectures.txt”, “r”)
for line in infile:
    countLines += 1
    print(line, end=“”)
print(“# of lines: “, countLines)
```

infile 파일 객체에 있는 모든 줄에 대해서 루프를 돈다

파일 복사하기

```
import os.path
import sys

def main():
    # 사용자로부터 파일이름을 입력받는다.
    f1 = input("원본 파일을 입력하세요: ").strip()
    f2 = input("대상 파일을 입력하세요: ").strip()

    # 대상 파일이 존재하는지 검사한다.
    if os.path.isfile(f2):
        print(f2 + " 이미 존재하는 파일입니다.")
        sys.exit()

    # 입력 및 결과 파일을 연다.
    infile = open(f1, "r")
    outfile = open(f2, "w")

    # 입력 파일에서 결과 파일로 복사한다.
    countLines = countChars = 0
    for line in infile:
        countLines += 1
        countChars += len(line)
        outfile.write(line)
    print(countLines, "개 행과", countChars, "개 문자가 복사

    infile.close()  # 입력 파일을 닫는다.
    outfile.close() # 결과 파일을 닫는다.

main() # main 함수를 호출한다.
```

데이터 추가하기

- 기존 파일에 데이터를 추가
  - 파일을 append 모드(“a”)로 연다

```
file = open(“Lectures.txt”, “a”)
file.write(“\n운영체제”)
file.close()
```

숫자 데이터 쓰고 읽기

- 파일에 숫자 쓰기
  - 숫자를 문자열로 변환한 후 파일에 쓴다
- 읽기
  - 문자열을 읽은 후에 eval() 함수를 통해 숫자로 변환한다

숫자 데이터 쓰고 읽기

WriteReadNumbers.py - D:₩GitRepo₩ScriptProgramming_2016-2₩ch13₩WriteReadNumbers.py (3.5.2)

File Edit Format Run Options Window Help

```
from random import randint

def main():
    # 데이터 쓰기 모드로 파일을 연다.
    outfile = open("Numbers.txt", "w")
    for i in range(10):
        outfile.write(str(randint(0, 9)) + " ")
    outfile.close() # 파일을 닫는다.

    # 데이터 읽기 모드로 파일을 연다.
    infile = open("Numbers.txt", "r")
    s = infile.read()
    numbers = [eval(x) for x in s.split()]
    for number in numbers:
        print(number, end = " ")
    infile.close() # 파일을 닫는다.

main() # main 함수를 호출한다.
```

Ln: 7 Col: 47

사례 연구: 파일의 각 문자별 문자 개수 세기

- 사용자에게 파일명을 입력받고
  파일에 포함된 각 문자가 대소문자 구분 없이 몇 번씩 나오는지
  횟수를 세는 프로그램

- 단계
  - 파일명을 입력 받고 파일을 읽기 모드로 연다 (예제: Letters.txt 파일)
  - 문자 개수를 저장할 counts 리스트를 생성한다 (26개 정수)
  - 파일로부터 각각의 행을 문자열로 읽는다
  - 문자열 문자를 소문자로 변환한다 (str 클래스의 lower() 메소드)
  - 문자열의 각각의 문자에 대해 소문자 문자인지 확인하고 맞으면 해당 개수 값을 증가한다
  - 결과를 출력한다

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

    counts = 26 * [0] # counts를 생성하고 초기화한다.
    for line in infile:
        # 각 문자를 세기 위한 countLetters 함수를 호출한다.
        countLetters(line.lower(), counts)

    # 결과를 출력한다.
    for i in range(len(counts)):
        if counts[i] != 0:
            print(chr(ord('a') + i) + ": " + str(counts[i])
              + "번 나타납니다.")

    infile.close() # 파일을 닫는다.

# 문자열의 각 문자의 횟수를 센다.
def countLetters(line, counts):
    for ch in line:
        if ch.isalpha():
            counts[ord(ch) - ord('a')] += 1

main() # main 함수를 호출한다.
```

웹에서 데이터 획득하기

- 웹 주소(URL)를 이용하여 데이터를 읽기
  - urlopen() 함수 이용
    - urllib.request 모듈에 정의된 함수

infile = urllib.request.urlopen(“http://www.yahoo.com/index.html”)

- URL에서 데이터를 읽고 출력하는 예제

```
import urllib.request
infile = urllib.request.urlopen("https://github.com/skang-koreatech/ScriptProgramming_2016-2/blob/master/README.md")
print(infile.read().decode())
```

예외 처리

- 존재하지 않는 파일을 read 혹은 append 모드로 열려고 하면 어떻게 될까?

- 현재 디렉토리에 filenotexist.txt 파일이 없는 경우

File = open(“filenotexist.txt”, “r”)

예외 처리

- 예외
  - 실행 시간에 발생하는 오류

```
>>> file = open("filenotexist.txt", "r")
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    file = open("filenotexist.txt", "r")
FileNotFoundError: [Errno 2] No such file or directory: 'filenotexist.txt'
>>>
```

예외 처리

- 예외 처리를 위한 문법
  - try: except: 문을 사용

```
try:
   # body
except <ExceptionType>:
   # exception handler
```

- body 부분
  - 예외(exception)가 발생할 수 있는 코드를 포함
- exception handler 부분
  - 예외 발생시 예외 타입이 except 다음의 exception type과 일치하면 exception handler 부분의 코드가 실행됨

예외 처리 예제

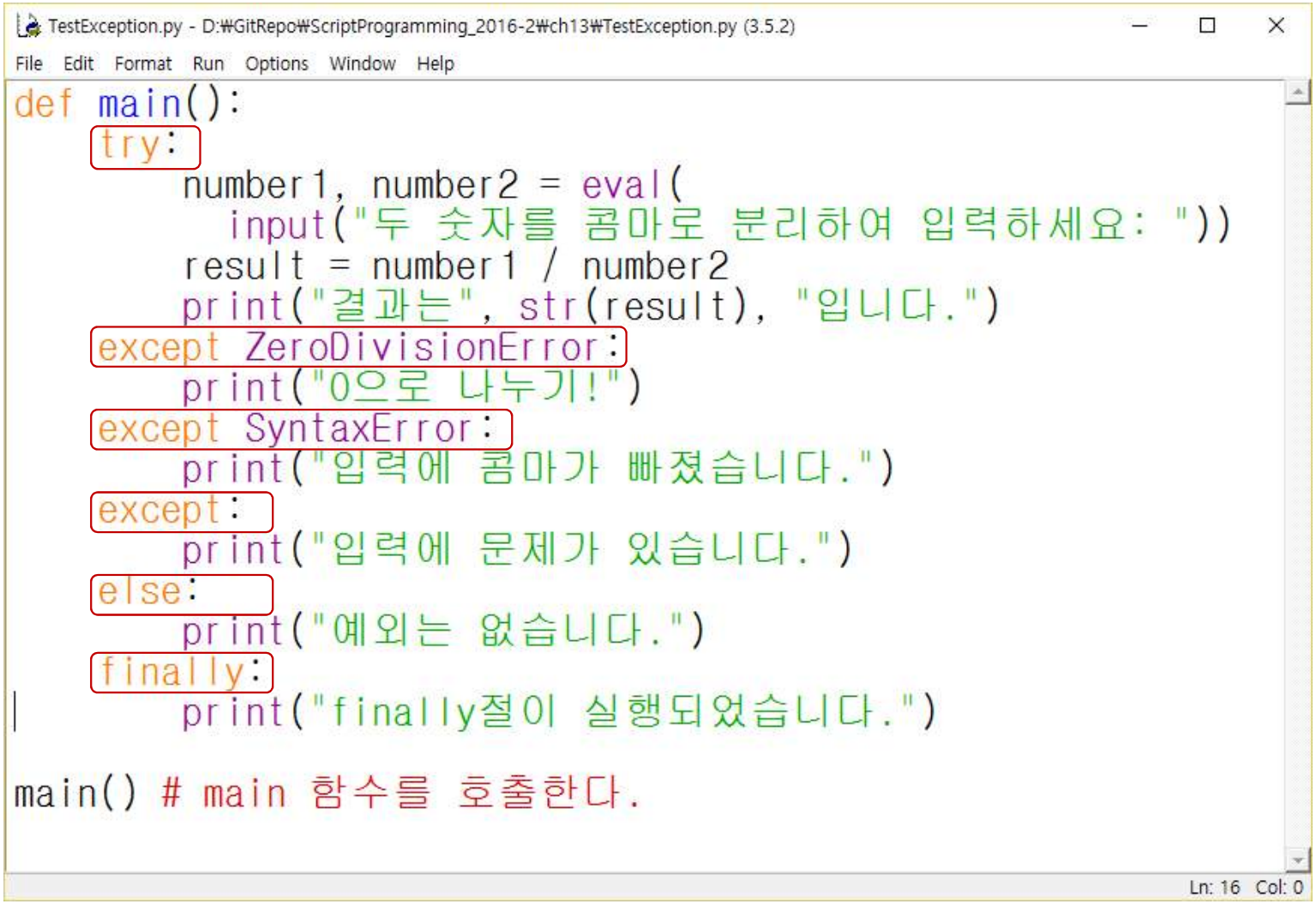
```
def main():
    while True:
        try:
            filename = input("파일명을 입력하세요: ").strip()
            infile = open(filename, "r") # 파일을 연다.
            break
        except IOError:
            print("파일 " + filename + " 이/가 존재하지 않습니다. 다시 시도하세요")

    print("파일을 읽기 모드로 열었습니다.")

main() # main 함수를 호출한다.
```

- while 루프 안에서 try 절(try와 except 사이 블록 내)의 명령문이 실행됨
- 만약 예외가 발생하지 않으면, break 문까지 실행되어 while 루프는 종료되고 except 절은 생략됨
- 만약 try 절을 실행하는 도중 예외가 발생하면, try 절의 나머지 부분은 생략됨
  - open에서 예외가 발생하면 break 명령문은 실행되지 않음
- 예외가 발생할 때, 발생한 예외가 except 키워드 뒤의 예외 이름과 일치하면 (예제의 경우 IOError) 해당 except 절이 실행되고 이후 명령문이 실행됨
- 예외 타입이 일치하지 않으면, 예외가 이 함수의 호출자에게 전달되고 해당 핸들러가 발견되지 않으면 처리되지 않은 예외로 오류 메시지가 출력되면서 프로그램 실행이 중단됨

예외 처리

- 복수 예외 처리
  - 여러 개의 예외 타입을 처리할 수 있도록 한 개 이상의 except 절을 포함할 수 있음
- else 절
  - 어떠한 예외도 발생하지 않은 경우 실행되는 명령문을 포함
- finally 절
  - 모든 상황에서 수행되어야 하는 마무리 기능을 정의하는 용도

예외 처리


TestException.py - D:₩GitRepo₩ScriptProgramming_2016-2₩ch13₩TestException.py (3.5.2)
File Edit Format Run Options Window Help
def main():
try:
number1, number2 = eval(
input("두 숫자를 콤마로 분리하여 입력하세요: "))
result = number1 / number2
print("결과는", str(result), "입니다.")
except ZeroDivisionError:
print("0으로 나누기!")
except SyntaxError:
print("입력에 콤마가 빠졌습니다.")
except:
print("입력에 문제가 있습니다.")
else:
print("예외는 없습니다.")
finally:
print("finally절이 실행되었습니다.")
main() # main 함수를 호출한다.
Ln: 16 Col: 0

예외 발생시키기

- 예외 발생
  - 함수가 오류를 감지하면, 함수는 적절한 예외 클래스로부터 객체를 생성하고 함수 호출자에게 예외를 전달

  raise ExceptionClass(“Something wrong.”)

  - 예

  # RuntimeError의 객체를 생성하고 예외를 발생함
  ex = RuntimeError(“잘못된 전달인자”)
  raise ex

예외 발생시키기

```python
import math

class Circle():
    def __init__(self, radius):
        super().__init__()
        self.setRadius(radius)

    def getRadius(self):
        return self.__radius

    def setRadius(self, radius):
        if radius < 0:
            raise RuntimeError("반지름이 음수")
        else:
            self.__radius = radius

    def getArea(self):
        return self.__radius * self.__radius * math.pi
```

```python
from CircleWithException import Circle

try:
    c1 = Circle(5)
    print("c1의 넓이는", c1.getArea(), "입니다.")
    c2 = Circle(-5)
    print("c2의 넓이는", c2.getArea(), "입니다.")
    c3 = Circle(0)
    print("c3의 넓이는", c3.getArea(), "입니다.")
except RuntimeError as ex:
    print("유효하지 않은 반지름입니다.", ex)
```

RuntimeError 예외 객체를 ex라는 이름으로 참조하겠다는 의미