파이썬 프로그래밍

예외 처리

한국기술교육대학교
온라인평생교육원

▣ 파이썬 예외의 종류

1. 예외 발생 예제 보기

- 구문 에러 (Syntax Error)
  - 문법적 에러
  - 이클립스 등의 통합개발환경 도구에서는 자동으로 실행 전에 구문 에러를 체크 해 줌
  - 파이썬은 상대적으로 언어적 문법이 간단하기 때문에 구문 자체의 에러 발생 비율이 낮거나 다른 도구를 사용하여 완벽하게 제거할 수 있음
- 예외 (Exception)
  - 구문 에러는 없으나 프로그램 실행 중 더 이상 진행 할 수 없는 상황

▪문법 에러는 이클립스에서 바로 체크함

- 예외 발생 예제 1: 정의되지 않은 변수 사용하기
  - NameError

```
4 + spam*3
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-6b1dfe582d2e> in <module>()
----> 1 4 + spam*3

NameError: name 'spam' is not defined
```

▪구문적인 에러는 없지만 spam 단어 미리 선언 X

▪선언이 X → NameError 발생

▪Error 이므로 예외보다는 강한 예외

▣ 파이썬 예외의 종류

1. 예외 발생 예제 보기

- 예외 발생 예제 2: 0으로 숫자 나누기
  - ZeroDivisionError

```
a = 10
b = 0
c = a / b
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-d055e09331c6> in <module>()
      1 a = 10
      2 b = 0
----> 3 c = a / b

ZeroDivisionError: integer division or modulo by zero
```

▪ZeroDivisionError 발생 → 0으로 나누었기 때문

▣ 파이썬 예외의 종류

1. 예외 발생 예제 보기

- [note] 예외가 발생하면 프로그램은 바로 종료된다.

```
def division():
    for n in range(0, 5):
        print 10.0 / n

division()
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-48-9618f69a4b0e> in <module>()
      3         print 10.0 / n
      4
----> 5 division()

<ipython-input-48-9618f69a4b0e> in division()
      1 def division():
      2     for n in range(0, 5):
----> 3         print 10.0 / n
      4
      5 division()

ZeroDivisionError: float division by zero
```

▪예외가 발생하면 프로그램이 종료됨

▣ 파이썬 예외의 종류

1. 예외 발생 예제 보기

- 예외 발생 예제 3: 문자열과 숫자 더하기
  - TypeError

```
'2' + 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-4c6dd5170204> in <module>()
----> 1 '2' + 2

TypeError: cannot concatenate 'str' and 'int' objects
```

▪TypeError 발생

▪문자열 + 수치형자료 ← 두 개의 타입이 달라서 연결 XX

- 예외 발생 예제 4: 참조 범위를 넘어서 인덱스 사용
  - IndexError

```
l = [1, 2]
print l[2]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-3-e36f806c83e5> in <module>()
      1 l = [1, 2]
----> 2 print l[2]

IndexError: list index out of range
```

▪l 리스트는 리스트를 두 개(0, 1) 가짐

▪l 인덱싱을 2를 하면 해당 내용이 없으므로 IndexError 발생

▣ 파이썬 예외의 종류

1. 예외 발생 예제 보기

- 예외 발생 예제 5: 등록되지 않은 키로 사전 검색
  - KeyError

```
d = {"a": 1, "b": 2}
print d['c']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-4-c5d228d3ae3c> in <module>()
      1 d = {"a": 1, "b": 2}
----> 2 print d['c']

KeyError: 'c'
```

▪KeyError → 사전에서 발생할 수 있는 예외

▪존재하지 않는 c를 검색하면 KeyError 발생

- 예외 발생 예제 6: 있지도 않은 파일을 열려고 할 때
  - IOError

```
a = open('aaa.txt')]
```

```
---------------------------------------------------------------------------
IOError                                   Traceback (most recent call last)
<ipython-input-6-971bbb0a7949> in <module>()
----> 1 a = open('aaa.txt')

IOError: [Errno 2] No such file or directory: 'aaa.txt'
```

▪이클립스에 수행되고 있는 파일이 없음

▪없는 파일을 열어서 IOError 발생

▪IOError : Input, Output (입, 출력) Error

▣ 파이썬 예외의 종류

2. 내장 예외의 종류

- 예외 클래스의 계층 구조 ([참고] https://docs.python.org/2/library/exceptions.html)

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StandardError
    |    +-- BufferError
    |    +-- ArithmeticError
    |    |    +-- FloatingPointError
    |    |    +-- OverflowError
    |    |    +-- ZeroDivisionError
    |    +-- AssertionError
    |    +-- AttributeError
    |    +-- EnvironmentError
    |    |    +-- IOError
    |    |    +-- OSError
    |    |         +-- WindowsError (Windows)
    |    |         +-- VMSError (VMS)
    |    +-- EOFError
    |    +-- ImportError
    |    +-- LookupError
    |    |    +-- IndexError
    |    |    +-- KeyError
    |    +-- MemoryError
    |    +-- NameError
    |    |    +-- UnboundLocalError
    |    +-- ReferenceError
    |    +-- RuntimeError
    |    |    +-- NotImplementedError
    |    +-- SyntaxError
    |    |    +-- IndentationError
    |    |         +-- TabError
    |    +-- SystemError
    |    +-- TypeError
```

▣ 파이썬 예외의 종류

2. 내장 예외의 종류

```
|    +-- ValueError
     |         +-- UnicodeError
     |              +-- UnicodeDecodeError
     |              +-- UnicodeEncodeError
     |              +-- UnicodeTranslateError
     +-- Warning
          +-- DeprecationWarning
          +-- PendingDeprecationWarning
          +-- RuntimeWarning
          +-- SyntaxWarning
          +-- UserWarning
          +-- FutureWarning
      +-- ImportWarning
      +-- UnicodeWarning
      +-- BytesWarning
```

▪Baseexception → Baseclass = Superclass

▪상위에 있는 Baseexception을 상속 → SystemExit……

파이썬 프로그래밍

예외 처리

한국기술교육대학교
온라인평생교육원

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

- 예외가 발생할 수 있는 상황을 예상하여 예외 발생 상황을 전체 코드 흐름을 함께 제어할 수 있다.
- try/except/else/finally 절

- 구문

```
try:
    (예외 발생 가능한) 일반적인 수행문들
except Exception:
    예외가 발생하였을 때 수행되는 문들
else:
    예외가 발생하지 않았을 때 수행되는 문들
finally:
    예외 발생 유무와 관계없이 무조건 수행되는 문들
```

▪except 키워드 뒤 → 발생할 수 있는 exception 내용(클래스이름) 적음
▪수행문에서 Zerodivision 발생 예상 → exception에 Zerodivision 삽입
▪exception 밑에 존재하는 클래스에 해당하는 예외 전부 catch
▪try가 나오면 밑으로 finally 구문은 무조건 수행
▪else와 finally는 optional한 절

```
try:
    print 1.0 / 0.0
except ZeroDivisionError:
    print 'zero division error!!!'
```

```
zero division error!!!
```

▪try는 새로운 구문이 시작되어야 하기 때문에 콜론(:) 삽입
▪0으로 나누고 있기 때문에 ZeroDivisionError 발생
▪try, except 절 사용 X → 빨간 Error 발생 (프로그램 비정상적 수행)
▪try 절을 수행하면 프로그램이 정상적으로 수행되어 종료 X

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

- 예외 처리를 하면 예외 발생시 프로그램 종료가 되지 않는다.

```
def division():
    for n in range(0, 5):
        try:
            print 10.0 / n
        except ZeroDivisionError, msg:
            print msg

division()
```

```
float division by zero
10.0
5.0
3.33333333333
2.5
```

▪msg 변수: ZeroDivisionError를 정의한 사람이 발생될 때 주는 메세지
▪예외 발생으로 프로그램이 종료되어 루프 진행 X
▪예외 처리를 하면 프로그램이 종료되지 않고 계속 진행됨

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

- else: 구문은 except: 구문없이 사용 못한다.

```
def division():
    for n in range(0, 5):
        try:
            print 10.0 / n
        else:
            print "Success"

division()
```

```
  File "<ipython-input-49-e039cc968e23>", line 5
    else:
       ^
SyntaxError: invalid syntax
```

▪else는 except 없이는 사용 X

▪else가 있는데 except이 없는건 구문적 error → 빨간 X, 밑줄 제시

▪finally는 관계없이 적을 수 있음

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

- 상황에 따라서는 에러와 함께 따라오는 정보를 함께 받을 수도 있다.

```
try:
    spam()
except NameError, msg:
    print 'Error -', msg
```

```
Error - name 'spam' is not defined
```

```
try:
    spam()
except NameError as msg:
    print 'Error -', msg
```

```
Error - name 'spam' is not defined
```

▪msg → NameError의 메시지가 구현됨
▪콤마(,) 대신 as 사용하여 메시지 받을 수 있음

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

- try 절 안에서 간접적으로 호출한 함수의 내부 예외도 처리할 수 있다.

```
def zero_division():
    x = 1 / 0

try:
    zero_division()
except ZeroDivisionError, msg:
    print 'zero division error!!! -', msg
```

```
zero division error!!! -  integer division or modulo by zero
```

▪0으로 나누게 되니까 예외 발생
▪함수에는 예외 발생상황이 안보이지만 함수 수행 시 예외 발생 가능

```
def zero_division():
    x = 1 / 0

try:
    zero_division()
except ZeroDivisionError as msg:
    print 'zero division error!!! -', msg
```

```
zero division error!!! - integer division or modulo by zero
```

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

- except 뒤에 아무런 예외도 기술하지 않으면 모든 예외에 대해 처리된다.

```
try:
    spam()
    print 1.0 / 0.0
except:
    print 'Error'
```

```
Error
```

▪NameError, NameEroor 둘 다 발생이 되는 상황
▪except 뒤에 아무런 내용 X → 모든 예외 catch
▪어떤 예외가 발생했는지는 알 수 없음

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

- 여러 예외들 각각에 대해 except 절을 다중으로 삽입할 수 있다.

```
b = 0.0
name = 'aaa.txt'
try:
    print 1.0 / b
    spam()
    f = open(name, 'r')
    '2' + 2
except NameError:
    print 'NameError !!!'
except ZeroDivisionError:
    print 'ZeroDivisionError !!!'
except (TypeError, IOError):
    print 'TypeError or IOError !!!'
else:
    print 'No Exception !!!'
finally:
    print 'Exit !!!'
```

```
ZeroDivisionError !!!
Exit !!!
```

▪ZeroDivisionError, NameError, IOError, TypeError 발생 가능

▪튜플 형태로 a or b로 묶어서 정의 가능

▪튜플은 가로가 없어도 사용 가능

▪각각의 경우에 따라서 제어를 해주고 싶으면 except 여러 번 사용 가능

▪예외가 발생하지 않아도 finally 이 부분은 항상 수행

▣ 예외 처리 방법

1. try/except/else/finally 절 사용하기

• 파일에서 숫자를 읽어와서 읽은 숫자로 나누기를 하는 예제
- 꼼꼼한 예외 처리 예제

```
import os
print os.getcwd()
filename = 't.txt'

try:
    f = open(filename, 'r')
except IOError, msg:
    print msg
else:
    a = float(f.readline())
    try:
        answer = 1.0 / a
    except ZeroDivisionError, msg:
        print msg
    else:
        print answer
    finally:
        print "Finally!!!"
        f.close()
```

```
/Users/yhhan/git/python-e-learning
1.0
Finally!!!
```

▪os.getcwd() → 현재 디렉토리를 알려줌
▪f = open이라는 구문은 try, except 사이에 항상 들어와야 함
▪file 이름을 잘못 적거나 파일이 없을 수도 있으므로 미리 방지 가능
▪보통 파일 안에 어떤 것이 있는지 모르므로 방지를 위해 사용

▣ 예외 처리 방법

2. 같은 부류의 예외 다 잡아내기

- 예외 클래스들은 상속에 의한 계층 관계를 지니고 있기 때문에 이를 이용하면 여러 예외들을 한꺼번에 잡을 수 있다.
- 예를 들어, ArithmeticError의 하위 클래스로서 FloatingPointError, OverflowError, ZeroDivisionError가 존재하기 때문에 이들 하위 클래스 예외가 발생하였을 경우 ArithmeticError로서 잡아낼 수 있다.

```
def dosomething():
    a = 1/0

try:
    dosomething()
except ArithmeticError:
    print "ArithmeticException occured"
```

```
ArithmeticException occured
```

▪ArithmeticError의 하위클래스로 같은 부류

▪except 옆에 상위 클래스 적으면 하위 클래스의 Error 모두 catch

▣ 예외 처리 방법

2. 같은 부류의 예외 다 잡아내기

- 예외가 임의의 except에 의해 잡히면 다른 except에 의해서는 잡히지 않는다.

```
def dosomething():
    a = 1/0

try:
    dosomething()
except ZeroDivisionError:  # ZeroDivisionError는 이곳에서 잡힌다.
    print "ZeroDivisionError occured"
except ArithmeticError:    # FloatingPointError, OverflowError는 이곳에서 잡힌다.
    print "ArithmeticException occured"
```

```
ZeroDivisionError occured
```

▪하위 클래스 예외를 정의하면, 상위 클래스 예외 정의 시 같이 적용 X

▣ 예외 처리 방법

2. 같은 부류의 예외 다 잡아내기

```
def dosomething():
    a = 1/0

try:
    dosomething()
except ArithmeticError:
    print "ArithmeticException occured"
except ZeroDivisionError:  # 이곳에서 ZeroDivisionError는 잡히지 않는다.
                                ==> 잘못된 코드
    print "ZeroDivisionError occured"
```

```
ArithmeticException occured
```

▪하위 클래스가 상위 클래스 안에 있는 것으로 상위클래스만 수행됨
▪아래 따로 정의한 하위 클래스 내용은 절대 수행 X

파이썬 프로그래밍

예외 처리

한국기술교육대학교
온라인평생교육원

▣ 예외 발생

1. Raise로 예외 발생하기

- 예외를 특정 상황 조건에서 raise 키워드를 통해 발생시킬 수 있다.
- 아래 예는 시퀀스 형 클래스를 설계할 때 인덱싱을 구현하는 __getitem__ 메소드에서 인덱스가 범위를 넘을 때 IndexError를 발생시킨다.

▪raise 키워드 → 예외를 임의로 발생시킬 수 있는 키워드
▪self.n = n → 인스턴스 객체가 내부적으로 n 값을 지님
▪__getitem__ → 인덱싱 연산에 대응되는 연산
▪ raise IndexError → 이미 존재하는 Error → 이것을 발생시키겠다
▪10이 되는 순간 if 절 조건을 만족하여 IndexError 발생
▪for~in 구문은 IndexError가 발생할 때까지만 수행
▪IndexError를 잡고 싶으면 try, except절 사용

▣ 예외 발생

1. Raise로 예외 발생하기

```
class SquareSeq:
    def __init__(self, n):
        self.n = n
    def __getitem__(self, k):
        if k >= self.n or k < 0 :
            raise IndexError # 첨자 범위를 벗어나면 IndexError 예외를 발생시킴
        return k * k
    def __len__(self):
        return self.n

s = SquareSeq(10)
print s[2], s[4]
for x in s:  # IndexError가 발생하는 시점까지 반복한다
    print x,
print s[20]  # 첨자 범위가 넘었다
```

```
4 16
0 1 4 9 16 25 36 49 64 81
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-6-f362c0e14e3a> in <module>()
     13 for x in s:  # IndexError가 발생하는 시점까지 반복한다
     14     print x,
---> 15 print s[20]  # 첨자 범위가 넘었다

<ipython-input-6-f362c0e14e3a> in __getitem__(self, k)
      4     def __getitem__(self, k):
      5         if k >= self.n or k < 0 :
----> 6             raise IndexError # 첨자 범위를 벗어나면 IndexError 예외를 발생시킴
      7         return k * k
      8     def __len__(self):

IndexError:
```

▣ 예외 발생

2. 사용자 클래스 예외 정의 및 발생시키기

- 사용자 정의 예외 클래스를 구현하는 일반적인 방법은 Exception 클래스를 상속 받아 구현한다.
  - Exception 클래스의 서브 클래스 중 하나를 상속 받아도 된다.
- 사용자 정의 예외 발생 방법
  - 내장 예외 발생 방법과 동일하게 raise [클래스의 인스턴스] 와 같이 해당 예외 클래스의 인스턴스를 던진다.
- 사용자 정의 예외를 잡는 방법
  - except [클래스 이름] 과 같이 해당 예외 클래스 이름을 사용한다.
- 아래 예에서 except Big이 잡는 예외는 Big과 Small 이다.
  - 이유: Small은 Big의 하위 클래스이기 때문

▪사용자 정의 예외 클래스 → 파이썬이 내장하고 있는 것 이외의 클래스

▣ 예외 발생

2. 사용자 클래스 예외 정의 및 발생시키기

```
class Big(Exception):
    pass

class Small(Big):
    pass

def dosomething1():
    x = Big()
    raise x

def dosomething2():
    raise Small()

for f in (dosomething1, dosomething2):
    try:
        f()
    except Big:
        print "Exception occurs!"
```

```
Exception occurs!
Exception occurs!
```

▪슈퍼클래스가 Exception으로 예외 클래스 정의한 것 사용 가능
▪예외를 발생시키는 방법 = 예외를 정의하는 방법
▪내장예외 발생시키는 방법 = 내장예외 처리하는 방법
▪Big, Small은 학습자가 스스로 정의하는 예외 클래스
▪함수 호출 후 보니, 함수 안 raise 있으면 예외 발생 → 예외 catch

▣ 예외 발생

3. 예외값 전달하기

- raise 키워드 뒤에 예외와 함께, 추가 메시지를 함께 던질 수 있다.

```
def f():
    raise Exception, 'message!!!'

try:
    f()
except Exception, a:
    print a
```

```
message!!!
```

▪raise란 키워드 사용 방법 → 예외 클래스, 메세지

- 생성자 안에 넣어준 에러 메시지는 except 키워드 사용시에 두 번째 인자로 해당 메시지를 받을 수 있다.

```
a = 10
b = 0
try:
    if b == 0:
        raise ArithmeticError('0으로 나누고 있습니다.')
    a / b
except ArithmeticError, v:
    print v
```

```
0으로 나누고 있습니다.
```

▪ArithmeticError는 파이썬에 조재하는 에러
▪예외(클래스)를 통해서 생성자를 보는 것
▪생성자를 부르면서 문자열로 넣어주는 인자는 , 뒤로 받아낼 수 있음