예외 처리(Exception Handling)

By 윤명근 / 박수현

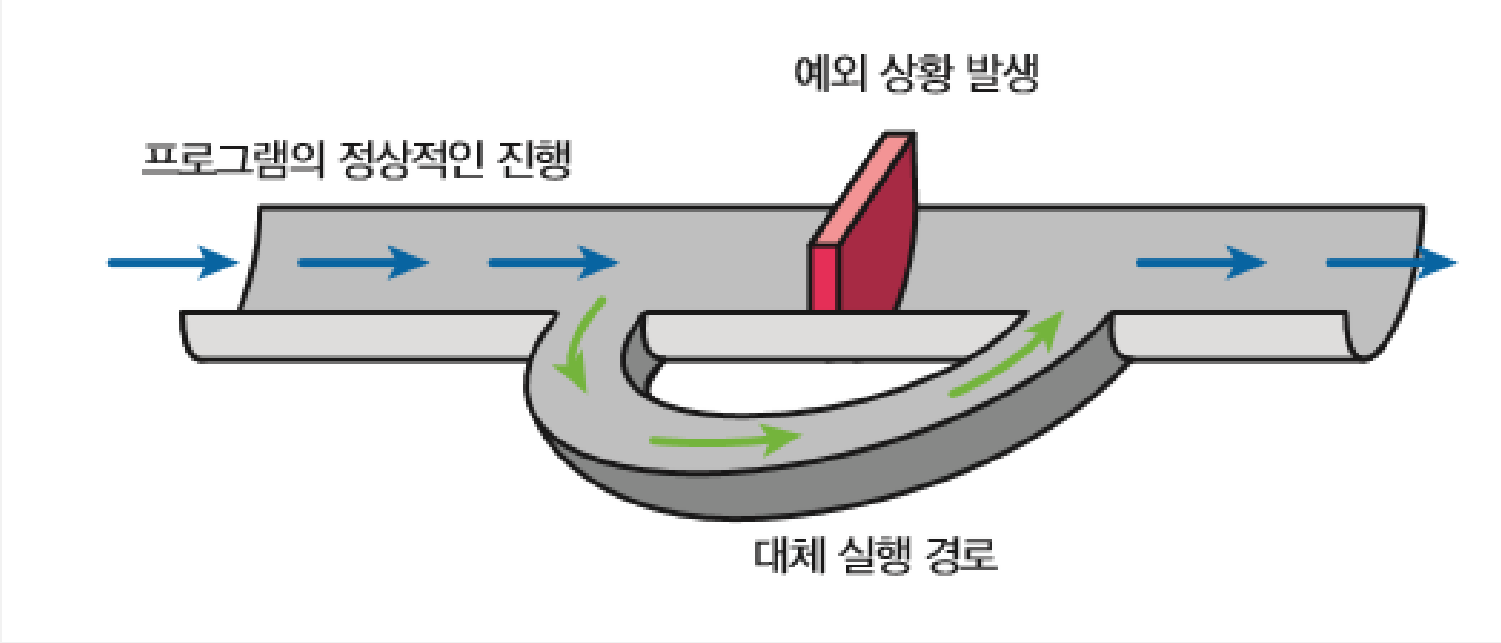
국민대학교 소프트웨어학부

수업목표

- 예외 처리 필요성
- 예외 처리 문법
- Python 제공 예외 처리

예외 처리의 개념

- 오류가 발생 했을 때 오류를 사용자에게 알려주고 모든 데이터를 저장하게 한 후에 사용자가 우아하게(gracefully) program을 종료할 수 있도록 하는 것이 바람직



예외 처리 필요성

- Program 수행 도중 발생하는 예외 상황 처리
  - 0으로 나누는 경우 (division by 0)
  - 존재하지 않는 파일을 읽으려는 경우
  - list, tuple등 나열형 data의 index 범위 밖을 접근하는 경우

    (ex) list index out of range)
- 예외 처리를 하지 않으면 전체 program이 갑자기 종료 또는 비정상 동작하게 됨

예외 처리

- 오류 !

```
>>> (x, y) = (2, 0)
>>> z = x / y
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    z = x / y
ZeroDivisionError: division by zero
>>>
```

Python에서의 예외 처리

전체적인 구조

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

예제

exception_1.py

File Edit Format Run Options Window Help

```
(x,y) = (2,0)
print('₩n1) x =', x, ', y = ', y)

try:
    print('₩n2) try 입구')
    z = y/x
    print('3) z = y/x  : ', z)

    print('₩n4) Divided by zero 에러 발생')
    z = x/y

    print('5) try 출구')

except ZeroDivisionError as e:
    print ("₩n6) e =", e)
    print ("7) 0으로 나누는 예외 발생₩n")
```

```
1) x = 2 , y =  0

2) try 입구
3) z = y/x  :  0.0

4) Divided by zero 에러 발생

6) e = division by zero
7) 0으로 나누는 예외 발생
```

예외 처리 문법

- try except

```
try:
    코드 블록
except [예외타입 [as 예외변수]]
    예외 처리 코드
[else:
    예외가 발생하지 않은 경우 수행할 코드 블록
finally:
    예외가 발생하든 하지 않든 try 블록 이후 수행할 코드블록]
```

  - except문 처리 방식
    - except 예외타입:
      - 특정 타입 의 예외를 처리하는 경우
    - except:
      - 모든 타입의 예외를 처리하는 경우

11example1.py

File Edit Format Run Options Window Help

```
def divide(m, n):
    try:
        print("d-1) in divide()..")
        result = m/n
    except ZeroDivisionError:
        print("d-2) 0으로 나눌 수 없습니다.")
    except:
        print("d-3) ZeroDivisionError 이외의 예외 발생")
    else:
        print("d-4) result = ", result)
        return result
    finally:
        print ("d-5) finally 부분의 반드시 실행되는 부분입니다.")
        print ("d-6) 나눗셈 연산입니다")

#main
print("0) main ")

if __name__ == "__main__":
    print("₩n1) call divide(3,2)")
    res = divide(3,2)
    print("2) res = ",res)

    print("₩n3) call divide(10,0)")
    res = divide(10,0)
    print("4) res = ",res)

    print("₩n5) call divide(200,3)")
    res = divide(200,3)
    print("6) res = ",res)
```

```
0) main

1) call divide(3,2)
d-1) in divide()..
d-4) result =  1.5
d-5) finally 부분의 반드시 실행되는 부분입니다.
d-6) 나눗셈 연산입니다
2) res =  1.5

3) call divide(10,0)
d-1) in divide()..
d-2) 0으로 나눌 수 없습니다.
d-5) finally 부분의 반드시 실행되는 부분입니다.
d-6) 나눗셈 연산입니다
4) res =  None

5) call divide(200,3)
d-1) in divide()..
d-4) result =  66.66666666666667
d-5) finally 부분의 반드시 실행되는 부분입니다.
d-6) 나눗셈 연산입니다
6) res =  66.66666666666667
```

exception_2.py

File Edit Format Run Options Window Help

```
class MyError(Exception):

    def __init__(self, msg):
        print("i-1) in init(), msg = ", msg)
        self.msg = msg
        print("i-2) self.msg = ", self.msg)

    def __str__(self):
        print("₩ns-1) in str(), self.msg = ", self.msg)
        return self.msg


def say_nick(nick):
    print("s-1) in say_nick()..")

    if nick == "바보":
            print("s-2) raise MyError()..")
            raise MyError("허용되지 않은 별명입니다.")

    print("s-3) nick = ", nick)


#main
print("₩n0) main. ")

if __name__ == "__main__":

    try:
        print("₩n1) call say_nick(₩"천사₩")")
        say_nick("천사")

        print("₩n2) call say_nick(₩"바보₩")")
        say_nick("바보")

    except MyError as e:
        print("₩n3) 사용자가 정의한 Exception handler 호출")
        print("4) e = ", e)

    finally:
        print ("₩n5) finally 부분으로  반드시 실행되는 부분입니다.")
```

http://blog.naver.com/shumin/220811661868 를 수정

```
0) main.

1) call say_nick("천사")
s-1) in say_nick()..
s-3) nick =  천사

2) call say_nick("바보")
s-1) in say_nick()..
s-2) raise MyError()..
i-1) in init(), msg =  허용되지 않은 별명입니다.
i-2) self.msg =  허용되지 않은 별명입니다.

3) 사용자가 정의한 Exception handler 호출
4) e =
s-1) in str(), self.msg =  허용되지 않은 별명입니다.
허용되지 않은 별명입니다.

5) finally 부분으로  반드시 실행되는 부분입니다.
```

Python 제공 예외 처리

- https://docs.python.org/ko/3/library/exceptions.html#exception-hierarchy

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      |    +-- ModuleNotFoundError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
      +-- OSError
      |    +-- BlockingIOError
      |    +-- ChildProcessError
      |    +-- ConnectionError
      |    |    +-- BrokenPipeError
      |    |    +-- ConnectionAbortedError
      |    |    +-- ConnectionRefusedError
      |    |    +-- ConnectionResetError
      |    +-- FileExistsError
      |    +-- FileNotFoundError
      |    +-- InterruptedError
      |    +-- IsADirectoryError
      |    +-- NotADirectoryError
      |    +-- PermissionError
      |    +-- ProcessLookupError
      |    +-- TimeoutError
      +-- ReferenceError
      +-- RuntimeError
      |    +-- NotImplementedError
      |    +-- RecursionError
      +-- SyntaxError
      |    +-- IndentationError
      |         +-- TabError
      +-- SystemError
      +-- TypeError
      +-- ValueError
      |    +-- UnicodeError
      |         +-- UnicodeDecodeError
      |         +-- UnicodeEncodeError
      |         +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
           +-- ResourceWarning
```