

UNIT 38

예외 처리 사용하기



파이썬 코딩 도장

따라하기, 연습하기, 심사하기로 배우는 파이썬



38 예외 처리 사용하기



» 예외 처리 사용하기

- 예외(exception)란 코드를 실행하는 중에 발생한 에러를 뜻함
- 다음과 같이 10을 어떤 값으로 나누는 함수 `ten_div`가 있을 때 인수에 따라 정상으로 동작하기도 하고 에러가 나기도 함

```
>>> def ten_div(x):  
...     return 10 / x  
...
```

```
>>> ten_div(2)  
5.0
```

38 예외 처리 사용하기



» 예외 처리 사용하기

- 0을 넣으면 실행하는 중에 예외가 발생함
- 이런 상황을 예외라고 하는데 여기서는 어떤 숫자를 0으로 나누어서 ZeroDivisionError 예외가 발생함

```
>>> ten_div(0)
Traceback (most recent call last):
  File "<pyshell#121>", line 1, in <module>
    ten_div(0)
  File "<pyshell#119>", line 2, in ten_div
    return 10 / x
ZeroDivisionError: division by zero
```

- ZeroDivisionError뿐만 아니라 지금까지 만난 AttributeError, NameError, TypeError 등 다양한 예외들도 모두 예외임

3

38.1 try except로 사용하기



» try except로 사용하기

```
try:
    실행할 코드
except:
    예외가 발생했을 때 처리하는 코드
```

- 이제 숫자를 0으로 나누었을 때 발생하는 예외를 처리해보자

try_except.py

```
try:
    x = int(input('나눌 숫자를 입력하세요: '))
    y = 10 / x
    print(y)
except: # 예외가 발생했을 때 실행됨
    print('예외가 발생했습니다.')
```

실행 결과

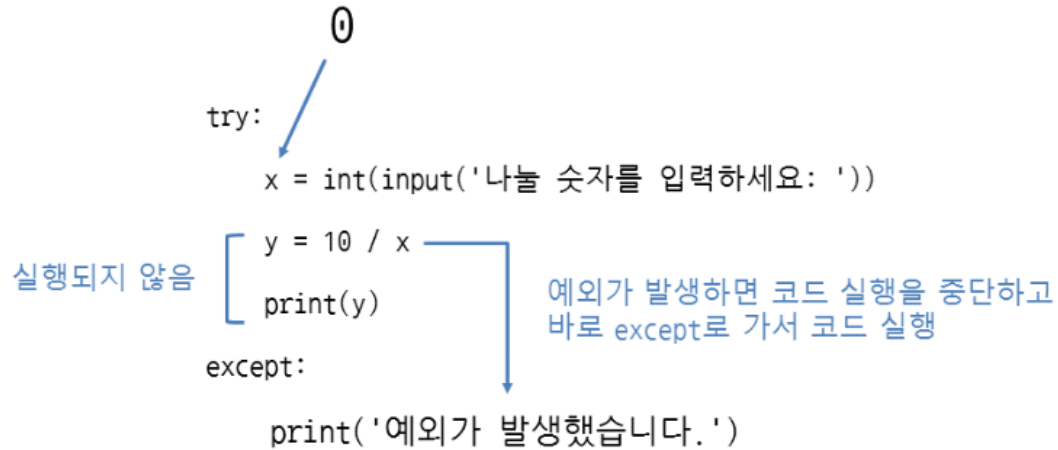
```
나눌 숫자를 입력하세요: 0 (입력)
예외가 발생했습니다.
```

4

38.1 try except로 사용하기



▼ 그림 38-1 예외 발생과 except



5

38.1 try except로 사용하기



» try except로 사용하기

나눌 숫자를 입력하세요: 2 (입력)
5.0

- try의 코드가 에러 없이 잘 실행 되면 except의 코드는 실행되지 않고 그냥 넘어감
- try의 코드에서 에러가 발생했을 때만 except의 코드가 실행됨

6

38.1 try except로 사용하기



» 특정 예외만 처리하기

```
try:
    실행할 코드
except 예외이름:
    예외가 발생했을 때 처리하는 코드
```

- 다음과 같이 정수 두 개를 입력받아서 하나는 리스트의 인덱스로 사용하고, 하나는 나누는 값으로 사용함
- except를 두 개 사용하고 각각 ZeroDivisionError와 IndexError를 지정함

try_except_exception.py

```
y = [10, 20, 30]

try:
    index, x = map(int, input('인덱스와 나눌 숫자를 입력하세요: ').split())
    print(y[index] / x)
except ZeroDivisionError:  # 숫자를 0으로 나눠서 예러가 발생했을 때 실행됨
    print('숫자를 0으로 나눌 수 없습니다.')
except IndexError:        # 범위를 벗어난 인덱스에 접근하여 예러가 발생했을 때 실행됨
    print('잘못된 인덱스입니다.')
```

7

38.1 try except로 사용하기



» 특정 예외만 처리하기

```
인덱스와 나눌 숫자를 입력하세요: 2 0 (입력)
숫자를 0으로 나눌 수 없습니다.
```

```
인덱스와 나눌 숫자를 입력하세요: 3 5 (입력)
잘못된 인덱스입니다.
```

- y = [10, 20, 30]은 요소가 3개 들어있는 리스트임
- 인덱스에 3을 지정하면 범위를 벗어나게 됨
- except IndexError:의 처리 코드가 실행됨

8

38.1 try except로 사용하기



» 예외의 에러 메시지 받아오기

```
try:
    실행할 코드
except 예외 as 변수:
    예외가 발생했을 때 처리하는 코드
```

- 앞에서 만든 코드의 except에 as e를 넣음
- 보통 예외(exception)의 e를 따서 변수 이름을 e로 지음

38.1 try except로 사용하기



» 예외의 에러 메시지 받아오기

try_except_as.py

```
y = [10, 20, 30]

try:
    index, x = map(int, input('인덱스와 나눌 숫자를 입력하세요: ').split())
    print(y[index] / x)
except ZeroDivisionError as e:
    print('숫자를 0으로 나눌 수 없습니다.', e)  # as 뒤에 변수를 지정하면 에러를 받아옴
except IndexError as e:
    print('잘못된 인덱스입니다.', e)  # e에 저장된 에러 메시지 출력
```

실행 결과

```
인덱스와 나눌 숫자를 입력하세요: 2 0 (입력)
숫자를 0으로 나눌 수 없습니다. division by zero
```

실행 결과

```
인덱스와 나눌 숫자를 입력하세요: 3 5 (입력)
잘못된 인덱스입니다. list index out of range
```

38.1 try except로 사용하기



» 예외의 에러 메시지 받아오기

- 2 0, 3 5처럼 예외가 발생하는 숫자를 넣어보면 해당 예외에 해당하는 에러 메시지가 출력됨
- 예외가 여러 개 발생하더라도 먼저 발생한 예외의 처리 코드만 실행됨(또는, 예외

```
except Exception as e: # 모든 예외의 에러 메시지를 출력할 때는 Exception을 사용
    print('예외가 발생했습니다.', e)
```

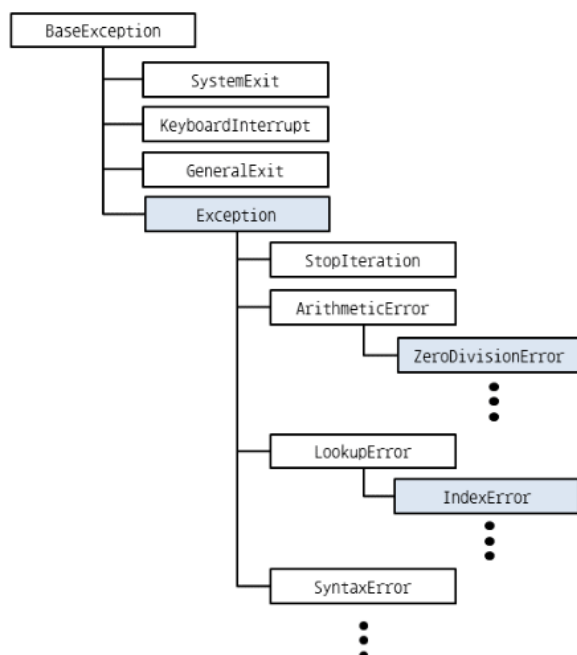
- 예외 처리는 에러가 발생하더라도 스크립트의 실행을 중단시키지 않고 계속 실행하고자 할 때 사용함

11

38.1 try except로 사용하기



▼ 그림 38-2 파이썬 예외 계층도



12

38.2 else와 finally 사용하기



» else와 finally 사용하기

- 다음과 같이 else는 except 바로 다음에 와야 하며 except를 생략할 수 없음

```
try:
    실행할 코드
except:
    예외가 발생했을 때 처리하는 코드
else:
    예외가 발생하지 않았을 때 실행할 코드
```

13

38.2 else와 finally 사용하기



» else와 finally 사용하기

- 그럼 10을 입력된 숫자로 나누고 예외가 발생하지 않으면 계산 결과를

try_except_else.py

```
try:
    x = int(input('나눌 숫자를 입력하세요: '))
    y = 10 / x
except ZeroDivisionError:    # 숫자를 0으로 나뉘서 에러가 발생했을 때 실행됨
    print('숫자를 0으로 나눌 수 없습니다.')
else:                       # try의 코드에서 예외가 발생하지 않았을 때 실행됨
    print(y)
```

나눌 숫자를 입력하세요: 2 (입력)
5.0

나눌 숫자를 입력하세요: 0 (입력)
숫자를 0으로 나눌 수 없습니다.

14

38.2 else와 finally 사용하기

» 예외와는 상관없이 항상 코드 실행하기

- finally는 except와 else를 생략할 수 있음

```
try:
    실행할 코드
except:
    예외가 발생했을 때 처리하는 코드
else:
    예외가 발생하지 않았을 때 실행할 코드
finally:
    예외 발생 여부와 상관없이 항상 실행할 코드
```

try_except_else_finally.py

```
try:
    x = int(input('나눌 숫자를 입력하세요: '))
    y = 10 / x
except ZeroDivisionError: # 숫자를 0으로 나눠서 에러가 발생했을 때 실행됨
    print('숫자를 0으로 나눌 수 없습니다.')
else: # try의 코드에서 예외가 발생하지 않았을 때 실행됨
    print(y)
finally: # 예외 발생 여부와 상관없이 항상 실행됨
    print('코드 실행이 끝났습니다.')
```

15

38.2 else와 finally 사용하기

» 예외와는 상관없이 항상 코드 실행하기

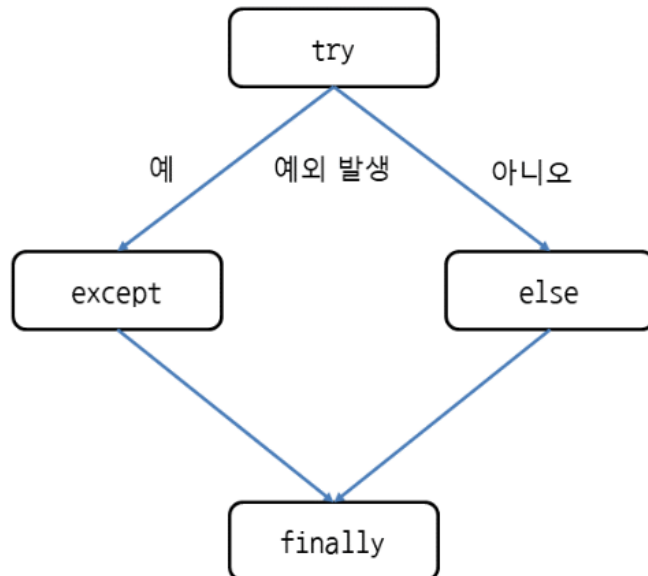
```
나눌 숫자를 입력하세요: 2 (입력)
5.0
코드 실행이 끝났습니다.
```

```
나눌 숫자를 입력하세요: 0 (입력)
숫자를 0으로 나눌 수 없습니다.
코드 실행이 끝났습니다.
```

16

38.2 else와 finally 사용하기

▼ 그림 38-3 try, except, else, finally의 실행 과정



17

38.3 예외 발생시키기

» 예외 발생시키기

- raise 예외('에러메시지')

try_except_raise.py

```

try:
    x = int(input('3의 배수를 입력하세요: '))
    if x % 3 != 0:
        raise Exception('3의 배수가 아닙니다.')
    print(x)
except Exception as e:
    print('예외가 발생했습니다.', e)
  
```

실행 결과

3의 배수를 입력하세요: 5 (입력)
예외가 발생했습니다. 3의 배수가 아닙니다.

18

38.3 예외 발생시키기



» raise의 처리 과정

- 다음은 함수 안에서 raise를 사용하지만 함수 안에는 try except가 없는 상태임

try_except_function_raise.py

```
def three_multiple():
    x = int(input('3의 배수를 입력하세요: '))
    if x % 3 != 0:
        raise Exception('3의 배수가 아닙니다.')
    print(x)

try:
    three_multiple()
except Exception as e:
    print('예외가 발생했습니다.', e)
```

x가 3의 배수가 아니면
예외를 발생시킴
현재 함수 안에는 except가 없으므로
예외를 상위 코드 블록으로 넘김

하위 코드 블록에서 예외가 발생해도 실행됨

실행 결과

```
3의 배수를 입력하세요: 5 (입력)
예외가 발생했습니다. 3의 배수가 아닙니다.
```

19

38.3 예외 발생시키기



» raise의 처리 과정

- 다음은 파이썬 셸에서 직접 three_multiple 함수를 호출했으므로 except가 없는 상태임

```
>>> three_multiple()
3의 배수를 입력하세요: 5 (입력)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    three_multiple()
  File "C:\project\try_except_function_raise.py", line 4, in three_multiple
    raise Exception('3의 배수가 아닙니다.')
Exception: 3의 배수가 아닙니다.
```

예외를 발생시킴

20

38.3 예외 발생시키기



» 현재 예외를 다시 발생시키기

- except 안에서 raise를 사용하면 현재 예외를 다시 발생시킴(re-raise)
- **raise**
- 다음은 three_multiple 코드 블록의 예외를 다시 발생시킨 뒤 상위 코드 블록에서 예외를 처리함

try_except_raise_in_except.py

```
def three_multiple():
    try:
        x = int(input('3의 배수를 입력하세요: '))
        if x % 3 != 0:
            raise Exception('3의 배수가 아닙니다.') # 예외를 발생시킴
        print(x)
    except Exception as e:
        # 함수 안에서 예외를 처리함
        print('three_multiple 함수에서 예외가 발생했습니다.', e)
        raise # raise로 현재 예외를 다시 발생시켜서 상위 코드 블록으로 넘김

try:
    three_multiple()
except Exception as e:
    # 하위 코드 블록에서 예외가 발생해도 실행됨
    print('스크립트 파일에서 예외가 발생했습니다.', e)
```

21

38.3 예외 발생시키기



» 현재 예외를 다시 발생시키기

실행 결과

3의 배수를 입력하세요: 5 (입력)
three_multiple 함수에서 예외가 발생했습니다. 3의 배수가 아닙니다.
스크립트 파일에서 예외가 발생했습니다. 3의 배수가 아닙니다.

- 참고로 raise만 사용하면 같은 예외를 상위 코드 블록으로 넘기지만 raise에 다른 예외를 지정하고 에러 메시지를 넣을 수도 있음

raise 예외('에러메시지')

```
if x % 3 != 0:
    raise Exception('3의 배수가 아닙니다.')
print(x)
except Exception as e:
    print('three_multiple 함수에서 예외가 발생했습니다.', e)
    raise RuntimeError('three_multiple 함수에서 예외가 발생했습니다.')
```

22

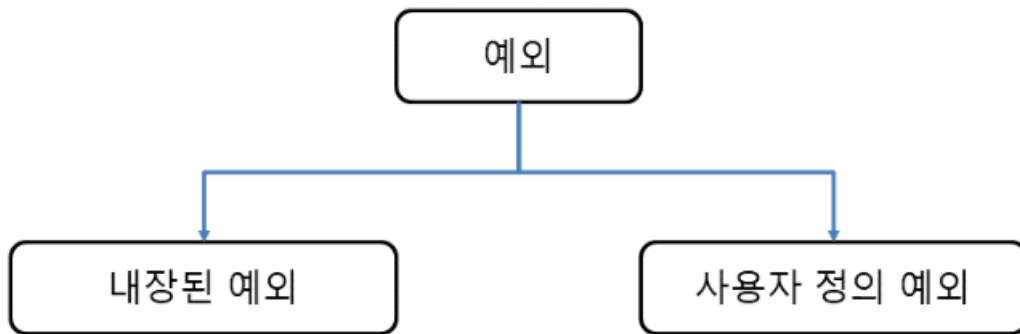
38.4 예외 만들기



» 예외 만들기

- 프로그래머가 직접 만든 예외를 사용자 정의 예외라고 함

▼ 그림 38-4 내장된 예외와 사용자 처리 예외



23

38.4 예외 만들기



» 예외 만들기

- 예외를 만드는 방법은 Exception을 상속받아서 새로운 클래스를 만들면 됨
- __init__ 메서드에서 기반 클래스의 __init__ 메서드를 호출하면서 에러 메시지를 넣어주면 됨

```
class 예외이름(Exception):  
    def __init__(self):  
        super().__init__('에러메시지')
```

24

38.4 예외 만들기



» 예외 만들기

- 그럼 입력된 숫자가 3의 배수가 아닐 때 발생시킬 예외를 만들어보자

exception_class.py

```
class NotThreeMultipleError(Exception):    # Exception을 상속받아서 새로운 예외를 만들
    def __init__(self):
        super().__init__('3의 배수가 아닙니다.')

    def three_multiple():
        try:
            x = int(input('3의 배수를 입력하세요: '))
            if x % 3 != 0:                # x가 3의 배수가 아니면
                raise NotThreeMultipleError    # NotThreeMultipleError 예외를 발생시킴
            print(x)
        except Exception as e:
            print('예외가 발생했습니다.', e)

three_multiple()
```

실행 결과

```
3의 배수를 입력하세요: 5 (입력)
예외가 발생했습니다. 3의 배수가 아닙니다.
```

25

38.4 예외 만들기



» 예외 만들기

- Exception을 상속받아서 NotThreeMultipleError 예외를 만들고 __init__ 메서드 안에서 기반 클래스의 __init__ 메서드를 호출하면서 에러 메시지를 넣었음

```
class NotThreeMultipleError(Exception):    # Exception을 상속받아서 새로운 예외를 만들
    def __init__(self):
        super().__init__('3의 배수가 아닙니다.')
```

- 예외를 발생시킬 때는 raise NotThreeMultipleError와 같이 raise에 새로 만든 예외를 지정해주면 됨

```
class NotThreeMultipleError(Exception):    # Exception만 상속받고
    pass                                  # 아무것도 구현하지 않음
```

```
raise NotThreeMultipleError('3의 배수가 아닙니다.')    # 예외를 발생시킬 때 에러 메시지를 넣음
```

26