예외처리

안 화 수

Exception

- 예외(Exception)는 문법적으로는 문제가 없는 코드를 실행하는 중에 발생하는 오류

```
def func(y):
    x = input("숫자를 입력하세요:")
    return int(x) ** y
print(func(2))



숫자를 입력하세요:t
Traceback (most recent call last):
  File "C:₩Users₩Administrator₩python₩test₩__main__.py", line 4, in
<module>
    print(func(2))
  File "C:₩Users₩Administrator₩python₩test₩__main__.py", line 3, in func
    return int(x) ** y
ValueError: invalid literal for int() with base 10: 't'
```

try ~ except

- try 절 안에 예외 발생 가능성이 있는 코드 블록을 배치
- except 절에는 예외가 발생했을 때 처리를 하는 코드 블록 배치

```
print(1/0)

Traceback (most recent call last):
  File "C:₩Users₩Administrator₩python₩test₩__main__.py",
line 1, in <module>
    print(1/0)
ZeroDivisionError: division by zero
```

```
try:
    print(1/0)
except:
    print("예외가 발생했습니다.")
```

try ~ except

- try 블록 안에서 여러 종류의 예외가 발생하는 경우에는 except에 예외형식을 기재해서 분리해서 처리 가능

```
try:
  # 예외가 발생할 가능성이 있는 코드
except 예외형식1:
# 예외형식1에 해당하는 예외가 발생했을 때 수행될 코드
except 예외형식2:
# 예외형식2에 해당하는 예외가 발생했을 때 수행될 코드
```

```
li = [1, 2, 3]

try:
    index = int(input("첨자를 입력하
    print(li[index]/0)
except ZeroDivisionError:
    print("0으로 나눌 수 없습니다.")
except IndexError:
    print("잘못된 첨자입니다.")
```

index가 0~2사이로 입력된다면 ZeroDivisionError가 일어납니다.

index가 0~2를 벗어나면 li[index]에서 IndexError가 발생합니다.

try ~ except

- 예외의 인스턴스를 활용하는 방법 : as 문 사용

```
try:
  # 문제가 없을 경우 실행할 코드
except 예외형식1 as err:
# 문제가 생겼을 때 실행할 코드
except 예외형식2 as err:
# 문제가 생겼을 때 실행할 코드
```

```
li = [1, 2, 3]

try:
    index = int(input("첨자를 입력하세요:"))
    print(li[index]/0)
except ZeroDivisionError as err:
    print("0으로 나눌 수 없습니다. ({0})".format(err))
except IndexError as err:
    print("잘못된 첨자입니다. ({0})".format(err))
```

try ~ except

- try에 대한 else가 아닌 "except절에 대한 **else**"

```
try:
    # 실행할 코드블록
except:
    # 예외 처리 코드블록
else:
    # except절을 만나지 않았을 경우 실행하는 코드블록
```

```
li = [1, 2, 3]

try:
    index = int(input("첨자를 입력하세요:"))
    print(li[index])
except IndexError as err:
    print("잘못된 첨자입니다. ({0})".format(err))
else:
    print("리스트의 요소 출력에 성공했습니다.")
```

try ~ except

- finally는 예외 발생 여부에 상관없이 수행되는 문장을 작성하는 영역
- finally가 else는 함께 사용하는 것도 가능함.

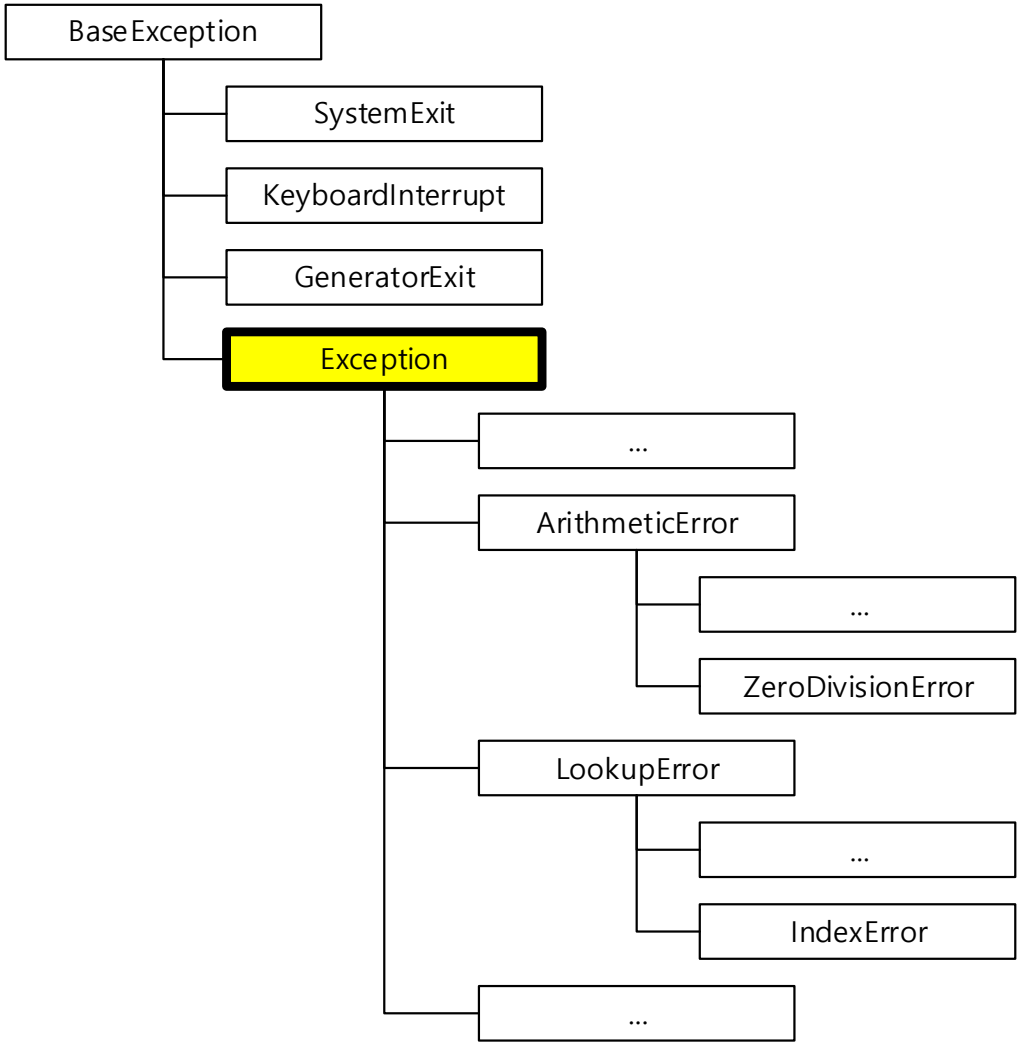
```
try:
    # 코드 블록
except:
    # 코드블록
else:
    # 코드블록
finally:
    # 코드블록
```

```
li = [1, 2, 3]

try:
    index = int(input("첨자를 입력하세요:"))
    print(li[index])
except IndexError as err:
    print("잘못된 첨자입니다. ({0})".format(err))
else:
    print("리스트의 요소 출력에 성공했습니다.")
finally:
    print("무조건 수행")
```

Exception 클래스

- 파이썬에서 제공하는 예외 형식들은 거의 모두 Exception 클래스로부터 파생



내장 예외의 강제 발생

- raise문을 이용하면 내장 예외를 강제로 일으킬 수 있음.

```
text = input()
if text.isdigit() == False:
    rasie Exception("입력받은 문자열이 숫자로 구성되어 있지 않습니다."):
```

raise문을 통해 예외를 일으킵니다.

```
>>> raise Exception("예외를 일으킵니다.")
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    raise Exception("예외를 일으킵니다.")
Exception: 예외를 일으킵니다.
```

예외를 처리하는 곳이 없다 보니 파이썬 인터프리터가 받아 예외 정보를 출력했습니다.

```
>>> try:
        raise Exception("예외를 일으킵니다.")
except Exception as err:
        print("예외가 일어났습니다. : {0}".format(err))


예외가 일어났습니다. : 예외를 일으킵니다.
```

예외의 외부 처리

```
def some_function():
    print("1~10 사이의 수를 입력하세요:")
    num = int(input())
    if num < 1 or num > 10:
        raise Exception("유효하지 않은 숫자입니다.: {0}".format(num))
    else:
        print("입력한 수는 {

try:
    some_function()
except Exception as err:
    print("예외가 발생했습니다. {0}".format(err))
```

함수 안에서 일어난 예외가 except문으로 처리되지 않으면 함수 밖으로 다시 던져집니다.

사용자 정의 예외

- 파이썬이 제공하는 내장 예외 형식만으로 충분하지 않을 때 직접 예외 클래스를 정의할 수 있음.
- 사용자 정의 예외 클래스는 Exception 클래스를 상속하여 정의함.

```
class MyException(Exception):
    pass
```

- 필요에 따라 다음과 같이 데이터 속성이나 메소드를 추가 가능.

```
class MyException(Exception):
    def __init__(self):
        super().__init__( "MyException이 발생했습니다." )
```

사용자 정의 예외

```
class InvalidIntException(Exception):
    def __init__(self, arg):
        super().__init__('정수가 아닙니다.: {0}'.format(arg))

def convert_to_integer(text):
    if text.isdigit(): # 부호(+, -) 처리 못함.
        return int(text)
    else:
        raise InvalidIntException(text)

if __name__ == '__main__':
    try:
        print('숫자를 입력하세요:')
        text = input()
        number = convert_to_integer(text)
    except InvalidIntException as err:
        print('예외가 발생했습니다 ({0})'.format(err))
    else:
        print('정수 형식으로 변환되었습니다 : {0}({1}'.format(number, type(number)))
```

assert

- 예외가 발생할 상황이 아님에도 사용자가 강제로 예외를 만들 수 있습니다.
- assert <중단 조건>, <에러 메시지>

```
a = 30
margin = 2 * 0.2
assert margin > 10, '마진이 작습니다.'


Traceback (most recent call last):
  File "C:₩Users₩Administrator₩python₩test₩__main__.py", line 3, in <module>
    assert margin > 10, '마진이 작습니다.'
AssertionError: 마진이 작습니다.
```