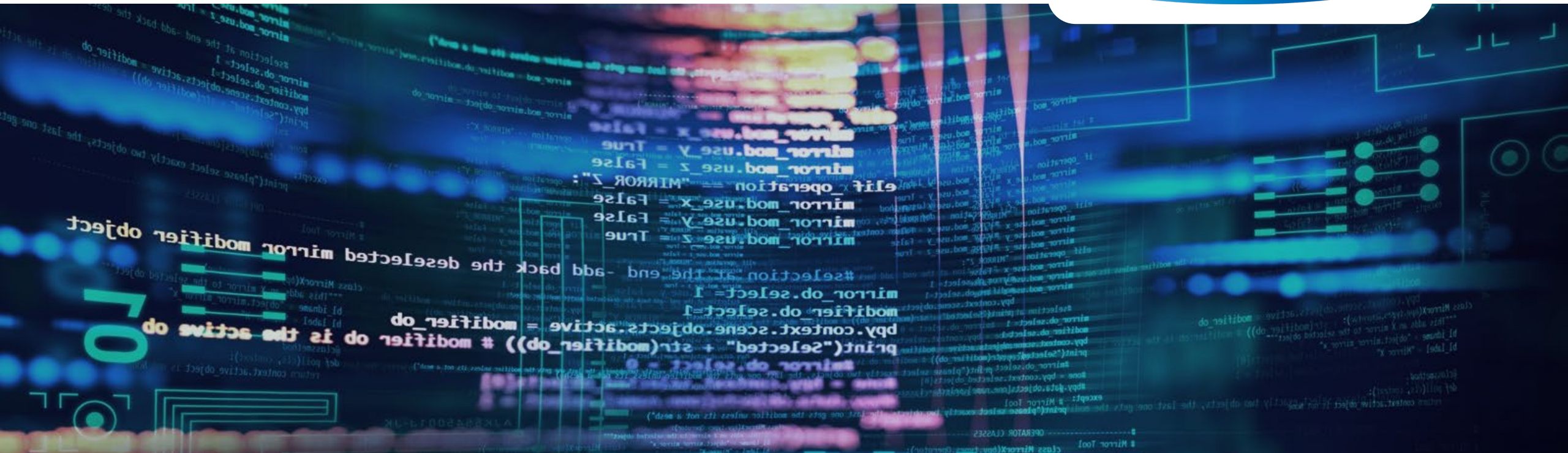
Lecture 12

에러 핸들링

KAIST



목차

KAIST

- 예외처리의 필요성
- try 와 except에 대한 이해
- else와 finally에 대한 이해
- 에러 강제로 발생시키기
- 예상치 못한 상황에서 강제 종료 시키기

예외처리의 필요성

에러가 발생하는 상황이 존재한다면 프로그램은 **실행 도중에 중지됨**

프로그램이 중지되어 절대 안된다면, 어떻게 대비해야할까?

예제1

```
def division(a, b):
    return a/b

print(division(5, 0)) # ZeroDivisionError: division by zero
print(division(5, 2)) # 출력안됨 (위에서 실행 멈춤)
```

try와 except에 대한 이해

에러가 발생하는 상황이 존재한다면 프로그램은 **실행 도중에 중지됨**

try와 except의 필요성!

예제2

```
def division(a, b):
    try:
        return a/b
    except:
        return '예외가 발생했습니다'

print(division(5, 0)) # 출력: 예외가 발생했습니다
print(division(5, 2)) # 출력: 2.5
```

try와 except에 대한 이해

에러가 발생하는 상황이 존재한다면 프로그램은 **실행 도중에 중지됨**

try와 except의 필요성!

예제3

```
def division(a, b):
    try:
        return a/b
    except ZeroDivisionError:
        return '0으로 나눌 수 없습니다'

print(division(5, 0)) # 출력: 0으로 나눌 수 없습니다
print(division(5, 2)) # 출력: 2.5
```

try와 except에 대한 이해

에러가 발생하는 상황이 존재한다면 프로그램은 **실행 도중에 중지됨**

try와 except의 필요성!

예제4

```
def division(a, b):
    try:
        return a/b
    except ZeroDivisionError as error:
        return f '0으로 나눌 수 없습니다 {error}'

print(division(5, 0)) # 출력: 0으로 나눌 수 없습니다 division by zero
print(division(5, 2)) # 출력: 2.5
```

try와 except에 대한 이해

에러가 발생하는 상황이 존재한다면 프로그램은 **실행 도중에 중지됨**

try와 except의 필요성!

예제5

```
def division(a, b):
    try:
        return a/b
    except ZeroDivisionError as error:
        return f'0으로 나눌 수 없습니다 {error}'
    except TypeError as error:
        return f'타입을 확인하세요 {error}'

print(division(5, 0)) # 출력: 0으로 나눌 수 없습니다 division by zero
print(division('5', 2)) # 출력: 타입을 확인하세요 unsupported operand type(s) for /: 'str' and 'int'
```

다양한 에러메세지 설명

파이썬에는 어떤 종류의 **에러메세지**가 있을까?

에러메시지	설명
SyntaxError	파이썬 문법의 오류
NameError	변수 이름의 오류
IndexError	리스트나 튜플의 인덱스 오류
AttributeError	클래스의 멤버함수/멤버변수의 오류
FileNotFoundError	파일/폴더의 오류 (예) 파일/폴더가 없는 경우
ValueError	잘못된 변수 오류 (예) int('안녕하세요')
ImportError	라이브러리/패키지 불러오기 오류

else와 finally에 대한 이해

KAIST

try와 except는 이해가 되겠는데, 뭐가 더 필요하지?

else와 finally의 필요성!

예제6

```
def division(a, b):
    try:
        print(a/b)
    except ZeroDivisionError:
        print(f'0으로 나눌 수 없습니다')
    except TypeError:
        print(f'타입을 확인하세요')
    else:
        print('else 오류가 없을 때만 실행')
    finally:
        print('finally 오류가 있던 없던 실행')

division(5, 2) #출력: 2.5, else 오류가 없을 때만 실행, finally 오류가 있던 없던 실행
division(5, 0) #출력: 0으로 나눌 수 없습니다 division by zero, finally 오류가 있던 없던 실행
division('5', 2) #출력: 타입을 확인하세요, finally 오류가 있던 없던 실행
```

else와 finally에 대한 이해

KAIST

try와 except는 이해가 되겠는데, 뭐가 더 필요하지?

else와 finally의 필요성!

예제6

```
def division(a, b):
    try:
        print(a/b)
    except ZeroDivisionError:
        print(f'0으로 나눌 수 없습니다')
    except TypeError:
        print(f'타입을 확인하세요')
    else:
        print('else 오류가 없을 때만 실행')
    finally:
        print('finally 오류가 있던 없던 실행')

division(5, 2) #출력: 2.5, else 오류가 없을 때만 실행, finally 오류가 있던 없던 실행
division(5, 0) #출력: 0으로 나눌 수 없습니다 division by zero, finally 오류가 있던 없던 실행
division('5', 2) #출력: 타입을 확인하세요, finally 오류가 있던 없던 실행
```

else와 finally 순서를 바꿔서 정의하면 오류!
반드시 else 다음에 finally 가 정의!

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제7

```
def add(a, b):
    try:
        if a==3:
            raise ZeroDivisionError
        print(a+b)
    except ZeroDivisionError:
        print(f'0으로 나눌 수 없습니다')

add(3, 5) # 출력: 0으로 나눌 수 없습니다
```

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제7

```
def add(a, b):
    try:
        if a==3:    에러 강제로 발생시키는 문법: raise
            raise ZeroDivisionError
        print(a+b)
    except ZeroDivisionError:
        print(f'0으로 나눌 수 없습니다')

add(3, 5) # 출력: 0으로 나눌 수 없습니다
```

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제7

```
def add(a, b):
    try:
        if a==3:
            raise ZeroDivisionError
        print(a+b)
    except ZeroDivisionError:
        print(f'0으로 나눌 수 없습니다')

add(3, 5) # 출력: 0으로 나눌 수 없습니다
```

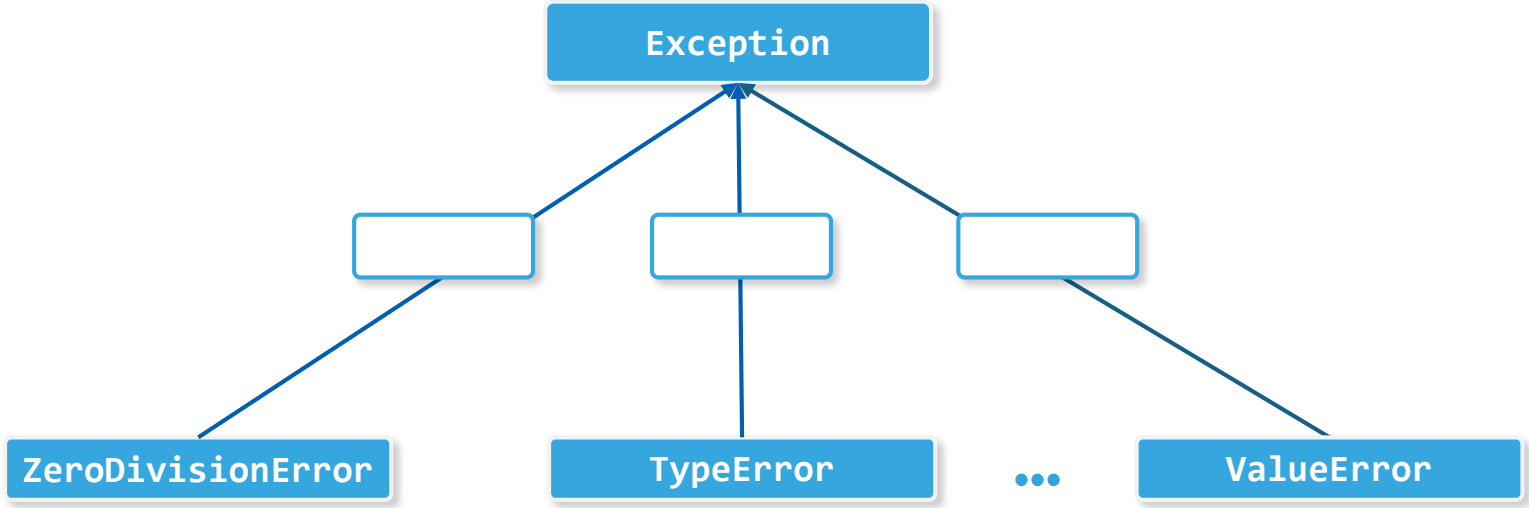
에러 강제로 발생시키는 문법: raise

except로 전이됨

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기



파이썬 에러메세지는 기본적으로 상속구조를 가지고 있음

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제8

```
def add(a, b):
    try:
        if a==3:
            raise Exception
        print(a+b)
    except Exception:
        print(f'커스텀 에러 메세지')

add(3, 5) # 출력: 커스텀 에러 메세지
```

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제8

```
def add(a, b):
    try:
        if a==3:
            raise Exception
        print(a+b)
    except Exception:
        print(f'커스텀 에러 메세지')

add(3, 5) # 출력: 커스텀 에러 메세지
```

커스텀 에러 만들기

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제9

```
def add(a, b):
    try:
        if a==3:
            raise Exception('내 마음대로 에러 메세지')
        print(a+b)
    except Exception as error:
        print(f'정말, {error}')

add(3, 5) # 출력: 정말, 내 마음대로 에러 메세지
```

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제9

```
def add(a, b):
    try:
        if a==3:
            raise Exception('내 마음대로 에러 메세지')
        print(a+b)
    except Exception as error:
        print(f'정말, {error}')

add(3, 5) # 출력: 정말, 내 마음대로 에러 메세지
```

커스텀 에러메세지 만들기

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제10

```
def add(a, b):
    try:
        if a==3:
            raise SyntaxError('하하하')
        print(a+b)
    except SyntaxError as error:
        print(f'정말, {error}')

add(3, 5) # 출력: 정말, 하하하
```

에러 강제로 발생시키기

KAIST

개발자 입장에서 원하는 에러로 **강제로 발생**시키기

예제10

```
def add(a, b):
    try:
        if a==3:
            raise SyntaxError('하하하')
        print(a+b)
    except SyntaxError as error:
        print(f'정말, {error}')

add(3, 5) # 출력: 정말, 하하하
```

기존의 에러 클래스의 메시지도 변경 가능

예상치 못한 상황에서 강제 종료시키기

KAIST

개발자 입장에서 원하는 곳에서 예상치 못한 동작을 할 때 프로그램을 **강제로 종료**시키기

- assert 조건
- assert 조건, 메시지(조건이 아닐경우)

예제11

```
def add(a, b):
    assert a!=3, 'a가 3이다!'
    print(a+b)
add(3, 5) # 출력: AssertionError: a가 3이다!
add(4, 5) # 출력: 9
```

예상치 못한 상황에서 강제 종료시키기

KAIST

개발자 입장에서 원하는 곳에서 예상치 못한 동작을 할 때 프로그램을 **강제로 종료**시키기

- assert 조건
- assert 조건, 메시지(조건이 아닐경우)

예제11

```
def add(a, b):
    assert a!=3, 'a가 3이다!'
    print(a+b)
add(3, 5) # 출력: AssertionError: a가 3이다!
add(4, 5) # 출력: 9
```

a가 3일 경우에 강제 종료되면서 메시지 출력