선문대학교 SUN MOON UNIVERSITY

사용자 정의 예외

11주차

예외처리

MEMO

사용자 정의 제외

· 개발자가 의도적으로 예외를 발생시켜야 하는 경우

raise 명령어

- 예외를 강제적으로 발생시킴
- raise로 예외를 발생시키면 raise아래에 있는 코드들을 실행되지 않고 바로 except로 넘어감

```
raise 예외("에러메시지")
raise 내장에러
```

- 반드시 내장 클래스인 Exception을 상속받아 정의해야 함

MEMO

인원수가 0인 경우와 입력한 값이 숫자가 아닌 경우 에러발생

```
try :
    a = int(input("인원을 입력하세요"))
    if (a == 0) :
        raise ZeroDivisionError

    b = int(input("금액을 입력하세요"))
    if (b < 0) :
        raise Exception("음수는 입력할 수 없습니다")

    c = b / a
    print("한사람당", c , "원씩 나누면 됩니다")

except ValueError:
    print("인원을 숫자로 입력하세요")
except ZeroDivisionError as e
    print("인원은 1이상이여야 합니다", e)
except Exception as e :
    print("알 수 없는 에러입니다", e)
```

MEMO

인원수가 0인 경우와 입력한 값이 숫자가 아닌 경우 에러발생

```
try :
    a = int(input("인원을 입력하세요"))
    if (a == 0) :
        raise ZeroDivisionError

    b = int(input("금액을 입력하세요"))
    if (b < 0) :
        raise Exception("음수는 입력할 수 없습니다")

    c = b / a
    print("한사람당", c , "원씩 나누면 됩니다")

except ValueError:
    print("인원을 숫자로 입력하세요")
except ZeroDivisionError as e
    print("인원은 1이상이여야 합니다", e)
except Exception as e :
    print("알 수 없는 에러입니다", e)
```

MEMO

인원수가 0인 경우와 입력한 값이 숫자가 아닌 경우 에러발생

```
def calc(a, b):
    if b < 0 :
        raise Exception("0 이상의 숫자를 입력하세요")
    c = b / a

try :
    a = int(input("인원을 입력하세요"))

    b = int(input("금액을 입력하세요"))

    c = calc(a, b)

except ValueError:
    print("인원을 숫자로 입력하세요")
except ZeroDivisionError as e :
    print("인원은 1이상이여야 합니다", e)
except Exception as e :
    print("알 수 없는 에러입니다", e)
else :
    print("한사람당 ", c, "원씩 나누면 됩니다")
finally :
    print("계산이 모두 완료되었습니다")
```

MEMO

인원수가 0인 경우와 입력한 값이 숫자가 아닌 경우 에러발생

- calc 함수안에서 try except가 없는 상태에서 raise를 발생시키면 함수 바깥쪽에 있는 except에서 예외 처리

```
def calc(a, b):
    if b < 0 :
        raise Exception("0이상의 숫자를 입력하세요")
    c = b / a
```

- except가 나올 때까지 계속 상위 코드 블록으로 올라가서 처리
- 찾아도 없으면 코드 실행은 중지되고 에러표시

MEMO

- 파일 입출력에 관해서 오류가 발생할 가능성이 매우 높음
- 파일을 읽고 쓸 때에는 반드시 파일 오류를 처리해 줄 수 있는 예외처리를 구현
  - 주로 해당파일이 없거나 다른 실행파일에서 해당파일을 사용 중일 때

MEMO

선문대학교 SUN MOON UNIVERSITY

파일 입출력 시 오류 발생

```
try :
    fopen = open("money1.txt", "r")
except :
    print("파일 읽기에 오류가 발생하였습니다")
```

MEMO

파일 입출력 시 오류 발생

EofError	파일 등에서 읽어 들일 데이터가 더 이상 없을 때 오류 발생
FileExistsError	이미 존재하는 파일이나 폴더를 새로 생성하려 할 때 오류 발생
FileNotFoundError	존재하지 않는 파일이나 폴더를 오픈하려 할 때 오류 발생

MEMO

파일 입출력 시 오류 발생

```
try :
    fr = open("money.txt", "r")
except FileNotFoundError :
    print("파일을 찾을수 없습니다")
except Except as e :
    print("알수없는 에러가 발생하였습니다", e)
else :
    print("총 ", len(fr.readlines()), "건의 데이터가 있습니다")
    fr.close()
finally :
    print("프로그램 수행이 완료되었습니다")
```

1 구문에러

- 문법상 맞지 않는 구문으로 작성하였을 때 소스코드를 실행하기 전에 발생하는 에러

MEMO

2 예외

- 소스코드 실행 중에 발생한 에러
- 사용자의 잘못된 입력 등으로 인해서 발생되는 에러
- 파이썬에서는 다양한 유형의 에러를 미리 정해 놓고 에러 타입을 정의해둠
- Try 구문을 이용하면 발생 가능한 예외를 적절하게 처리할 수 있음

MEMO

3 try except문의 사용 방법

```
try :
    실행할 명령문
except 예외이름 :
    예외처리 명령문
except (예외1, 예외2) :
    예외처리 명령문
except 예외3 as 인자 :
    예외처리 명령문
else :
    예외가 발생하지 않을 경우,실행할 명령문
finally :
    예외 발생 유무에 상관없이 try이후 수행할 문장
```

MEMO

4 예외처리

- 모든 에러에 대해서 예외처리는 Exception에서 처리함
- except Exception as e 콜론 혹은 except 콜론은 모든 에러에 대해서 예외처리에 가능하며 프로그램에서 많이 사용됨
- else는 예외상황이 발생하면 except구문을 실행하고, 예외상황이 발생하지 않으면 else구문을 실행함

MEMO

4 예외처리

- finally는 try문 수행 도중 예외 발생 여부에 상관없이 항상 수행되어야 하는 문장을 있을 때 사용됨

MEMO

5 사용자 정의 예외

- 개발자가 의도적으로 예외를 발생시켜야 하는 경우
- raise 명령어를 사용해 예외를 강제적으로 발생시킴
- raise로 예외를 발생시키면 raise 아래에 있는 코드들은 실행되지 않고 바로 except로 넘어가게 됨
- raise 예외(에러 메시지), raise 내장에러, rasie 방식으로 사용됨

MEMO

5 사용자 정의 예외

- 사용자 정의 예외를 생성하는 경우 반드시 내장 클래스인 Exception을 상속받아 정의해야 함

MEMO

도서관에서 책을 빌리는 프로그램

- 1 –호밀밭의 파수꾼들, 2-데미안, 3:총, 균, 쇠 4: 해리포터와 불의 잔, 5:파이썬 프로그램, 6:자바 프로그램, 7:칼의 노래
- 목록에 없는 책을 빌릴 경우 : “없는 도서 목록입니다”라는 사용자 정의 에러 발생
- 책 목록에 숫자가 들어가지 않으면, valueError를 이용해 예외처리

```
빌리려고 하는 책 목록은?7
칼의 노래를 빌립니다
프로그램을 종료합니다

빌리려고 하는 책 목록은?15
알 수 없는 에러가 발생하였습니다 없는 도서 목록입니다
프로그램을 종료합니다

빌리려고 하는 책 목록은?a
목록은 숫자로 되어있습니다 숫자를 입력하세요
프로그램을 종료합니다
```

MEMO

다음 시간에는

12주차. 객체지향과 클래스

MEMO