Chapter 1

예외처리

예외 처리

- 예외(Exception) : 실행 중에 발견 된 오류

- 예외 처리(Exception Handling) : 오류 발생에 대한 대처 방법 중의 하나

- 예외 처리는 시스템 스스로 오류를 복구 하는 것이 아니고 **오류발생 가능성이 있는 부분에 대한 처리를 미리 프로그래밍 하는 것**

- 다음과 같은 상황들은 예외 처리를 해야 할 필요가 있음
    - 파일을 다룰 때 파일이 없거나 쓰기 금지로 인한 오류
    - 데이터베이스 프로그래밍 시 제약조건 등에 의한 데이터베이스 서버 오류
    - 네트워크 프로그래밍 시 네트워크 연결 실패로 인한 오류
    - 리스트 또는 튜플의 인덱스를 벗어난 참조로 인한 오류

예외의 종류들

주요 실행 예외

1. NameError: 정의되지 않은 변수나 함수, 클래스를 사용할 때 발생합니다.

2. ValueError: 주로 형 변환 시 발생하며, 내부 값의 형태가 잘못되었을 때 발생합니다.

3. ZeroDivisionError: 숫자를 0으로 나누었을 때 발생합니다.

4. IndexError, KeyError: 존재하지 않는 인덱스나 키를 사용하여 시퀀스, 딕셔너리를 조회했을 때 발생합니다.

5. TypeError: 연산 수행 시 피 연산자의 데이터 타입이 올바르지 않을 경우 발생합니다.

예외의 종류 까진 외울 필요는 없다.
하지만 이런 실행예외들은 개발자의 경험에 의해서 예외 처리 코드를 삽입해야 한다

예외 처리 방법

1. 예외 처리하기

- try는 예외를 처리할 때 예외가 발생할 가능성이 있는 문장을 작성
- try 절(또는 블록)에서 예외가 발생하면 except 절을 찾음
- except 절은 예외가 발생했을 경우 실행되어야 할 코드를 작성

```
try :
    # 예외가 발생할 가능성이 있는 문장
except :
    # 예외가 발생했을 경우 실행할 문장
```

```
while True:
    try:
        x = int(input('정수를 입력하세요: '))
        print('입력한 정수는 {}입니다.'.format(x))
        break
    except:
        print('유효한 정수가 아닙니다. 다시 시도하세요.')
```

```
정수를 입력하세요: hello
유효한 정수가 아닙니다. 다시 시도하세요.
정수를 입력하세요: 12345
입력한 정수는 12345입니다.
```

예외 처리 방법

2. 예외의 종류 지정하기

- except 절은 예외 이름을 생략 할 수 있음
- 프로그래밍 오류를 쉽게 처리 할 수 있지만 실제로 어떤 오류를 처리하는지는 알 수 없음
- 가능하다면 except 절에 **예외를 지정**해서 사용하는 것이 좋음

```
try :
    # 예외가 발생할 가능성이 있는 문장
except Exception :
    # 예외가 발생했을 경우 실행할 문장
```

- 발생한 예외의 유형이 except 키워드 다음에 명명 된 예외와 일치하면 해당 except 절이 실행

- 예외를 처리할 except 절이 발견 되지 않으면 처리되지 않은 예외 이며 프로그램 실행은 중지됨

```
while True:
    try:
        x = int(input('정수를 입력하세요: '))
        print('입력한 정수는 {}입니다.'.format(x))
        break
    except ValueError:
        print('유효한 정수가 아닙니다. 다시 시도하세요.')
```

예외 처리

3. 다중 예외 처리

- try 블록에서 발생할 수 있는 예외가 여러 개일 경우 각각의 예외를 처리하도록 catch 블록을 정의해 놓을 수 있음

```
try :
    # 예외가 발생할 가능성이 있는 문장
except Error1 :
    # Error1이 발생했을 경우 실행할 문장
except Error2 :
    # Error2가 발생했을 경우 실행할 문장
```

```
while True:
    try:
        x = int(input('정수를 입력하세요: '))
        print('입력한 정수는 {}입니다.'.format(x))
        print('100을 입력한 수로 나누면 {}입니다.'.format(100/x))
        break
    except ValueError:
        print('유효한 정수가 아닙니다. 다시 시도하세요.')
    except ZeroDivisionError:
        print('0으로 나눌 수 없습니다.')
```

```
정수를 입력하세요: 0
입력한 정수는 0입니다.
0으로 나눌 수 없습니다.
정수를 입력하세요: hello
유효한 정수가 아닙니다. 다시 시도하세요.
정수를 입력하세요: 20
입력한 정수는 20입니다.
100을 입력한 수로 나누면 5.0입니다.
```

예외 처리

4. Finally 키워드

finally 키워드는 예외 발생 여부와 상관없이 항상 실행해야 하는 코드가 있을 경우 사용하는 예외 처리 방식입니다.

```
try :
    코드 작성 영역...
except :
    처리 영역...
finally :
    예외와 상관없이 무조건 실행
```

예외 일부러 만들기

예외 강제로 발생시키기

- 프로그래밍을 진행하다 보면 일부러 예외를 발생시켜서 코드의 흐름을 전환해야 하는 경우가 발생합니다.
- raise라는 명령을 사용해서 오류를 강제로 발생시킬 수 있습니다.

```
def calc_sum(end):
    if end <= 0:
        raise Exception

    total = 0
    for n in range(end + 1):
        total += n
    return total
```

예외가 발생되고 즉시 종료된다.
함수를 호출할 때는 예외처리를 대신 진행한다.

```
try:
    result2 = calc_sum(-120)
except:
    처리문장...
```

Chapter 9

수고하셨습니다