[COM1002]
프로그래밍1

exception handling

#10. 예외처리

김현하

한양대학교 ERICA 소프트웨어학부

2021.12.7
2021년도 2학기

python™

예외

- 예외exception
  - 프로그램 실행 중 오류가 발생하여 오작동하거나 작동을 멈춘 상황
  - (고의든 아니든) 외부 입력에 의해 프로그래머의 의도와 다르게 프로그램이 작동하는 상황
- 예외 발생의 결과
  - 경제적인 손해부터 인적 피해까지 발생 가능

역대 10대 IT 재난

- [1983] 소련의 조기 경보 시스템 오류 -> 3차 세계대전 발발 위기
  - 미국이 다섯 대의 탄도 미사일을 발사했다고 거짓 경보 발생 (구름 위에서 반사된 햇빛을 위성이 미사일로 오인)
- [1990] AT&T 전화망 마비
  - 114 교환 센터의 교환기 한대가 사소한 오류 후 가동 중지
  - 복잡한 업그레이드 중 한줄의 코드에 있던 오류로 발발
- [1996] 아리안5 폭발
  - 64비트 숫자를 16비트 공간에 채우면서 오버플로우 오류 발생, 36.7초만에 로켓 폭발 (고장을 대비한 시스템에도 같은 오류가...)
- [2006] 에어버스 A380
  - 그룹 내 두 계열사 사이에 CATIA 버전이 달라서 배선이 불일치, 1년 가까이 프로젝트가 지연
- [1998] 화성 기후 탐사선이 너무 낮은 고도로 날다가 파괴
  - NASA 의 하청 업체에서 단위를 미터법이 아니라 인치/피트 를 사용
- [1999-2000] 밀레니엄 버그
  - 초기 컴퓨터의 프로그램들이 년도를 19XX 만 생각하고 두자리로 설정
- ...

출처 : https://zdnet.co.kr/view/?no=00000039163624

원문 : https://www.zdnet.com/article/top-10-worst-it-disasters-of-all-time/,
Colin Barker "Top 10 worst IT disasters of all time", 2007/11/23

예외 상황에 안전히 대처할 수 있도록 예방하는 코딩 활동

방어 프로그래밍
Defensive Programming

안전 코딩
Secure Coding

예외 처리
Exception Handling

내장 예외

내장 예외

- 내장 예외built-in exception
  - 프로그램 실행 중 많이 발생하리라 예상되어 미리 정해둔 예외
  - https://docs.python.org/3/library/exceptions.html

예외 타입	발생 상황
SyntaxError	문법이 틀린 경우
TypeError	피연산자 또는 함수 인수의 타입이 틀린 경우
ValueError	피연산자 또는 함수 인수의 값이 틀린 경우
NameError	지정한 적이 없는 모르는 이름이 나타난 경우
IndexError	없는 인덱스를 사용한 경우
KeyError	없는 키를 사용한 경우
ZeroDivisionError	0으로 나누려 하는 경우
IOError	없는 파일을 열거나, 열지 않고 파일을 읽거나 쓰려는 경우

예외 처리 제어 구조

예외 처리 제어 구조

```
1  x = int(input("Enter a number: "))
2  reciprocal = 1 / x
3  print("The reciprocal of", x, "is", reciprocal)
```

```
1  try:
2      x = int(input("Enter a number: "))
3      reciprocal = 1 / x
4      print("The reciprocal of", x, "is", reciprocal)
5  except ValueError:
6      print("Not a number.")
7  except ZeroDivisionError:
8      print("The reciprocal of 0 does not exist.")
```

예외 처리 제어 구조

```
1  x = int(input("Enter a number: "))
2  reciprocal = 1 / x
3  print("The reciprocal of", x, "is", reciprocal)
```

```
1  while True:
2      try:
3          x = int(input("Enter a number: "))
4          reciprocal = 1 / x
5          print("The reciprocal of", x, "is", reciprocal)
6          break
7      except ValueError:
8          print("Not a number.")
9      except ZeroDivisionError:
10         print("The reciprocal of 0 does not exist.")
11         break
```

예외 처리 제어 구조

문법

```
try:
    <블록>
except <예외이름>1:
    <블록>1
except <예외이름>2:
    <블록>2
...
except <예외이름>n:
    <블록>n
```

의미

- try 블록인 <블록>을 실행한다.
  - 예외 상황이 발생하지 않고 <블록>의 실행을 종료하면, except 블록은 모두 무시한다.
  - <블록>의 실행 도중 예외상황이 발생하면 try 블록의 남은 부분은 무시하고, <예외이름>$_1$, <예외이름>$_2$, ..., <예외이름>$_n$ 중에서 발생한 예외와 같은 이름(종류)을 위에서부터 순서대로 찾아서 해당 블록을 실행한다.
  - 발생한 예외와 같은 이름의 <예외이름>이 없으면 try 블록을 벗어나 예외를 발생raise시킨다.
  - 예외가 프로그램 범위를 벗어날 경우, 발생한 오류 메시지를 내주면서 실행을 멈춘다.

예외 처리 제어 구조

```
1   while True:
2       try:
3           x = int(input("Enter a number: "))
4           reciprocal = 1 / x
5           print("The reciprocal of", x, "is", reciprocal)
6           break
7       except ValueError:
8           print("Not a number.")
9       except ZeroDivisionError:
10          print("The reciprocal of 0 does not exist.")
11          break
12      except:
13          print("Unexpected exception occurred.")
14          break
```

예외이름을 생략(무명 except)하면 모든 예외를 처리 가능

제대로 예외를 처리했는지 확인이 불가능하므로 남용 금지

예외 처리 제어 구조

```
1  while True:
2      try:
3          x = int(input("Enter a number: "))
4          reciprocal = 1 / x
5          print("The reciprocal of", x, "is", reciprocal)
6          break
7      except ValueError as message:
8          print(message)
9      except ZeroDivisionError as message:
10         print(message)
11         break
```

예외에 담긴 오류 메시지를 사용

예외 처리 제어 구조

```
1   while True:
2       try:
3           x = int(input("Enter a number: "))
4           reciprocal = 1 / x
5       except ValueError:
6           print("Not a number.")
7       except ZeroDivisionError:
8           print("The reciprocal of 0 does not exist.")
9           break
10      else:
11          print("The reciprocal of", x, "is", reciprocal)
12          break
```

else 블록은 try 블록에서 예외가 발생하지 않은 경우만 실행

예외 처리 제어 구조

```
1   while True:
2       try:
3           x = int(input("Enter a number: "))
4           reciprocal = 1 / x
5       except ValueError:
6           print("Not a number.")
7       except ZeroDivisionError:
8           print("The reciprocal of 0 does not exist.")
9           break
10      else:
11          print("The reciprocal of", x, "is", reciprocal)
12          break
13      finally:
14          print(":-)")
```

`finally` 블록은 예외발생 여부에 상관 없이 무조건 실행

assert 문

assert 문

문법

```
assert <논리식>
```

의미

- <논리식>의 계산 결과가
  - True 이면 그냥 통과
  - False 이면 AssertionError 라는 이름의 예외를 발생

```
1  def fac(n):
2      ans = 1
3      while n > 1:
4          ans = n * ans
5          n = n - 1
6      return ans
```

```
1  def factorial():
2      n = int(input("Enter a number: "))
3      print("factorial(", n, ")= ", fac(n), sep='')
```

```
1  def factorial():
2      n = int(input("Enter a number: "))
3      assert n >= 0
4      print("factorial(", n, ")= ", fac(n), sep='')
```

assert 문

```
1  def fac(n):
2      ans = 1
3      while n > 1:
4          ans = n * ans
5          n = n - 1
6      return ans
```

```
1  def factorial():
2      while True:
3          try:
4              n = int(input("Enter a number: "))
5              assert n >= 0
6          except ValueError:
7              print("Not a number.")
8          except AssertionError:
9              print("Not a natural number.")
10         else:
11             print("factorial(", n, ")= ", fac(n), sep='')
12             print("Goodbye!")
13             break
```

사용자 정의 예외

사용자 정의 예외

```
1  def fac(n):
2      ans = 1
3      while n > 1:
4          ans = n * ans
5          n = n - 1
6      return ans
```

정의

발생

처리

```
1  class NonPositive(Exception): pass
2
3  def factorial():
4      while True:
5          try:
6              n = int(input("Enter a number: "))
7              if n < 1:
8                  raise NonPositive
9          except ValueError:
10             print("Not a number.")
11         except AssertionError:
12             print("Not a natural number.")
13         else:
14             print("factorial(", n, ")= ", fac(n), sep='')
15             print("Goodbye!")
16             break
```