>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

CHAPTER 10

예외 처리

예외

Exception

방어 프로그래밍
Defensive Programming

안전 코딩
Secure Coding

방어 프로그래밍
Defensive Programming

안전 코딩
Secure Coding

예외 처리
Exception Handling

CHAPTER 10

예외 처리

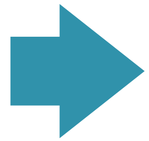
CHAPTER 10

예외 처리

내장 예외

Built-in Exception

예외 타입	발생 상황
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
FileNotFoundError	없는 파일을 열려고 하는 경우, 열지 않고 파일을 읽거나 쓰려고 하는 경우 등



>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

p.463

실습 10.1 예외 발생시켜 보기

ValueError, IndexError, KeyError 예외를 발생시키는 사례를 각각 실행창에서 만들어 확인하자.

CHAPTER 10

예외 처리

예외 처리

Exception Handling

code : 10-1.py

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

code : 10-2.py

```
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

Exception Handling

code : 10-1.py

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

code : 10-3.py

```
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
```

구문

구문 10.1 예외 처리문

```
try:
    ⟨블록⟩
except ⟨예외이름⟩1:
    ⟨블록⟩1
except ⟨예외이름⟩2:
    ⟨블록⟩2
…
except ⟨예외이름⟩n:
    ⟨블록⟩n
```

의미

의미 10.1 예외 처리문

try 블록인 ⟨블록⟩을 실행한다.

- 예외 상황이 발생하지 않고 ⟨블록⟩의 실행을 종료하면, except 블록은 모두 무시한다.
- ⟨블록⟩의 실행도중 예외 상황이 발생하면 try 블록의 남은 부분은 무시하고, ⟨예외이름⟩$_1$, ⟨예외이름⟩$_2$, …, ⟨예외이름⟩$_n$ 중에서 발생한 예외와 같은 이름(종류)을 위에서부터 나타나는 순서대로 찾아서 해당 블록을 실행한다. 여러 개의 예외처리 블록 중에서 가장 먼저 일치한 예외처리 블록 하나만 선택하여 실행한다.
- 발생한 예외와 같은 이름이 ⟨예외이름⟩$_1$, ⟨예외이름⟩$_2$, …, ⟨예외이름⟩$_n$ 중에 없으면 발생한 오류메시지를 내주면서 실행을 멈춘다.

예외 처리

Exception Handling

code : 10-4.py

```
while True:
    try:
        x = int(input("Enter a number: "))
        reciprocal = 1 / x
        print("The reciprocal of", x, "is", reciprocal)
        break
    except ValueError:
        print("Not a number.")
    except ZeroDivisionError:
        print("The reciprocal of 0 does not exist.")
        break
    except:
        print("Unexpected exception occurred.")
        break
```

예외 처리

Exception Handling

code : 10-5.py

```
while True:
    try:
        x = int(input("Enter a number: "))
        reciprocal = 1 / x
        print("The reciprocal of", x, "is", reciprocal)
        break
    except ValueError as message:
        print(message)
    except ZeroDivisionError as message:
        print(message)
        break
```

code : 10-3.py

```
while True:
    try:
        x = int(input("Enter a number: "))
        reciprocal = 1 / x
        print("The reciprocal of", x, "is", reciprocal)
        break
    except ValueError:
        print("Not a number.")
    except ZeroDivisionError:
        print("The reciprocal of
        break
```

code : 10-6.py

```
while True:
    try:
        x = int(input("Enter a number: "))
        reciprocal = 1 / x
    except ValueError:
        print("Not a number.")
    except ZeroDivisionError:
        print("The reciprocal of 0 does not exist.")
        break
    else:
        print("The reciprocal of", x, "is", reciprocal)
        break
```

code : 10-3.py

```
while True:
    try:
        x = int(input("Enter a number: "))
        reciprocal = 1 / x
        print("The reciprocal of
        break
    except ValueError:
        print("Not a number.")
    except ZeroDivisionError:
        print("The reciprocal of
        break
```

code : 10-7.py

```
while True:
    try:
        x = int(input("Enter a number: "))
        reciprocal = 1 / x
    except ValueError:
        print("Not a number.")
    except ZeroDivisionError:
        print("The reciprocal of 0 does not exist.")
        break
    else:
        print("The reciprocal of", x, "is", reciprocal)
        break
    finally:
        print(":-)")
```

p.471

실습 10.2 실수 입력 확인

3장의 〈실습 3.10〉에서는 키보드 입력이 고정소수점 수 형식의 실수인지 통과조건을 만들어 입력 문자열을 확인하였다. 이번에는 키보드 입력 문자열을 `float` 타입으로 변환하는 과정에서 `ValueError` 예외가 발생하면 예외 처리를 통하여 재입력을 받는 방식으로 입력 확인하는 `input_float()` 함수를 작성하자.

CHAPTER 10

예외 처리

구문 / 의미

구문 10.2 assert 문

assert 〈논리식〉

의미 10.2 assert 문

〈논리식〉의 계산 결과가 True이면 그냥 통과하고, False이면 AssertionError라는 이름의 예외를 발생시킨다.

code : 10-8.py

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

code : 10-9.py

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

code : 10-10.py

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

code : 10-10.py

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

code : 10-11.py

```
1  def factorial():
2      while True:
3          try:
4              n = int(input("Enter a number: "))
5              assert n >= 1
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
```

실습 10.3 조합 계산 서비스 구현

7.2절에서 완성한 조합 함수 comb_pascal을 사용하여 다음과 같이 두 자연수 n과 r을 키보드 입력받아 ${}_nC_r$을 계산하여 프린트해주는 프로그램을 작성하자. 이 프로그램은 실행창에서 키보드로 다음과 같이 소통하면서 작동해야 한다.

- 프로그램을 실행하면 먼저 다음 두 줄을 실행창에 프린트한다.

```
This program computes combination of two natural numbers, n and r.
Press Control+C to quit.
```

- 그리고 n 값과 r 값을 하나씩 다음과 같이 키보드에서 입력 받는다.

```
Enter n: 6
Enter r: 4
```

여기서 Enter n:과 Enter r:은 실행창에 보여주는 메시지이고, 6과 4는 키보드 입력이다.

실습 10.3 조합 계산 서비스 구현

- 키보드 입력이 정수가 아닌 경우 int 함수로 타입 변환하는 과정에서 ValueError 예외가 발생한다. 이 경우 예외 처리하여 다음과 같이 메시지를 실행창에 출력하며 재입력 받는다.

```
Must be a number.
```

- n 값과 r 값이 0 이상이 아니거나, r 값이 n 값보다 큰 경우 조합을 정의할 수 없다. 이 경우 assert 문을 활용하여 AssertionError 예외를 발생시키고, 이를 처리하여 다음과 같은 형식으로 메시지를 출력창에 출력하고 재입력 받는다.

```
4C6 is not defined.
```

pp.474~476

실습 10.3 조합 계산 서비스 구현

- comb_pascal(6,4)를 호출한 결과를 다음과 같은 형식으로 실행창에 프린트한다.

```
6C4 = 15
```

- n 값과 r 값을 키보드에서 입력받아 조합을 계산하여 프린트해주는 작업을 무한정 반복한다.
- 사용자가 Control+C 키를 동시에 누르면 다음을 실행창에 프린트하고 프로그램을 종료한다. (KeyboardInterrupt 예외 처리 활용)

```
Goodbye!
```

CHAPTER 10

예외 처리

사용자 정의 예외

User-defined Exception

code : 10-12.py

예외 정의 →

예외 발생 →

예외 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 NonPositive:
12             print("Not a natural number.")
13         else:
14             print("factorial(", n, ") = ", fac(n), sep='')
15             print("Goodbye!")
16             break
```

실습 10.4 실수 입력 확인 (범위 제한)

〈실습 10.2〉에서 작성한 input_float() 함수를 변형하여, −1.0~1.0 범위의 수만 입력을 허용하도록 제한하는 input_float_one() 함수를 작성하자. 범위의 양쪽 끝은 허용하며, 사용자 정의 예외를 하나 만들어 써야 한다.

>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

CHAPTER 10

예외 처리