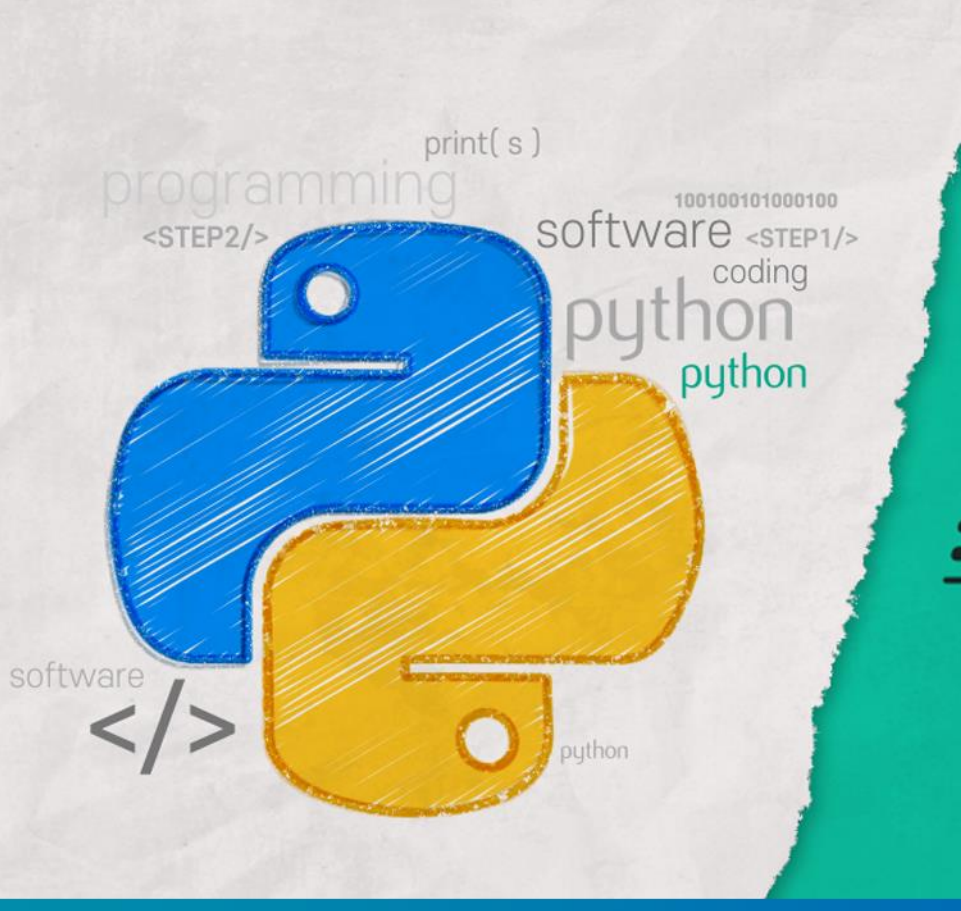


파이썬 </> 프로그래밍

파이썬의 예외 처리

한국기술교육대학교
온라인평생교육원

학습목표

- **오류 발생 종류**에 대해 알고, 예외 처리를 할 수 있다.
- **여러 가지 예외 처리 방법**에 대해 알고, 상황에 맞는 예외를 적용할 수 있다.

학습내용

- 예외의 종류와 예외 처리
- 예외 처리 방법

예외의 종류와 예외 처리

1 예외의 종류

① 프로그래밍 언어의 오류

구문 오류(Syntax Error)

- **프로그램 실행 전**에 발생하는 오류
- 이클립스, 파이참 등 통합개발환경 도구에서는 자동으로 실행 전에 오류를 체크함

논리적 오류(Logical Error) 혹은 런타임 오류(Runtime Error)

- **프로그램 실행 중**에 발생하는 오류
- 문법적으로 틀린 것이 없으므로, 즉시 인식되지 않지만 의도치 않은 결과를 초래할 수 있음

```
print (1
```

```
  File "<ipython-input-54-8f25d5090095>", line 1
    print (1
           ^
SyntaxError: unexpected EOF while parsing
```

예외의 종류와 예외 처리

1 예외의 종류

② 예외 발생의 예시

```
a = 10
b = 0
c = a / b
```

```
---------------------------------------
ZeroDivisionError
<ipython-input-49-41671660f656> in <
      1 a = 10
      2 b = 0
----> 3 c = a / b

ZeroDivisionError: division by zero
```

예외의 종류와 예외 처리

1 예외의 종류

2 예외 발생의 예시

```
4 + new*3
```

```
---------------------------------------
NameError
<ipython-input-50-730367ce567f> in <mo
----> 1 4 + new*3

NameError: name 'new' is not defined
```

```
print('2' + 2)
```

```
---------------------------------------
TypeError
<ipython-input-52-f01985f6a385> i
----> 1 print('2' + 2)

TypeError: must be str, not int
```

예외의 종류와 예외 처리

1 예외의 종류

2 예외 발생의 예시

```
l = [1, 2]
print(l[2])
```

```
---------------------------------------
IndexError
<ipython-input-53-08b5b9cc3f1f> in <
      1 l = [1, 2]
----> 2 print(l[2])

IndexError: list index out of range
```

```
d = {"a": 1, "b": 2}
print(d['c'])
```

```
-----------------------------
KeyError
<ipython-input-55-e0f66c508f6
      1 d = {"a": 1, "b": 2}
----> 2 print(d['c'])

KeyError: 'c'
```

예외의 종류와 예외 처리

1 예외의 종류

③ 파이썬 내장 예외 종류

https://docs.python.org/3/library/exceptions.html

- StopIteration, ImportError, NameError, SyntaxError …
- 계층 구조로 이루어져 있음

예외의 종류와 예외 처리

1 예외의 종류

3 파이썬 내장 예외 종류

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      |    +-- ModuleNotFoundError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
      +-- OSError
```

```
+-- OSError
|    +-- BlockingIOError
|    +-- ChildProcessError
|    +-- ConnectionError
|    |    +-- BrokenPipeError
|    |    +-- ConnectionAbortedError
|    |    +-- ConnectionRefusedError
|    |    +-- ConnectionResetError
|    +-- FileExistsError
|    +-- FileNotFoundError
|    +-- InterruptedError
|    +-- IsADirectoryError
|    +-- NotADirectoryError
|    +-- PermissionError
|    +-- ProcessLookupError
|    +-- TimeoutError
+-- ReferenceError
+-- RuntimeError
|    +-- NotImplementedError
|    +-- RecursionError
+-- SyntaxError
|    +-- IndentationError
|         +-- TabError
```

```
+-- SystemError
+-- TypeError
+-- ValueError
|    +-- UnicodeError
|         +-- UnicodeDecodeError
|         +-- UnicodeEncodeError
|         +-- UnicodeTranslateError
+-- Warning
     +-- DeprecationWarning
     +-- PendingDeprecationWarning
     +-- RuntimeWarning
     +-- SyntaxWarning
     +-- UserWarning
     +-- FutureWarning
     +-- ImportWarning
     +-- UnicodeWarning
     +-- BytesWarning
     +-- ResourceWarning
```

예외의 종류와 예외 처리

2 예외 처리

① 원의 둘레 및 넓이를 구하는 프로그램

정수를 입력하지 않는다면?

```
a = input("정수 입력 : ")
a = float(a)
print("반지름 : ", a)
print("둘레 : ", 2 * 3.14 * a)
print("넓이 :", 3.14 * a * a)
```

```
정수 입력 : 5
반지름 :  5.0
둘레 :  31.400000000000002
넓이 : 78.5
```

```
a = input("정수 입력 : ")
a = float(a)
print("반지름 : ", a)
print("둘레 : ", 2 * 3.14 * a)
print("넓이 :", 3.14 * a * a)
```

```
정수 입력 : 정수

---------------------------------------------------------
ValueError                                Traceback (
<ipython-input-61-3565f7dd6917> in <module>()
      1 a = input("정수 입력 : ")
----> 2 a = float(a)
      3 print("반지름 : ", a)
      4 print("둘레 : ", 2 * 3.14 * a)
      5 print("넓이 :", 3.14 * a * a)

ValueError: could not convert string to float: '정수'
```

예외의 종류와 예외 처리

2 예외 처리

① 원의 둘레 및 넓이를 구하는 프로그램

조건문을 사용한 예외 처리

```
a = input("정수 입력 : ")
if a.isdigit():
    a = float(a)
    print("반지름 : ", a)
    print("둘레 : ", 2 * 3.14 * a)
    print("넓이 :", 3.14 * a * a)

else:
    print("예외 처리되었습니다.")
```

```
정수 입력 : 정수
예외 처리되었습니다.
```

만약, 0이 입력된다면? 음수가 입력된다면? …

수많은 조건문 필요!

예외 처리 방법

1 예외 처리 방법

1 try, except 구문

- 예외가 발생할 수 있는 상황을 예상하여 예외를 제어할 수 있음

- try : (예외 발생 가능한) 일반적인 수행문들
- except : 예외가 발생하였을 때 수행문들

```
a = input("정수 입력 : ")

try:
    a = float(a)
    print("반지름 : ", a)
    print("둘레 : ", 2 * 3.14 * a)
    print("넓이 :", 3.14 * a * a)
except:
    print("예외 발생!!!")
```

```
정수 입력 : 정수
예외 발생!!!
```

예외 처리 방법

1 예외 처리 방법

1 try, except 구문

- 프로그램 실행에 치명적이지 않은 오류지만,
  오류로 인해 프로그램 실행이 멈추는 것을 방지하려면?
  ➡ 예외를 그냥 넘어가고 싶은 경우 pass 키워드 사용

```
num = ["3",'안녕하세요','4',2,67,'python']
digit_num = []

for i in num:
    try:
        digit_num.append(int(i))
    except:
        pass
print(digit_num)
```

```
[3, 4, 2, 67]
```

예외 처리 응용
리스트에서
숫자만 뽑아내고
싶은 경우

예외 처리 방법

1 예외 처리 방법

2 try, except, else 구문

- try, except 뒤에 else를 붙여서 사용하면, 예외가 발생하지 않았을 때 실행할 코드를 지정할 수 있음

  ➡ 예외가 발생할 수 있는 코드만 try에 넣어서 활용 (가독성, 유지보수 향상)

```
a = input("정수 입력 : ")

try:
    a = float(a)
except:
    print("예외 발생 !!")
else:
    print("반지름 : ", a)
    print("둘레 : ", 2 * 3.14 * a)
    print("넓이 :", 3.14 * a * a)
```

```
정수 입력 : 정수
예외 발생 !!
```

```
a = input("정수 입력 : ")

try:
    a = float(a)
except:
    print("예외 발생 !!")
else:
    print("반지름 : ", a)
    print("둘레 : ", 2 * 3.14 * a)
    print("넓이 :", 3.14 * a * a)
```

```
정수 입력 : 4
반지름 :  4.0
둘레 :  25.12
넓이 : 50.24
```

예외 처리 방법

1 예외 처리 방법

3 try, except, else, finally 구문

- trytry, except, else 뒤에 finally를 붙여서 사용하면, 예외 발생 유무에 관계 없이 실행되는 코드를 작성할 수 있음

```
a = input("정수 입력 : ")

try:
    a = float(a)
except:
    print("예외 발생 !!")
else:
    print("반지름 : ", a)
    print("둘레 : ", 2 * 3.14 * a)
    print("넓이 :", 3.14 * a * a)
finally:
    print("프로그램이 종료되었습니다.")
```

```
정수 입력 : 정수
예외 발생 !!
프로그램이 종료되었습니다.
```

```
a = input("정수 입력 : ")

try:
    a = float(a)
except:
    print("예외 발생 !!")
else:
    print("반지름 : ", a)
    print("둘레 : ", 2 * 3.14 * a)
    print("넓이 :", 3.14 * a * a)
finally:
    print("프로그램이 종료되었습니다.")
```

```
정수 입력 : 4
반지름 :  4.0
둘레 :  25.12
넓이 : 50.24
프로그램이 종료되었습니다.
```

else, finally는
꼭 활용하지 않아도
다른 방식으로
프로그래밍 가능

예외 처리 방법

1 예외 처리 방법

3 try, except, else, finally 구문

- 규칙
  - try 구문은 단독으로 사용할 수 없음
  - else 구문은 반드시 except 구문 뒤에 와야 함
- 아래 조합 외에는 오류 발생
  - try + except
  - try + except + else
  - try + except + finally
  - try + except + else + finally
  - try + finally

```
try:
    print("dd")
else:
    print(1)
```

```
  File "<ipython-input-80-94d8
    else:
       ^
SyntaxError: invalid syntax
```

예외 처리 방법

2 예외 객체와 예외 구분

① 예외 객체

예외가 발생하면, 예외와 관련된 정보가 생성 (예외 객체로 활용 가능)

- try : (예외 발생 가능한) 일반적인 수행문들
- except 예외의 종류 as 예외 객체를 활용할 변수 : 예외가 발생했을 때 수행문들

예외의 종류를 모르겠다면, exception(모든 예외 포함)

```
a = 0.0

try:
    print(1.0 / a)
except ZeroDivisionError as msg:
    print('ZeroDivisionError !!!')
```

```
ZeroDivisionError !!!
```

```
print(msg)
```

```
---------------------------------------
NameError
<ipython-input-84-8a886db20f58> in <mod
----> 1 print(msg)

NameError: name 'msg' is not defined
```

예외 처리 방법

2 예외 객체와 예외 구분

② 예외 구분

예외 객체를 활용해 조건문처럼 예외 종류에 따라 다른 코딩을 할 수 있음

- try : (예외 발생 가능한) 일반적인 수행문들
- except 예외 종류 A : 예외가 발생했을 때 수행문들
- except 예외 종류 B : 예외가 발생했을 때 수행문들
- except 예외 종류 C : 예외가 발생했을 때 수행문들

예외 처리 방법

2 예외 객체와 예외 구분

② 예외 구분

```
num = ["3",'안녕하세요','4',2,67,'python']
try:
    a = input("정수를 입력 해주세요 : ")
    a = int(a)
    print(a,'번째 요소는 : ', num[a], '입니다.')
except ValueError:
    print("정수를 입력해주세요")

except IndexError:
    print("리스트의 범위를 벗어났습니다.")
```

```
정수를 입력 해주세요 : 정수
정수를 입력해주세요
```

```
num = ["3",'안녕하세요','4',2,67,'python']
try:
    a = input("정수를 입력 해주세요 : ")
    a = int(a)
    print(a,'번째 요소는 : ', num[a], '입니다.')
except ValueError:
    print("정수를 입력해주세요")

except IndexError:
    print("리스트의 범위를 벗어났습니다.")
```

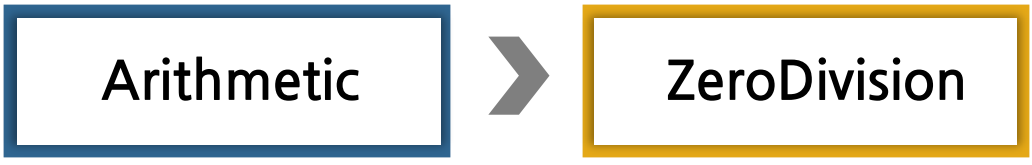
```
정수를 입력 해주세요 : 61
리스트의 범위를 벗어났습니다.
```

예외 처리 방법

2 예외 객체와 예외 구분

③ 예외 구분의 잘못된 예

- 예외 처리의 순서(예외의 포함 관계)




```
try:
    print(1/0)
except ArithmeticError:
    print("ArithmeticException occured")
except ZeroDivisionError:
    print("ZeroDivisionError occured"   )
```

```
ArithmeticException occured
```

```
ArithmeticError
 +-- FloatingPointError
 +-- OverflowError
 +-- ZeroDivisionError
```

예외 처리 방법

2 예외 객체와 예외 구분

④ 강제로 예외 발생시키기

- raise 예외 종류(메시지)

1. 사용자 정의 클래스를 만들 때
(연산자 오버로딩 등)

2. 아직 구현이 덜 된 코드

3. 그 외 문법적으로 정상적인 코드지만
예외 처리가 필요한 경우

예외 처리 방법

2 예외 객체와 예외 구분

④ 강제로 예외 발생시키기

- raise 예외 종류(메시지)

```
a = input("정수 입력 : ")
a = int(a)

if (a > 0):
    raise Exception
else :
    print("음수입니다.")
```

```
정수 입력 : -1
음수입니다.
```

```
a = input("정수 입력 : ")
a = int(a)

if (a > 0):
    raise Exception('message!!!')
else :
    print("음수입니다.")
```

```
정수 입력 : 4

---------------------------------------------
Exception
<ipython-input-95-63e8c0896c43> in <module
      3
      4 if (a > 0):
----> 5     raise Exception('message!!!')
      6 else :
      7     print("음수입니다.")

Exception: message!!!
```

예외 처리 방법

2 예외 객체와 예외 구분

5 예외 처리 활용

```
num = ["3",'안녕하세요','4',2,67,'python']
try:
    a = input("정수를 입력 해주세요 : ")
    a = int(a)
    print(a,'번째 요소는 : ', num[a], '입니다.')
except ValueError as msg:
    print("정수를 입력해주세요")

except IndexError as msg:
    print("리스트의 범위를 벗어났습니다.")

except Exception as msg:
    print("예상하지 못한 오류가 발생했습니다.")

else:
    print("파이썬 프로그래밍")
finally:
    print("프로그램이 종료되었습니다.")
```

```
정수를 입력 해주세요 : 4
4 번째 요소는 :  67 입니다.
파이썬 프로그래밍
프로그램이 종료되었습니다.
```

예외 처리에서 가장 중요한 것

➡ **이 코드에서 어떤 경우에 어떤 예외가 발생할 것인가?**

Run! 프로그래밍

Mission 정수를 입력 받아 구구단을 출력하는 프로그램 작성

```
num = input ("2~ 9 사이의 숫자를 입력해주세요 : ")

try :
    if(num.isdigit()):
        num = int(num)
        print("-----------------")
except Exception as msg:
    print(msg)

else:
    if(num > 9):
        num = 9
    elif(num<2):
        num = 2
    else:
        print(num,"단을 출력합니다.")
        print()
        for i in range(1,10):
            print("{} X {} = {}".format(num,i,num*i))
finally:
    print("-----------------")
```

학습정리

1. 예외의 종류와 예외 처리

예외의 종류	• 프로그램 오류는 크게 두 가지로 구문 오류와 논리적 오류로 나뉨 • 예외란 논리적 오류에 해당되며 문법적으론 문제가 없으나 의도하지 않은 결과를 나타낼 수 있음
예외 처리	• 조건문으로도 할 수 있으나, 수많은 경우의 수를 생각하기 힘들뿐더러 코드의 유지보수 측면에서도 좋지 않음

2. 예외 처리 방법

예외 처리	• 예외에는 여러가지 종류가 있으며, 예외 발생 시 해당 예외 객체에 담아 사용할 수 있음 • try, except, else, finally 구문을 이용해 예외 발생 전, 예외 발생, 예외 발생 후, 예외에 상관 없이 각각 경우에 따라 프로그램을 실행할 수 있음 • 예외는 예외 종류에 따라 여러 개를 정의할 수 있으며, 이 때 예외 포함 관계에 따라 순서를 잘 적어주어야 함 • else와 finally 구문은 꼭 사용하지 않아도 되지만, 프로그램의 가독성, 유지보수, 재사용 측면에서 사용하면 편리함
예외 객체와 예외 구분	• 필요에 따라 raise 키워드를 사용해 예외를 강제로 발생시킬 수도 있음