

예외 고급

목차

- 시작하기 전에
- 예외 객체
- 예외 구분하기
- 모든 예외 잡기
- raise 구문
- 키워드로 정리하는 핵심 포인트
- 확인문제

[핵심 키워드] : 예외 객체, raise 구문, GitHub 검색

[핵심 포인트]

프로그램을 개발하다 보면 수많은 오류를 만나게 된다. 또한 처음 프로그램을 개발했을 때 모든 오류를 예측하고 처리하는 경우는 거의 없다. 개발이 완료된 뒤에도 예측하지 못한 오류들이 계속 발생하기 때문에 유지보수가 필요하다.

- 예외 객체 (exception object)
 - 예외 발생 시 예외 정보가 저장되는 곳

try:

예외가 발생할 가능성이 있는 구문

except 예외의 종류 as 예외 객체를 활용할 변수 이름:

예외가 발생했을 때 실행할 구문

- Exception

- “모든 예외의 어머니”
- 예시

```
01  # try except 구문으로 예외를 처리합니다.
02  try:
03      # 숫자로 변환합니다.
04      number_input_a = int(input("정수 입력> "))
05      # 출력합니다.
06      print("원의 반지름:", number_input_a)
07      print("원의 둘레:", 2 * 3.14 * number_input_a)
08      print("원의 넓이:", 3.14 * number_input_a * number_input_a)
09  except Exception as exception:
10      # 예외 객체를 출력해봅니다.
11      print("type(exception):", type(exception))
12      print("exception:", exception)
```

```
정수 입력> yes!! 
```

```
type(exception): <class 'ValueError'>
```

```
exception: invalid literal for int() with base 10: 'yes!!'
```

- 다양한 예외들이 발생할 때 그 정보를 메일 등으로 보내도록 해서 수집하면 큰 규모의 웹서비스 등에서 프로그램 개선에 큰 도움이 됨

예외 구분하기

- 예외 객체 사용하면 except 구문을 if 조건문처럼 사용해서 예외를 구분할 수 있음
- 여러 가지 예외가 발생할 수 있는 상황

```
01  # 변수를 선언합니다.
02  list_number = [52, 273, 32, 72, 100]
03
04  # try except 구문으로 예외를 처리합니다.
05  try:
06      # 숫자를 입력받습니다.
07      number_input = int(input("정수 입력> "))
08      # 리스트의 요소를 출력합니다.
09      print("{}번째 요소: {}".format(number_input, list_number[number_input]))
10  except Exception as exception:
11      # 예외 객체를 출력해봅니다.
12      print("type(exception):", type(exception))
13      print("exception:", exception)
```

- 정상적으로 정수 입력한 경우

```
정수 입력> 2 [Enter]
```

```
2번째 요소: 32
```

- 정수로 변환될 수 없는 값 입력한 경우 - ValueError

```
정수 입력> yes!! [Enter]
```

```
type(exception): <class 'ValueError'>
```

```
exception: invalid literal for int() with base 10: 'yes!!'
```

- 정수 입력하나 리스트 길이를 넘는 인덱스인 경우 - IndexError

```
정수 입력> 100 [Enter]
```

```
type(exception): <class 'IndexError'>
```

```
exception: list index out of range
```


- 예외 구분하기

- except 구문 뒤에 예외 종류 입력해서 구분할 수 있음

```
try:
```

```
    예외가 발생할 가능성이 있는 구문
```

```
except 예외의 종류A:
```

```
    예외A가 발생했을 때 실행할 구문
```

```
except 예외의 종류B:
```

```
    예외B가 발생했을 때 실행할 구문
```

```
except 예외의 종류C:
```

```
    예외C가 발생했을 때 실행할 구문
```

– 예시 – ValueError와 IndexError

```
01  # 변수를 선언합니다.
02  list_number = [52, 273, 32, 72, 100]
03
04  # try except 구문으로 예외를 처리합니다.
05  try:
06      # 숫자를 입력받습니다.
07      number_input = int(input("정수 입력> "))
08      # 리스트의 요소를 출력합니다.
09      print("{}번째 요소: {}".format(number_input, list_number[number_input]))
10  except ValueError:
11      # ValueError가 발생하는 경우
12      print("정수를 입력해 주세요!")
13  except IndexError:
14      # IndexError가 발생하는 경우
15      print("리스트의 인덱스를 벗어났어요!")
```

- 정수 아닌 값 입력해 ValueError 발생시키는 경우

정수 입력> yes!!

정수를 입력해 주세요!

- 리스트의 인덱스를 넘는 숫자 입력해 IndexError인 경우

정수 입력> 100

리스트의 인덱스를 벗어났어요!

- 예외 구분 구문과 예외 객체
 - as 키워드 사용
 - 각각의 except 구문 뒤에 예외 객체 붙여 예외 구분에 활용

```
01  # 변수를 선언합니다.
02  list_number = [52, 273, 32, 72, 100]
03
04  # try except 구문으로 예외를 처리합니다.
05  try:
06      # 숫자를 입력 받습니다.
07      number_input = int(input("정수 입력> "))
08      # 리스트의 요소를 출력합니다.
09      print("{}번째 요소: {}".format(number_input, list_number[number_input]))
10  except ValueError as exception:
11      # ValueError가 발생하는 경우
12      print("정수를 입력해 주세요!")
13      print("exception:", exception)
14  except IndexError as exception:
15      # IndexError가 발생하는 경우
16      print("리스트의 인덱스를 벗어났어요!")
17      print("exception:", exception)
```

- 코드 실행 후 인덱스 벗어나는 숫자 입력하여 IndexError 발생시킬 경우

정수 입력> 100

리스트의 인덱스를 벗어났어요!

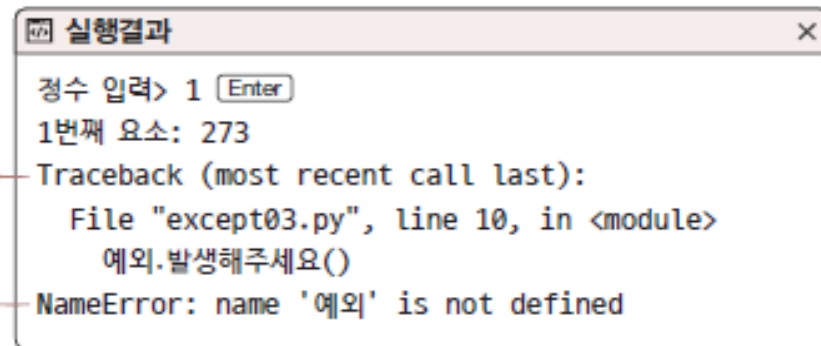
exception: list index out of range

- except 구문으로 예외 구분하면 if, elif, else 조건문처럼 차례대로 오류 검출하며 확인. 만약 예외 조건에 일치하는 것이 없다면 당연히 예외가 발생하며 프로그램이 강제 종료
 - 예시 - 예외 처리를 했지만 예외를 못 잡는 경우

```
01  # 변수를 선언합니다.
02  list_number = [52, 273, 32, 72, 100]
03
04  # try except 구문으로 예외를 처리합니다.
05  try:
06      # 숫자를 입력받습니다.
07      number_input = int(input("정수 입력> "))
08      # 리스트의 요소를 출력합니다.
09      print("{}번째 요소: {}".format(number_input, list_number[number_input]))
10      예외.발생해주세요() → 이 부분에서 잡지 않은 예외가 발생합니다.
```

```
11 except ValueError as exception:
12     # ValueError가 발생하는 경우
13     print("정수를 입력해 주세요!")
14     print(type(exception), exception)
15 except IndexError as exception:
16     # IndexError가 발생하는 경우
17     print("리스트의 인덱스를 벗어났어요!")
18     print(type(exception), exception)
```

try except 구문을 사용했는데도
프로그램이 죽어 버렸어요!



```
실행결과
정수 입력> 1 Enter
1번째 요소: 273
Traceback (most recent call last):
  File "except03.py", line 10, in <module>
    예외.발생해주세요()
NameError: name '예외' is not defined
```

- 예외가 발생해 프로그램이 강제 종료
- else 구문처럼 마지막에 Exception 넣어서 프로그램 죽지 않게 하는 것이 좋음

- 예시

```
01  # 변수를 선언합니다.
02  list_number = [52, 273, 32, 72, 100]
03
04  # try except 구문으로 예외를 처리합니다.
05  try:
06      # 숫자를 입력 받습니다.
07      number_input = int(input("정수 입력> "))
08      # 리스트의 요소를 출력합니다.
09      print("{}번째 요소: {}".format(number_input, list_number[number_input]))
10      예외.발생해주세요()
11  except ValueError as exception:
```



```
11 except ValueError as exception:
12     # ValueError가 발생하는 경우
13     print("정수를 입력해 주세요!")
14     print(type(exception), exception)
15 except IndexError as exception:
16     # IndexError가 발생하는 경우
17     print("리스트의 인덱스를 벗어났어요!")
18     print(type(exception), exception)
19 except Exception as exception: → ValueError와 IndexError가 아닌 예외가 발생했을 때
20     # 이외의 예외가 발생한 경우 실행됩니다.
21     print("미리 파악하지 못한 예외가 발생했습니다.")
22     print(type(exception), exception)
```

실행결과

```
정수 입력> 1 
1번째 요소: 273
미리 파악하지 못한 예외가 발생했습니다.
<class 'NameError'> name '예외' is not defined
```

- raise 키워드

- 예외를 강제로 발생시킴
- 프로그램 개발 단계에서 아직 구현되지 않은 부분에 일부러 예외를 발생시켜 잊어버리지 않도록 함

raise 예외 객체

```
# 입력을 받습니다.  
number = input("정수 입력> ")  
number = int(number)  
  
# 조건문 사용  
if number > 0:  
    # 양수일 때: 아직 미구현 상태입니다.  
    raise NotImplementedError  
else:  
    # 음수일 때: 아직 미구현 상태입니다.  
    raise NotImplementedError
```

- **예외 객체** : 예외와 관련된 정보 담고 있는 객체
- **raise 구문** : 예외 강제로 발생시킬 때 사용하는 구문
- **GitHub 검색** : 많은 사람이 함께 개발하는 소셜 코딩 사이트 GitHub 이용하는 것으로, 유능한 개발자들의 정제된 코드 살펴볼 수 있음

- 예외를 강제로 발생시킬 때 사용하는 키워드로 맞는 것은 무엇일까요?
 - throw
 - raise
 - runtime
 - Error
- 본문에서 살펴보았던 GitHub에서 코드를 찾는 방법으로, 인공지능 개발에서 많이 사용되는 수치 연산 라이브러리, 텐서플로우 (<http://github.com/tensorflow/tenseflow>)에서 raise 구문이 사용되는 예를 세 가지 찾아보세요.

- 예시

```
raise ValueError(  
    'incompatible dtype; specified: {}, inferred from {}: {}'.format(  
        element_dtype, elements, inferred_dtype))
```

```
raise ValueError(  
    'element shape may not be specified when creating list from tensor')
```

```
raise NotImplementedError('tensor lists only support removing from the end')
```