

Ask Django

예외 (Exceptions)

예외 (Exceptions)

- 프로그램이 처리되는 동안 특정한 문제(예외)가 일어났을 때 진행 중인 루틴을 중단하고, 콜스택을 거슬러 올라가, 예외를 전파하는 메커니즘
 - 이 예외를 처리할 수 있는 핸들러를 찾아, **함수 호출 역순**으로 거슬러 올라가, 본 예외를 처리할 수 있는 핸들러를 찾아내면 그 곳에 처리를 맡긴다.
 - 예외를 처리하는 핸들러를 찾을 수 없다면, 그 즉시 파이썬 프로그램이 비정상종료되며, 그 예외 내역에 대한 StackTrace¹를 출력

```
print('line 1')
value = int('a') + 1
print('line 2')
```

위 코드를 실행하면, ValueError 예외가 발생하며, 코드 실행이 중단

```
line 1
-----
ValueError                                Traceback (most recent call last)
<ipython-input-2-b26f9034fbc9> in <module>()
----> 1 value = int('a') + 1

ValueError: invalid literal for int() with base 10: 'a'
```

¹ 프로그램 실행 중 오류 시점에서의 오류 정보

예외 잡기

```
print('line 1')
try:
    value = int('a') + 1
except ValueError as e:
    print(e)
print('line 2')
```

실행결과 : 예외를 정확하게 잡아서 처리했기 때문에, 다음 루틴이 이어서 계속 실행

```
line 1
invalid literal for int() with base 10: 'a'
line 2
```

호출한 함수 내에서 발생한 예외도 잡을 수 있습니다.

```
def fn1(x, y):  
    return x + y  
  
def fn2(a, b):  
    return 10 * fn1(a, b)  
  
try:  
    print(fn2('a', 10))  
except TypeError as e:  
    print(e)
```

실행결과

```
must be str, not int
```

흔히 만나는 빌트인 예외

- Exception : 최상위 예외 클래스
- StopIteration
 - Iterator 내에서 더 이상 생산할 Item이 없을 때
 - **for in** 구문에서 이 예외를 통해 반복문 종단을 처리
- AttributeError : attribute 참조 실패 혹은 설정이 실패한 경우
- ImportError : 지정 모듈/패키지를 import 하지 못한 경우

- NotImplementedError : 구현하지 않은 부분임을 명시할 때, 개발자가 직접 본 예외를 발생 (raise)
- IndexError : 범위 밖의 인덱스 참조시
- KeyError : 존재하지 않는 Key에 접근시
- NameError : local/global name 을 찾지못한 경우
- TypeError : 부적절한 연산/함수를 적용했을 때, ex) 1 + '1'
- ValueError : 부적절한 값을 발견했을 때, ex) int('a')
- IndentationError : 소스코드 내에 부적절한 들여쓰기가 있을 때

NotImplementedError 예시

아직 구현하지 않은 부분임을 명시

```
class Person:
    def run(self):
        raise NotImplementedError

class Doctor:
    pass

class Developer(Person):
    def run(self):
        print('개발자는 오늘도 뵙니다.')
```

```
Doctor().run()      # NotImplementedError 발생. 자식 클래스에게 run함수구현(Overriding)을 강제하는 효과
Developer().run()   # 예외없이 정상적으로 수행
```

예외처리

- tuple 로 예외를 다수 지정할 수 있습니다.
- as 를 통해 예외 인스턴스를 획득 가능
- else : 예외가 발생하지 않았을 때 호출되는 블록
- finally : 예외 발생 유무에 상관없이 호출되는 블록


```
try:
    some_code()
    some_code()
    some_code()
except ValueError:
    print('ValueError 가 발생했어요. ')
except (KeyError, TypeError):
    print('ValueError/TypeError 중에 하나가 발생')
except ZeroDivisionError as e:
    print('0으로 나누지마세요. : {}'.format(e))
else:
    print('예외가 발생하지 않았어요. ')
finally:
    print('예외발생 유무에 상관없이 호출됩니다. ')
```

다수 예외를 한 번에 처리

예외 인스턴스 획득

사용자 예외 정의 (1)

아래 코드에서는 **Big/Small** 분기 처리가 어렵습니다.

```
def fn(i):  
    if i > 100:  
        raise ValueError('Too Big Number : {}'.format(i))  
    elif i < -100:  
        raise ValueError('Too Small Number : {}'.format(i))  
    return i * 10  
  
try:  
    fn(210)  
except ValueError as e:  
    print(e)
```

사용자 예외 정의 (2)

손쉬운 예외 분류를 위해, 사용자 예외 정의


```
class TooBigNumberException(ValueError):
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return 'too big number {}'.format(self.value)

class TooSmallNumberException(ValueError):
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return 'too small number {}'.format(self.value)
```

```
def fn(i):  
    if i > 100:  
        raise TooBigNumberException(i)  
    elif i < -100:  
        raise TooSmallNumberException(i)  
    return i * 10  
  
try:  
    fn(200)  
except TooSmallNumberException as e:  
    print(e)  
except TooBigNumberException as e:  
    print(e)
```

A person wearing a red shirt and dark shorts is climbing a steep, rocky mountain. The scene is set during sunset or sunrise, with a warm, golden light illuminating the sky and the person's shirt. The person is positioned in the lower right quadrant of the frame, reaching up towards the rocky peak. The background shows the rugged silhouette of the mountain against the bright sky.

*Life is short,
use Python3/Django.*