

Day 10 - Exception in Python

예외가 발생하는 경우:

ValueError

- 잘못된 값을 함수나 연산에 제공할 때 발생합니다.
- 예) 숫자가 아닌 문자열을 int() 함수로 변환하려고 할 때 발생.

TypeError

- 올바르지 않은 유형의 객체를 연산에 사용하려 할 때 발생합니다.
- 예) 문자열과 숫자를 함께 더하려고 할 때 발생.

ZeroDivisionError

- 숫자를 0으로 나누려고 할 때 발생합니다.

IndexError

- 리스트, 튜플, 문자열 등의 시퀀스 유형에서 범위를 벗어난 인덱스에 접근하려 할 때 발생합니다.
- 예) 길이가 3인 리스트에 대해 4번째 요소에 접근하려고 할 때 발생.

KeyError

- 딕셔너리에서 존재하지 않는 키를 사용하여 값을 검색하려고 할 때 발생합니다.

AttributeError

- 객체에 없는 속성이나 메서드에 접근하려고 할 때 발생합니다.

FileNotFoundError

- 존재하지 않는 파일을 열려고 할 때 발생합니다.

ImportError

- 존재하지 않는 모듈을 가져오려고 할 때 또는 모듈 내에 해당 속성/함수가 없을 때 발생합니다.

NameError

- 정의되지 않은 변수나 함수를 사용하려고 할 때 발생합니다.
- 예) 프로그램에서 정의되지 않은 변수 x를 사용하려고 할 때 발생.

OverflowError

- 수치 연산 결과가 너무 커서 표현할 수 없을 때 발생합니다.

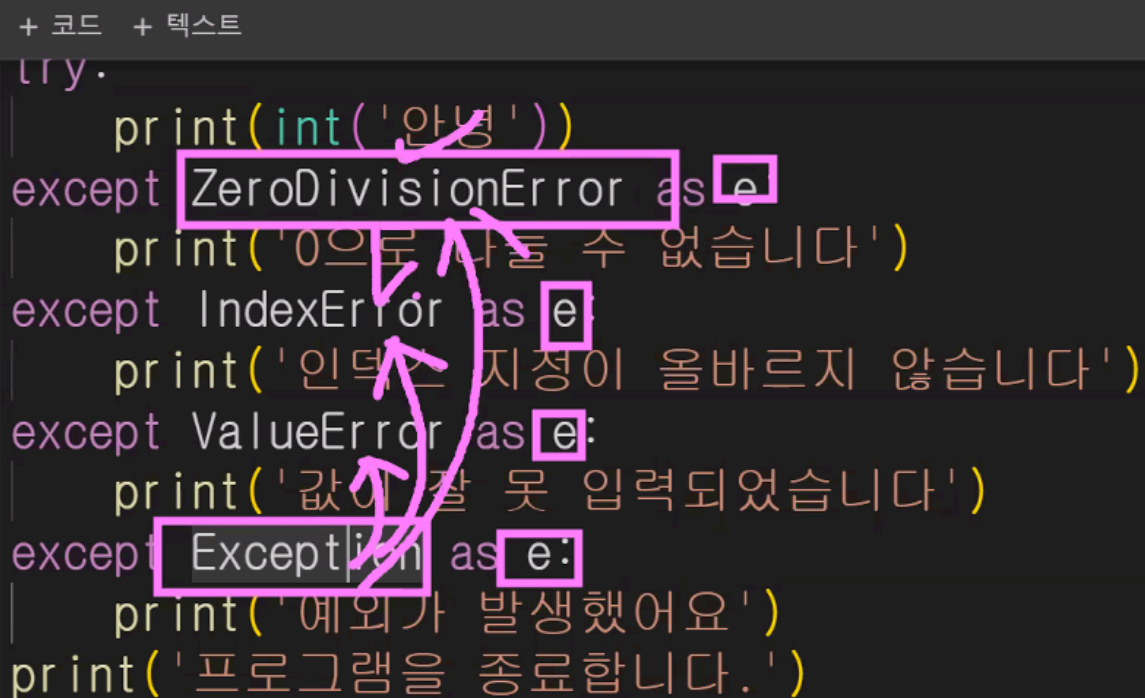
MemoryError

- 프로그램이 사용 가능한 모든 메모리를 소진했을 때 발생합니다.

예외 처리 기본 구조:

```
try:
    # 예외가 발생할 가능성이 있는 코드
except ExceptionType1: # 'ExceptionType1'에는 실제 예외 유형이 들어갑니다.
    # ExceptionType1 예외가 발생했을 때 실행될 코드
except ExceptionType2: # 'ExceptionType2'에는 다른 예외 유형이 들어갑니다.
    # ExceptionType2 예외가 발생했을 때 실행될 코드
# 추가적인 except 블록을 계속 추가할 수 있습니다.
else:
    # try 블록에서 예외가 발생하지 않았을 때 실행될 코드
finally:
    # 예외 발생 여부와 관계없이 항상 실행될 코드
```

Exception Class:



```
+ 코드 + 텍스트
try:
    print(int('안녕'))
except ZeroDivisionError as e:
    print('0으로 나눌 수 없습니다')
except IndexError as e:
    print('인덱스 지정이 올바르지 않습니다')
except ValueError as e:
    print('값이 잘 못 입력되었습니다')
except Exception as e:
    print('예외가 발생했어요')
print('프로그램을 종료합니다.')
```

- 모든 오류 개체의 아버지는 Exception class 이다.

Exception as e:

```
try:
    print(int('안녕'))
except ZeroDivisionError as e:
    print(e)
except IndexError as e:
    print(e)
except ValueError as e:
    print(e)
except Exception as e:
    print(e)
print('프로그램을 종료합니다.')
```

```
invalid literal for int() with base 10: '안녕'
프로그램을 종료합니다.
```

- e를 출력하면 exception class의 `__string__`의 메시지가 찍히게 된다.

else:

```
try :
    # print(int('안녕'))
    data = [10, 20, 30, 40, 50]
    print(data[0])
except ZeroDivisionError as e: # ZeroDivisionError 에러가 났을때만 실행
    print(e)
except IndexError as e: # IndexError 에러가 났을때만 실행
    print(e)
except ValueError as e:
    print(e)
except Exception as e:
    print(e)
else:
    print('에러가 발생하지 않은 정상적인 프로그램')
print('프로그램을 종료합니다.')
```

```
10
에러가 발생하지 않은 정상적인 프로그램
프로그램을 종료합니다.
```

- 정상적으로 실행 됐을 때 else 문을 부른다.

Finally:

```
try :
    print(int('안녕'))
    # data = [10, 20, 30, 40, 50]
    # print(data[0])
except ZeroDivisionError as e: # ZeroDivisionError 에러가 났을때만 실행
    print(e)
except IndexError as e: # IndexError 에러가 났을때만 실행
    print(e)
except ValueError as e:
    print(e)
except Exception as e: # Except는 맨 마지막
    print(e)
else:
    print('에러가 발생하지 않은 정상적인 프로그램')
finally:
    print('에러에 관계없이 무조건 실행되는 문장')
print('프로그램을 종료합니다.')
```

invalid literal for int() with base 10: '안녕'
에러에 관계없이 무조건 실행되는 문장
프로그램을 종료합니다.

- 에러에 관계없이 무조건 실행된다.
- 연결하다가 끊어줄때 사용한다. (무조건 해야하는 업무가 있을때)

4. Exception Class

- Exception 클래스는 파이썬의 내장 예외 계층 구조에서 거의 모든 내장 예외의 기본 클래스입니다.
- 이 클래스는 사용자 정의 예외를 만들거나 특정 예외 유형을 잡기 위한 기본적인 인터페이스를 제공합니다.
- 예외 유형은 Exception을 상속받아서 정의됩니다. 예를 들면 ValueError, TypeError, FileNotFoundError 등이 있습니다.
- 이 상속 구조 덕분에 except Exception 블록은 Exception을 상속받은 모든 예외를 처리할 수 있습니다.

↓ raise

```
try:
    raise Exception('예외가 발생했어요!!')
```

- raise exception은 메시지로 나온다.
- 직접 에러를 만들 수 있다.

5. 사용자 정의 예외 클래스 및 활용:

```
class AgeLimitError(Exception):  
    def __init__(self, age, message='MZ세대 나이의 범위가 아님'):  
        self.age = age  
        self.message = message  
        super().__init__(self.message)  
  
def check_age(age):  
    if age < 14:  
        raise AgeLimitError(age, 'MZ 나이 범위보다 작음')  
    elif age > 43:  
        raise AgeLimitError(age, 'MZ 나이 범위보다 큼')  
    else:  
        return f'{age} 나이: MZ 나이 범위안에 포함'
```

```
ages = [17, 60, 46, 20, 80, 55, 26, 12, 5]  
  
for age in ages:  
    try:  
        print(check_age(age))  
    except AgeLimitError as e:  
        print(f'{e.age} 나이 에러 {e}')
```

- e 안에는 메시지가 포함되어 있다. 또한 나이에 접근이 가능하다.