예외 처리

예외(Exception) 처리

- 문법적으로 문제가 없는 코드를 실행하는 중에 발생하는 오류

- 예외처리 구문

```
try:
    # 문제가 없을 경우 실행 할 코드

except:
    # 문제가 생겼을 때 실행 할 코드
```

Exception 클래스

- BaseException
  - SystemExit
  - Keyboardinterrupt
  - GeneratorExit
  - Exception
    - ...
    - ArithmeticError
      - ZeroDivsionError
      - ...
    - LookupError
      - IndexError
      - ...
    - ...

예외(Exception) 처리 구문

- 예외처리 구문

```
try:
    # 문제가 없을 경우 실행 할 코드

except 예외형식1:
    # 문제가 생겼을 때 실행 할 코드

except 예외형식2:
    # 문제가 생겼을 때 실행 할 코드
```

예외(Exception) 처리 구문

- 예외처리 구문

```
try:
    # 문제가 없을 경우 실행 할 코드

except 예외형식1 as e:
    # 문제가 생겼을 때 실행 할 코드

except 예외형식2 as e:
    # 문제가 생겼을 때 실행 할 코드
```

예외(Exception) 처리 구문

- 예외처리 구문
  - try절을 무사히 실행하면 만날 수 있는 else

```
try:
    # 실행할 코드 블록

except:
    # 예외 처리 코드 블록

else:
    # except절을 만나지 않았을 경우 실행하는 코드 블록
```

예외(Exception) 처리 구문

- 예외처리 구문
  - 반드시 실행되는 finally

```
try:
    # 코드 블록

except:
    # 코드 블록

else:
    # 코드 블록

finally:
    # 코드 블록
```

예외(Exception) 발생

- 예외 발생 처리1

```
try:
    raise Exception("예외를 일으킵니다.")
except Exception as e:
    print("예외가 발생하였습니다. :{0}".format(e))
```

- 예외 발생 처리2

```
try:
    # 예외 발생

except:
    raise
```

사용자 정의 예외(Exception) 발생

- 형식

```
class MyException(Exception):
    def __init__(self):
        super().__init__("MyException이 발생했습니다.")

if everything_is_fine == False:
    raise MyException()
```