■ 오류가 발생하는 여러가지 상황

- ZeroDivisionError : division by zero

  ```
  2 / 0
  ```

- IndexError : list index out of range

  ```
  list = []
  list[0]
  ```

- TypeError : unsupported operand type(s) for +: 'int' and 'str'

  ```
  4 + ''
  ```

- FileNotFoundError: [Errno 2] No such file or directory: 'python.txt'

  ```
  file = open('python.txt', 'r')
  ```

- TypeError: unsupported operand type(s) for +: 'set' and 'set'

  ```
  {1, 2} + {3}
  ```

■ 오류 예외 처리 기법

● 기본 구조

```
try:
    …
except [발생 오류[as 오류 메시지 변수]]:
    …
```

- 오류가 발생하면 해당 라인에서 동작을 멈추고 except 블록으로 이동
- 오류가 발생하지 않으면 except 블록은 실행되지 않음

● try – except

```
try:
    …
except:
    …
```

- 오류가 발생하면 종류에 상관없이 except 블록으로 이동

● try – except 발생 오류

```
try:
    …
except 발생 오류:
    …
```

- 오류가 발생하면 명시된 오류명과 같을때만 except 블록으로 이동

■ 오류 예외 처리 기법

● try – except 발생 오류 as 오류 메시지 변수

```
try:
    …
except 발생 오류 as 오류 메시지 변수:
    …
```

- 오류가 발생하면 명시된 오류명과 같을때만 except 블록으로 이동
- 오류 메시지의 내용까지 활용 가능

■ 예외 처리 적용

● try – except

- ZeroDivisionError

```
try:
    2 / 0
except:
    print('예외발생')
```

```
예외발생
```

- IndexError

```
try:
    list = []
    list[0]
except:
    print('예외발생')
```

```
예외발생
```

- FileNotFoundError

```
try:
    file = open('python.txt', 'r')
except:
    print('예외발생')
```

```
예외발생
```

■ 예외 처리 적용

● try – except 발생 오류

- ZeroDivisionError

```
try:
    2 / 0
except ZeroDivisionError:
    print('ZeroDivisionError 예외발생')
```

```
ZeroDivisionError 예외발생
```

- IndexError

```
try:
    list = []
    list[0]
except IndexError:
    print('IndexError 예외발생')
```

```
IndexError 예외발생
```

- FileNotFoundError

```
try:
    file = open('python.txt', 'r')
except FileNotFoundError:
    print('FileNotFoundError 예외발생')
```

```
FileNotFoundError 예외발생
```

■ 예외 처리 적용

● try – except 발생 오류 as 오류 메시지 변수

- ZeroDivisionError

```
try:
    2 / 0
except ZeroDivisionError as e:
    print(e, '예외발생')
```

```
division by zero 예외발생
```

- IndexError

```
try:
    list = []
    list[0]
except IndexError as e:
    print(e, '예외발생')
```

```
list index out of range 예외발생
```

- FileNotFoundError

```
try:
    file = open('python.txt', 'r')
except FileNotFoundError as e:
    print(e, '예외발생')
```

```
[Errno 2] No such file or directory: 'python.txt' 예외발생
```

■ try - except - else (오류가 없는 경우만 실행)

● 기본 구조

```
try:
    …
except:
    …
else:
    …
```

- 오류가 발생하지 않으면 else 블록 실행
- 반드시 except 절 다음에 위치

● 예외 처리 적용

```
try:
    file = open('python.txt', 'r')
except:
    print('예외발생')
else:  # 파일이 정상적으로 열렸다면
    file.read()
    print(file)
    file.close()
```

■ try – except – finally (항상 실행)

● 기본 구조

```
try:
    …
except:
    …
finally:
    …
```

- 오류 발생 유무와 관계없이 항상 실행
- 반드시 실행되어야 하는 경우 사용 (ex 파일 사용 후 닫기, DB 연결 후 종료)

● 예외 처리 적용

```
try:
    file = open('test.txt', 'r')
    print('file open')
    text = file.read()
    print(text)
    text += 1  # TypeError 발생
except:
    print('예외발생')
finally:
    file.close()  # 항상 실행
    print('file close')
```

```
file open
1234
가나다라
ACBD

예외발생

file close
```

■ try - except - except (여러개의 오류 처리)

● 기본 구조

```
try:
    …
except 발생 오류1:
    …
except 발생 오류2:
    …
```

- 여러가지 오류를 처리하기 위해 작성
- 특정 오류를 만나면 해당 except로 이동

● 예외 처리 적용

```
try:
    print(4 / 1)
    list = [1, 2]
    list[4] = 10  # IndexError 발생
except ZeroDivisionError as e:
    print(e, '예외발생')
except IndexError as e:
    print(e, '예외발생')
```

```
4.0
list assignment index out of range 예외발생
```

■ 의도적인 오류 발생

● raise

```
class Car:
    def move(self):
        raise NotImplementedError

class Taxi(Car):
    pass

taxi = Taxi()
taxi.move()  # NotImplementedError 발생
```

- 의도적으로 오류를 발생시키기 위해 사용
- 라이브러리를 설계하는 개발자가 활용하려는 개발자에게 반드시 메소드를 오버라이딩 후 사용하도록 하기 위해

```
class Car:
    def move(self):
        raise NotImplementedError

class Taxi(Car):
    def move(self):  # 오버라이딩
        print('fast move')

taxi = Taxi()
taxi.move()
```

■ 오류 만들기

● 기본 구조

```
class CustomError(Exception):
    pass
```

- 파이썬에 미리 만들어놓은 많은 오류 모듈이 있긴 하지만

  현재 개발하려는 서비스 상황에 알맞은 오류를 찾을수 없을 때

  직접 오류를 만들어서 쓰는 것도 가능

- Exception 이라는 클래스를 상속받는 것만으로 가능

● CustomError 적용

```
import random

class CustomError(Exception):
    pass

num = random.randint(1, 10)  # 1 ~ 10 사이의 난수 발생
if num > 5:
    print(num)
    raise CustomError  # 난수가 5보다 크면 CustomError 발생
else:
    print(num)
```

```
7
Traceback (most recen
  File "D:/dev/worksp
    raise CustomError
CustomError
```

■ 오류 만들기

● 오류를 발생시키면서 메시지 출력

```
class CustomError(Exception):
    def __init__(self, msg):
        self.msg = msg

    def __str__(self):
        return self.msg
```

● CustomError 메시지 출력

```
class CustomError(Exception):
    def __init__(self, msg):
        self.msg = msg

    def __str__(self):
        return self.msg

num = random.randint(1, 10)
if num > 5:
    print(num)
    raise CustomError('5보다 큰 수 발생')
else:
    print(num)
```

```
8
Traceback (most recent call last):
  File "D:/dev/workspace_python/07_10_cust
    raise CustomError('5보다 큰 수 발생')
CustomError: 5보다 큰 수 발생
```