예외처리란?

예외처리는 이러한 예외적 상황이 발생했을 때, 프로그램이 비정상적으로 종료되는 것을 방지

예외처리를 적절하게 수행하면 프로그램의 오류를 예측하고 대처할 수 있으며, 디버깅 시간을 줄일 수 있음

- try ~ except문을 이용하여 예외처리를 할 수 있음
- try문에서 예외가 발생할 가능성이 있는 코드를 작성하고, except문에서 해당 예외를 처리
- 만약 예외가 발생하기 않으면 except문은 실행되지 않음

아래 코드를 예외처리를 하는 가장 기본적인 구조

try: # 예외가 발생할 가능성이 있는 코드 except 예외종류 as 예외변수: # 해당 예외를 처리하는 코드

In [7]:

```
def ten_div(x):
    return 10 / x
```

In [8]:

```
ten_div(2)
```

Out[8]:

5.0

In [9]:

```
ten_div(0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~₩AppData₩Local₩Temp₩ipykernel_14620₩1084687051.py in <module>
----> 1 ten_div(0)

~₩AppData₩Local₩Temp₩ipykernel_14620₩76904369.py in ten_div(x)
      1 def ten_div(x):
----> 2     return 10 / x

ZeroDivisionError: division by zero
```

ZeroDivisionError뿐만 아니라 AttributeError, NameError, TypeError 등 다양한 에러들도 모두 예외

- 예외가 발생했을 때도 스크립트 실행을 중단하지 않고 계속 실행하게 해주는 예외 처리 방법

try except로 사용하기

예외 처리를 하려면 다음과 같이 try에 실행할 코드를 넣고 except에 예외가 발생했을 때 처리하는 코드를 넣습니다.

```
try:
    실행할 코드
except:``
    예외가 발생했을 때 처리하는 코드
```

- 숫자를 0으로 나누었을 때 발생하는 예외를 처리

In [3]:

```
try:
    x = int(input('나눌 숫자를 입력하세요: '))
    y = 10 / x
    print(y)
except:    # 예외가 발생했을 때 실행됨
    print('예외가 발생했습니다.')
```

```
나눌 숫자를 입력하세요: 0
예외가 발생했습니다.
```

- 숫자를 0으로 나누면 ZeroDivisionError 예외가 발생합니다.
- 여기서는 except에서 예외 처리를 하도록 만들었으므로 '예외가 발생했습니다.'가 출력

- 예외가 발생하면 해당 줄에서 코드 실행을 중단하고 바로 except로 가서 코드를 실행
- 즉, try의 y = 10 / x를 비롯하여 그 아래줄에 있는 print(y)도 실행되지 않음

- 즉, try의 코드에서 에러가 발생했을 때만 except의 코드가 실행

In [ ]:

```

```

특정 예외만 처리하기

except에 예외 이름을 지정해서 특정 예외가 발생했을 때만 처리 코드를 실행

In [ ]:

```
try:
    실행할 코드
except 예외이름:
    예외가 발생했을 때 처리하는 코드
```

- 정수 두 개를 입력받아서 하나는 리스트의 인덱스로 사용
- 하나는 나누는 값으로 사용

- except를 두 개 사용하고 각각 ZeroDivisionError와 IndexError를 지정

In [12]:

```
y = [10, 20, 30]

try:
    index, x = map(int, input('인덱스와 나눌 숫자를 입력하세요: ').split())
    print(y[index] / x)
except ZeroDivisionError:    # 숫자를 0으로 나눠서 에러가 발생했을 때 실행됨
    print('숫자를 0으로 나눌 수 없습니다.')
except IndexError:           # 범위를 벗어난 인덱스에 접근하여 에러가 발생했을 때 실행됨
    print('잘못된 인덱스입니다.')
```

```
인덱스와 나눌 숫자를 입력하세요: 2 0
숫자를 0으로 나눌 수 없습니다.
```

- 2 0을 입력하면 10 / 0이 되므로 숫자를 0으로 나누게 됨.
- 이때는 except ZeroDivisionError:의 처리 코드가 실행

In [13]:

```
y = [10, 20, 30]

try:
    index, x = map(int, input('인덱스와 나눌 숫자를 입력하세요: ').split())
    print(y[index] / x)
except ZeroDivisionError:    # 숫자를 0으로 나눠서 에러가 발생했을 때 실행됨
    print('숫자를 0으로 나눌 수 없습니다.')
except IndexError:           # 범위를 벗어난 인덱스에 접근하여 에러가 발생했을 때 실행됨
    print('잘못된 인덱스입니다.')
```

```
인덱스와 나눌 숫자를 입력하세요: 3 5
잘못된 인덱스입니다.
```

- y = [10, 20, 30]은 요소가 3개 들어있는 리스트
- 인덱스에 3을 지정하면 범위를 벗어나게 됨
- 이때는 except IndexError:의 처리 코드가 실행

예외의 에러 메시지 받아오기

특히 except에서 as 뒤에 변수를 지정하면 발생한 예외의 에러 메시지를 받아올 수 있음

```
try:
    실행할 코드
except 예외 as 변수:
    예외가 발생했을 때 처리하는 코드
```

- 코드의 except에 as e를 넣습니다. 보통 예외(exception)의 e를 따서 변수 이름을 e

In [18]:

```
y = [10, 20, 30]

try:
    index, x = map(int, input('인덱스와 나눌 숫자를 입력하세요: ').split())
    print(y[index] / x)
except ZeroDivisionError as e:                    # as 뒤에 변수를 지정하면 에러를 받아옴
    print('숫자를 0으로 나눌 수 없습니다.', e)    # e에 저장된 에러 메시지 출력
except IndexError as e:
    print('잘못된 인덱스입니다.', e)
```

```
인덱스와 나눌 숫자를 입력하세요: 2 0
숫자를 0으로 나눌 수 없습니다. division by zero
```

In [19]:

```
y = [10, 20, 30]

try:
    index, x = map(int, input('인덱스와 나눌 숫자를 입력하세요: ').split())
    print(y[index] / x)
except ZeroDivisionError as e:                    # as 뒤에 변수를 지정하면 에러를 받아옴
    print('숫자를 0으로 나눌 수 없습니다.', e)    # e에 저장된 에러 메시지 출력
except IndexError as e:
    print('잘못된 인덱스입니다.', e)
```

```
인덱스와 나눌 숫자를 입력하세요: 3 5
잘못된 인덱스입니다. list index out of range
```

- 참고로 모든 예외의 에러 메시지를 출력하고 싶다면
- 다음과 같이 except에 Exception을 지정하고 as 뒤에 변수를 넣으면 됩니다.

In [ ]:

```
except Exception as e:    # 모든 예외의 에러 메시지를 출력할 때는 Exception을 사용
    print('예외가 발생했습니다.',
```

- 가장 기본적으로 사용되는 절은 try 절과 except 절
- try 안에는 기본적으로 실행하는 코드를 넣고 except 절에는 에러가 발생했을 경우 시행할 코드를 작성

- 파일에 대한 예외사항

In [ ]:

```
try:
    # 예외가 발생할 가능성이 있는 코드
    file = open("example.txt", "r")
    content = file.read()
    file.close()
except FileNotFoundError:
    # 해당 예외를 처리하는 코드
    print("파일을 찾을 수 없습니다.")
except PermissionError:
    # 해당 예외를 처리하는 코드
    print("파일에 대한 권한이 없습니다.")
except:
    # 해당 예외를 처리하는 코드
    print("예상치 못한 예외가 발생했습니다.")
```

예외 처리 try ~ except

In [20]:

```
try:
    print(5/0)

except:
    print('wrong division')
```

```
wrong division
```

- 결과는 wrong division 이라는 문자열이 출력
- try, except 만 사용한다면 어떠한 종류의 에러가 발생하더라도 except 절의 코드가 실행

In [21]:

```
try:
    print(5/0)

except ZeroDivisionError:
    print('wrong division')
```

```
wrong division
```

- ZeroDivisionError 이라는 오류를 명시
- 이 경우 다른 에러가 발생하면 except 절에서 처리하지 않음.

In [22]:

```
try:
    print(5/0)

except ZeroDivisionError as e:
    print(e)
```

```
division by zero
```

- 이 코드는 'division by zero' 라는 메시지를 출력
- ZeroDivisionError 에러에 저장되어있는 string 을 출력

예외 처리 try~except ~ else

- 예외처리의 기본인 try 절과 except 절에 else 절을 추가해서 에외처리문을 구성
- else 절은 예외가 발생하지 않아 except 절을 실행하지 않았을 경우 실행되는 절

In [23]:

```
try:
    print(5/1)

except:
    print('error')

else:
    print('no error')
```

```
5.0
no error
```

- 이 코드는 5.0 과 'no error' 를 출력
- try 절에서 문제가 없었기 때문에 except 절이 실행되지 않고 else 절이 실행되는 것을 볼 수 있음

예외 처리 try~except ~ finally

- finally 절은 try 절에서 예외의 발생여부에 관계없이 항상 실행되는 절

In [24]:

```
try:
    print(5/0)

except:
    print('error')

finally:
    print('end')
```

```
error
end
```

- finally 절은 try 절에서 예외의 발생여부에 관계없이 항상 실행되는 절

예외 처리- 여러개의 에러 처리하기

- 어떤 예외가 나와도 동일하게 처리할 수도 있지만, 다른 예외가 발생하면 다르게 처리하는 것이 좋은 예외 처리 방법.

In [25]:

```
try:
    print(5/0)
    print(e)

except NameError as e:
    print(e)

except ZeroDivisionError as e:
    print(e)
```

```
division by zero
```

- 코드에서 예외는 보통 한 가지만 나타나지 않음
- 변수가 설정되지 않았을 경우 발생하는 에러인 NameError 과 분모가 0 일 경우 발생하는 ZeroDivisionError 에러를 예외처리

In [26]:

```
try:
    print(5/0)
    print(e)

except NameError as e:
    print(e)

except ZeroDivisionError as e:
    pass
```

예외 처리- 오류 발생시키기

- 프로그램에서 입력을 받는 경우에 목적에 맞게 입력을 받아야 할 경우
- 예를 들어 나이를 입력받는다면 0보다 작으면 안되는 것 처럼 문법상에는 오류가 없지만 오류가 필요한 경우 오류를 일부러 발생시킬 수 있음

In [28]:

```
try:
    age=int(input())
    if age < 0:
        raise NotImplementedError
    print(age)
except NotImplementedError:
    print('NotImplementedError')
```

```
-1
NotImplementedError
```

In [ ]: