Errors and Exceptions

- 발생할 수 있는 오류와 예외처리를 확인해봅시다.

문법 에러(Syntax Error)

- 가장 많이 만날 수 있는 에러로 발생한 `파일 이름`과 `줄`, `^`을 통해 파이썬이 읽어 들일 때(parser)의 문제 발생 위치를 표현한다.

In [1]:

```
# if문을 통해 발생시켜봅시다!
if True:
    print('참')
else
    print('거짓')
```

```
  File "<ipython-input-1-ad301a5287d3>", line 4
    else
       ^
SyntaxError: invalid syntax
```

In [2]:

```
# print문을 통해 다른 오류를 발생시켜봅시다!
# EOL 오류를 봅시다.    # EOL = End of Line
print('hi)
```

```
  File "<ipython-input-2-e49455d49585>", line 3
    print('hi)
                ^
SyntaxError: EOL while scanning string literal
```

In [3]:

```
# EOF 에러도 보게 됩니다.
print('hi'
```

```
  File "<ipython-input-3-bbe254e9d0ff>", line 2
    print('hi'
                ^
SyntaxError: unexpected EOF while parsing
```

- 정확한 위치를 지정하지 않을 수도 있으므로 앞뒤로 모두 확인을 해봐야합니다.

In [4]:

```
if True print('참')
```

```
  File "<ipython-input-4-ae56d80f3d7b>", line 1
    if True print('참')
                 ^
SyntaxError: invalid syntax
```

예외 (Exceptions)

- 문법이나 표현식이 바르게 되어있지만, 실행시 발생하는 에러입니다.
- 아래 제시된 모든 에러는 Exception을 상속받아 이뤄집니다.

In [5]:

```
# ZeroDivisionError를 확인해봅시다.
10 * (1/0)

# 0으로 나눌 수는 없죠!
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-5-dd6f62bf6990> in <module>
      1 # ZeroDivisionError를 확인해봅시다.
----> 2 10 * (1/0)
      3
      4 # 0으로 나눌 수는 없죠!

ZeroDivisionError: division by zero
```

In [6]:

```
# NameError를 확인해봅시다.
print(ssafy)

# 지역 혹은 전역 이름 공간내에서 유효하지 이름, 즉 정의되지 않은 변수를 호출 하였을 경우
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-58b5fea814cf> in <module>
      1 # NameError를 확인해봅시다.
----> 2 print(ssafy)
      3
      4 # 지역 혹은 전역 이름 공간내에서 유효하지 이름, 즉 정의되지 않은 변수를 호출 하였을 경우

NameError: name 'ssafy' is not defined
```

In [7]:

```
# TypeError를 확인해봅시다.
1 + '1'

# 자료형에 대한 타입 자체가 잘못 되었을 경우
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-e75d1782c21d> in <module>
      1 # TypeError를 확인해봅시다.
----> 2 1 + '1'
      3
      4 # 자료형에 대한 타입 자체가 잘못 되었을 경우

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

In [8]:

```
# 함수 호출과정에서 TypeError도 발생하게 됩니다. 확인해봅시다.
round('3.5')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-f45051a456cd> in <module>
      1 # 함수 호출과정에서 TypeError도 발생하게 됩니다. 확인해봅시다.
----> 2 round('3.5')

TypeError: type str doesn't define __round__ method
```

In [10]:

```
# 함수호출 과정에서 다양한 오류를 확인할 수 있습니다. : 필수 argument 누락
import random
random.sample([1,2,3])
```

```
---------------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)
<ipython-input-10-4a9ab8806444> in <module>
      1 # 함수호출 과정에서 다양한 오류를 확인할 수 있습니다. : 필수 argument 누락
      2 import random
----> 3 random.sample([1,2,3])

TypeError: sample() missing 1 required positional argument: 'k'
```

In [11]:

```
# 함수호출 과정에서 다양한 오류를 확인할 수 있습니다. : argument 많은 경우
random.choice([1,2,3], 5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-1072eec09af4> in <module>
      1 # 함수호출 과정에서 다양한 오류를 확인할 수 있습니다. : argument 많은 경우
----> 2 random.choice([1,2,3], 5)

TypeError: choice() takes 2 positional arguments but 3 were given
```

In [12]:

```
# ValueError를 확인해봅시다.
int('3.5')

# 자료형에 대한 타입은 올바르나 값이 적절하지 않는 경우
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-c7d2327c0d29> in <module>
      1 # ValueError를 확인해봅시다.
----> 2 int('3.5')
      3
      4 # 자료형에 대한 타입은 올바르나 값이 적절하지 않는 경우

ValueError: invalid literal for int() with base 10: '3.5'
```

In [13]:

```
# ValueError를 확인해봅시다.
a = [1, 2]
a.index(3)

# 값이 적절하지 않은 경우(값이 없는데 찾으려고함)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-4f49465e7148> in <module>
      1 # ValueError를 확인해봅시다.
      2 a = [1, 2]
----> 3 a.index(3)
      4
      5 # 값이 적절하지 않은 경우(값이 없는데 찾으려고함)

ValueError: 3 is not in list
```

In [18]:

```
# IndexError를 확인해봅시다.
a = []
a[-1]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-18-5bf88c1938f7> in <module>
      1 # IndexError를 확인해봅시다.
      2 a = []
----> 3 a[-1]

IndexError: list index out of range
```

IndexError: list index out of range

In [19]:

```
# KeyError를 확인해봅시다.
a = {'sia': 'candy cane lane'}
a['beyonce']
# 딕셔너리에서 Key가 없는 경우 발생합니다.
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-19-d4f6c2bd63d3> in <module>
      1 # KeyError를 확인해봅시다.
      2 a = {'sia': 'candy cane lane'}
----> 3 a['beyonce']
      4 # 딕셔너리에서 Key가 없는 경우 발생합니다.

KeyError: 'beyonce'
```

In [21]:

```
# ModuleNotFoundError를 확인해봅시다.
import ssafy

# 모듈을 찾을 수 없는 경우
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-21-f6058e9b8ff5> in <module>
      1 # ModuleNotFoundError를 확인해봅시다.
----> 2 import ssafy
      3
      4 # 모듈을 찾을 수 없는 경우

ModuleNotFoundError: No module named 'ssafy'
```

In [23]:

```
# ImportError를 확인해봅시다.
from random import ssafy

# 모듈을 찾았으나 가져오는 과정에서 실패하는 경우(대부분 없는 클래스/메소드를 불러옴)
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-23-7c1c2ab9bab7> in <module>
      1 # ImportError를 확인해봅시다.
----> 2 from random import ssafy
      3
      4 # 모듈을 찾았으나 가져오는 과정에서 실패하는 경우(대부분 없는 클래스/메소드를 불러옴)

ImportError: cannot import name 'ssafy'
```

In [24]:

```
# KeyboardInterrupt를 확인해봅시다.
while True:
    continue

# 주피터 노트북에서는 정지 버튼이지만, 실제로 우리가 돌릴 때는 ctrl+c를 통해 종료하였을 때 발생
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-24-85303763ef44> in <module>
      1 # KeyboardInterrupt를 확인해봅시다.
      2 while True:
----> 3     continue
      4
      5 # 주피터 노트북에서는 정지 버튼이지만, 실제로 우리가 돌릴 때는 ctrl+c를 통해 종료하였을 때 발생

KeyboardInterrupt:
```

예외 처리

기본 - `try` `except`

`try` 구문을 이용하여 예외 처리를 할 수 있습니다.

기본은 다음과 같은 구조를 가지고 있습니다.

```
try:
    codeblock1
except 예외:
    codeblock2
```

- `try` 절이 실행됩니다.
- 예외가 발생되지 않으면, `except` 없이 실행이 종료가 됩니다.
- 예외가 중간에 발생하면, 남은 부분을 수행하지 않고, `except` 가 실행됩니다.

In [25]:

```
# 사용자로부터 값을 받아 정수로 변환하여 출력해봅시다.
num = input('값을 입력하시오 : ')
print(int(num))
```

```
값을 입력하시오 : 10
10
```

In [26]:

```
# 사용자가 문자열을 넣어 해당 오류(ValueError)가 발생하면, 숫자를 입력하라고 출력해봅시다.
try:
    num = input('값을 입력하시오 : ')
    print(int(num))
except ValueError:
    print('바보야 숫자를 입력해!')
```

```
값을 입력하시오 : ㅎㅎ
바보야 숫자를 입력해!
```

In [27]:

```
# Error 발생 시점 이후의 코드는 실행되지 않고, except로 바로 넘어갑니다.
try:
    num = input('값을 입력하시오 : ')
    print(int(num))
    print(num)
except ValueError:
    print('바보야 숫자를 입력해!')
```

```
값을 입력하시오 : ㅎㅎ
바보야 숫자를 입력해!
```

복수의 예외 처리

- 두 가지 예외를 모두 처리할 수 있습니다.

```
try:
    codeblock1
except (예외1, 예외2):
    codeblock2
```

In [28]:

```
# 100을 사용자가 입력한 값으로 나눈 후 출력하는 코드를 작성해봅시다.
num = input('값을 입력하시오 : ')
print(100/int(num))
```

```
값을 입력하시오 : 10
10.0
```

In [31]:

```
# 문자열일때와 0일때 모두 처리를 해봅시다.
try:
    num = input('값을 입력하시오 : ')
    print(100/int(num))
except (ValueError, ZeroDivisionError):
    print('바보')
```

```
값을 입력하시오 : 0
바보
```

In [55]:

```
# 각각 다른 오류를 출력할 수 있습니다.
try:
    num = input('값을 입력하시오 : ')
    print(100/int(num))
except ValueError:
    print('바보야 숫자를 입력해!')
except ZeroDivisionError:
    print('바보야 0은 안돼!')
except:
    print('모르겠는데 에러야')
```

```
값을 입력하시오 : ㅁ
바보야 숫자를 입력해!
```

- 여기서 중요한 내용은 에러가 순차적으로 수행됨으로, 가장 작은 범주부터 시작해야합니다.

In [35]:

```
try:
    num = input('값을 입력하시오 : ')
    print(100/int(num))
except Exception:
    print('모르겠지만 에러야')
except ZeroDivisionError:
    print('0으로 나누면 안돼')
```

```
값을 입력하시오 : 0
모르겠지만 에러야
```

에러 문구 처리

- 에러 문구를 함께 넘겨줄 수 있습니다.

```
try:
    codeblock1
except 예외 as e:
    codeblock2
```

In [36]:

```
# 에러 메세지를 넘겨줄 수도 있습니다.
try:
    a = []
    print(a[-1])
except IndexError as e:
```

```
except IndexError as e:
    print(f'{e}, 오류가 발생했습니다.')
```

```
list index out of range, 오류가 발생했습니다.
```

else

- 에러가 발생하지 않는 경우 수행되는 문장은 `else`를 이용합니다.

```
try:
    codeblock1
except 예외:
    codeblock2
else:
    codeblock3
```

In [37]:

```
try:
    a = [1, 2, 3]
    b = a[1]
except IndexError:
    print('인덱스 오류야!!!')
else:
    print(b*100)
```

```
200
```

finally

- 반드시 수행해야하는 문장은 `finally`를 활용합니다.

```
try:
    codeblock1
except 예외:
    codeblock2
finally:
    codeblock3
```

In [38]:

```
try:
    a = {'python' : 'nojam'}
    a['java']
except KeyError as e:
    print(f'{e}는 딕셔너리에 없는 키입니다.')
finally:
    print(a)
```

```
'java'는 딕셔너리에 없는 키입니다.
{'python': 'nojam'}
```

예외 발생시키기

`raise`를 통해 예외를 발생시킬 수 있습니다.

In [39]:

```
raise ValueError
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

```
<ipython-input-39-e4c8e09828d5> in <module>
----> 1 raise ValueError

ValueError:
```

메세지를 함께 출력할 수 있습니다.

In [40]:

```
raise ValueError('hi')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-40-7365820dd368> in <module>
----> 1 raise ValueError('hi')

ValueError: hi
```

실습 문제

> **양의 정수** 두개를 받아 첫번째 수를 두번째 수로 나눈 결과를 출력하는 함수를 만들어보세요.

`def my_div(num1,num2)`

- num2가 0이여서 발생하는 오류인 경우 **에러메시지**를 출력해주세요.

  예)

  `division by zero` 오류가 발생하였습니다.

- 인자가 string이여서 발생하는 경우는 **ValueError와 함께 '나눗셈은 숫자만 가능합니다.'를 출력**해주세요. (실제로 이 경우에 발생하는 것은 TypeError입니다.)
- 정상적인 경우에는 결과를 return합니다.

In [64]:

```
# 아래에 코드를 작성해주세요.
def my_div(num1, num2):
    try:
        result = num1 / num2
    except ZeroDivisionError as e:
        print(f'{e} 오류가 발생하였습니다.')
    except:
        raise ValueError('나눗셈은 숫자만 가능합니다.')
    else:
        return result
```

In [66]:

```
# print(my_div(1, 5))
# my_div(1, 0)
my_div('1', '5')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-64-853bd562461e> in my_div(num1, num2)
      3     try:
----> 4         result = num1 / num2
      5     except ZeroDivisionError as e:

TypeError: unsupported operand type(s) for /: 'str' and 'str'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-66-1c924d837fff> in <module>
      1 # print(my_div(1, 5))
```

```
      2 # my_div(1, 0)
----> 3 my_div('1', '5')

<ipython-input-64-853bd562461e> in my_div(num1, num2)
      6         print(f'{e} 오류가 발생하였습니다.')
      7     except:
----> 8         raise ValueError('나눗셈은 숫자만 가능합니다.')
      9     else:
     10         return result

ValueError: 나눗셈은 숫자만 가능합니다.
```

`assert`

`assert` 문은 예외를 발생시키는 다른 방법이다.

보통 상태를 검증하는데 사용되며 무조건 `AssertionError`가 발생한다.

```
assert Boolean expression, error message
assert a > 1, 'a 는 1보다 커야해요!'
```

위의 검증식이 거짓일 경우를 발생한다.

`raise`는 항상 예외를 발생시키고, 지정한 예외가 발생한다는 점에서 다르다.

실습 문제

> **양의 정수** 두개를 받아 첫번째 수를 두번째 수로 나눈 결과를 출력하는 함수를 만들어보세요.

`def my_div(num1,num2)`

- assert를 활용하여, int가 아닌 경우 AssertionError를 발생시켜봅시다.

In [68]:

```
# 아래에 코드를 작성해주세요.
def my_div(num1, num2):
    assert type(num1) == int and type(num2) == int, '입력된 값이 정수가 아님'
    try:
        result = num1 / num2
    except ZeroDivisionErroras as z:
        print(f'{z} 오류가 발생하였습니다.')
    else:
        return result
```

In [70]:

```
my_div(1.3, 1)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-70-d1ad026a49dc> in <module>
----> 1 my_div(1.3, 1)

<ipython-input-68-d7be77044ca8> in my_div(num1, num2)
      1 # 아래에 코드를 작성해주세요.
      2 def my_div(num1, num2):
----> 3     assert type(num1) == int and type(num2) == int, '입력된 값이 정수가 아님'
      4     try:
      5         result = num1 / num2

AssertionError: 입력된 값이 정수가 아님
```

In [ ]:

```

```