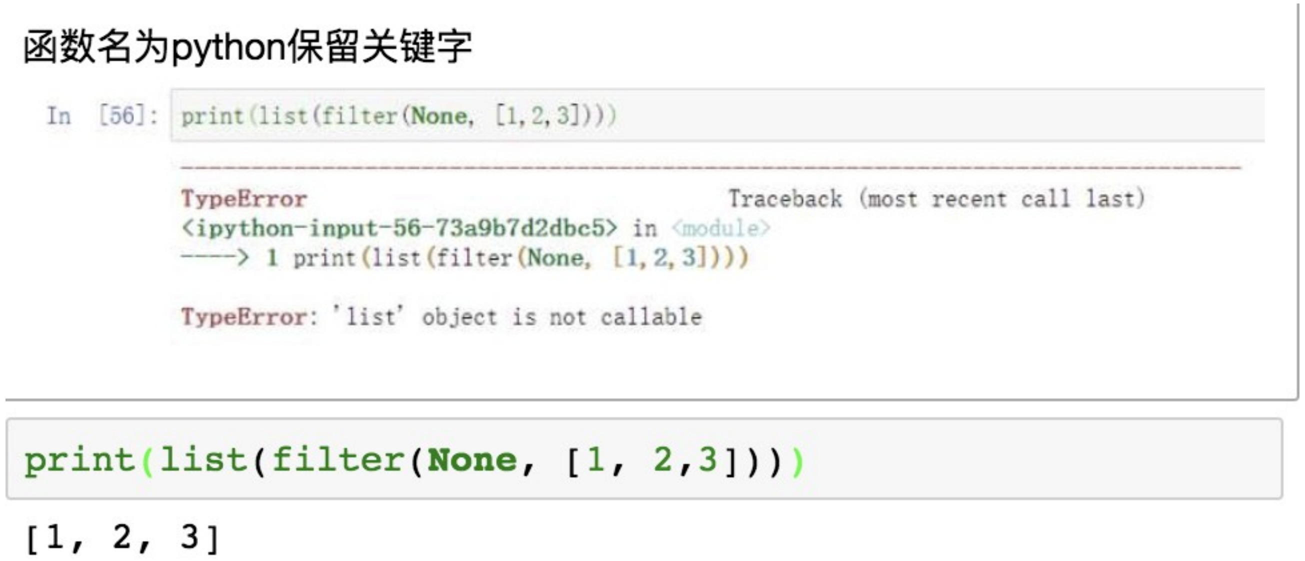
Python Exceptions

Yinuo Guo

2.24

微信群问题

- 鼓励大家有问题在群里提出，助教和老师会帮大家解答，同时大家也能共同学习



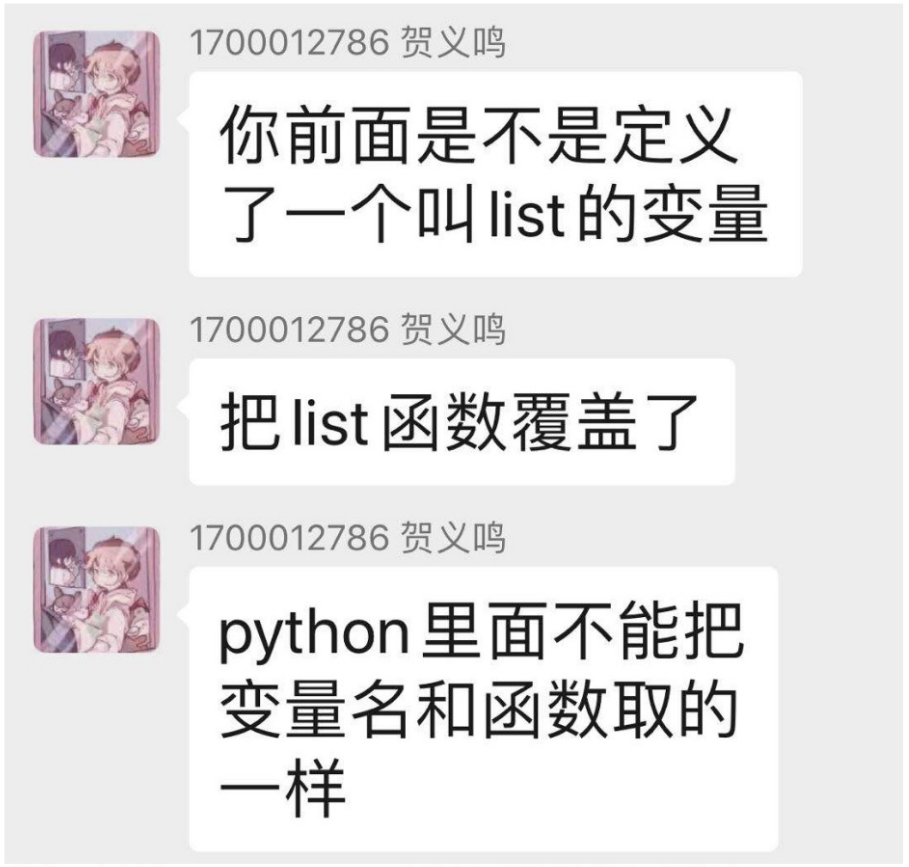
函数名为python保留关键字

```
In  [56]: print(list(filter(None, [1,2,3])))

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-56-73a9b7d2dbc5> in <module>
----> 1 print(list(filter(None, [1,2,3])))

TypeError: 'list' object is not callable
```

```
print(list(filter(None, [1, 2,3])))
```

```
[1, 2, 3]
```

1700012786 贺义鸣同学给出解答



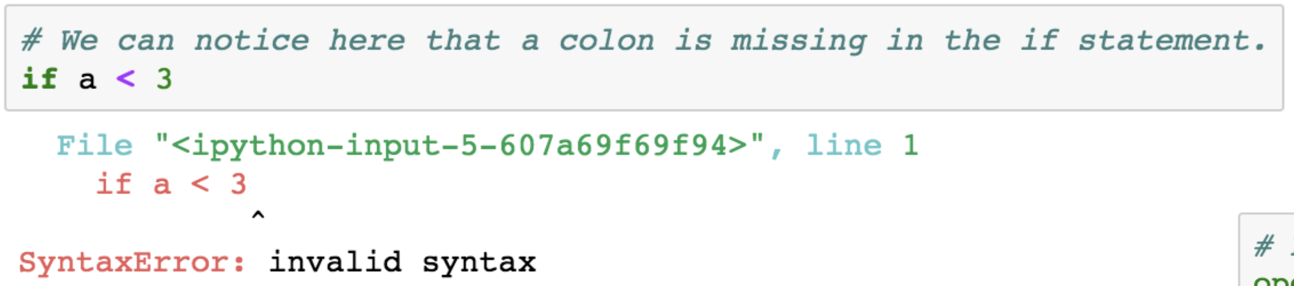
1700012786 贺义鸣
你前面是不是定义了一个叫 list 的变量

1700012786 贺义鸣
把 list 函数覆盖了

1700012786 贺义鸣
python 里面不能把变量名和函数取的一样

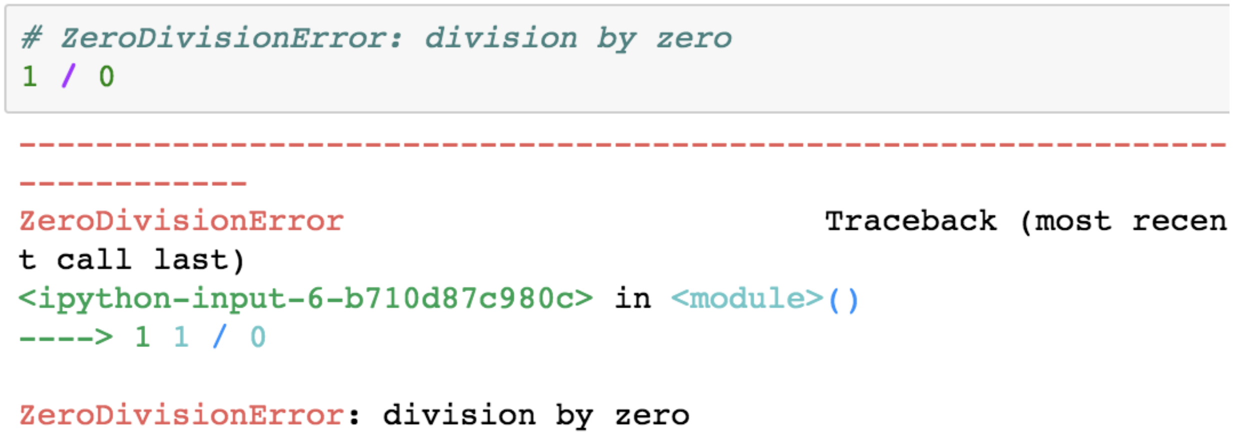
!注意：如果在jupyter notebook中发现了错误并改正，记得去restart kernel, 否则可能会得到一样的结果

Examples



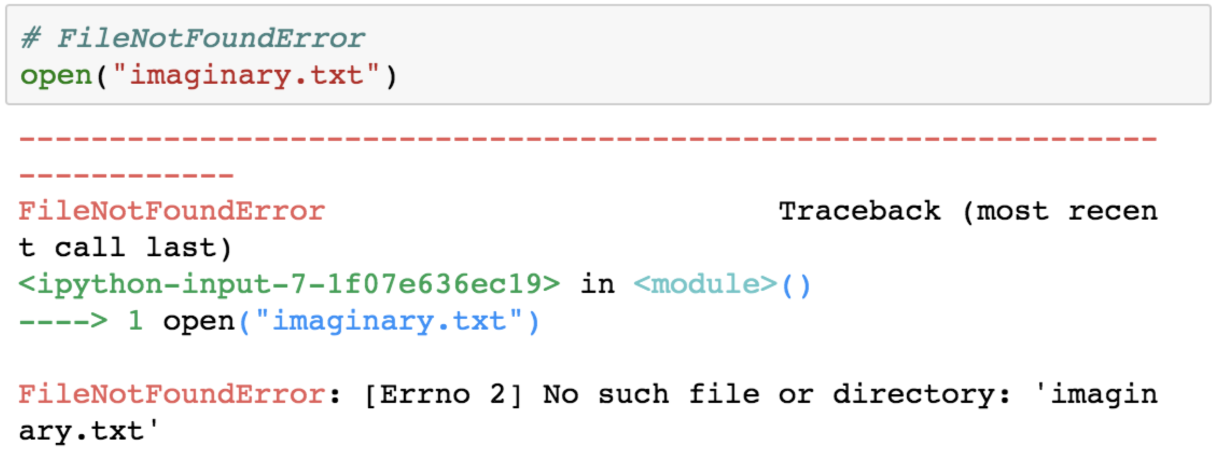
```
# We can notice here that a colon is missing in the if statement.
if a < 3
```

```
  File "<ipython-input-5-607a69f69f94>", line 1
    if a < 3
            ^
SyntaxError: invalid syntax
```



```
# ZeroDivisionError: division by zero
1 / 0
```

```
---------------------------------------------------------------
------------
ZeroDivisionError                         Traceback (most recen
t call last)
<ipython-input-6-b710d87c980c> in <module>()
----> 1 1 / 0

ZeroDivisionError: division by zero
```

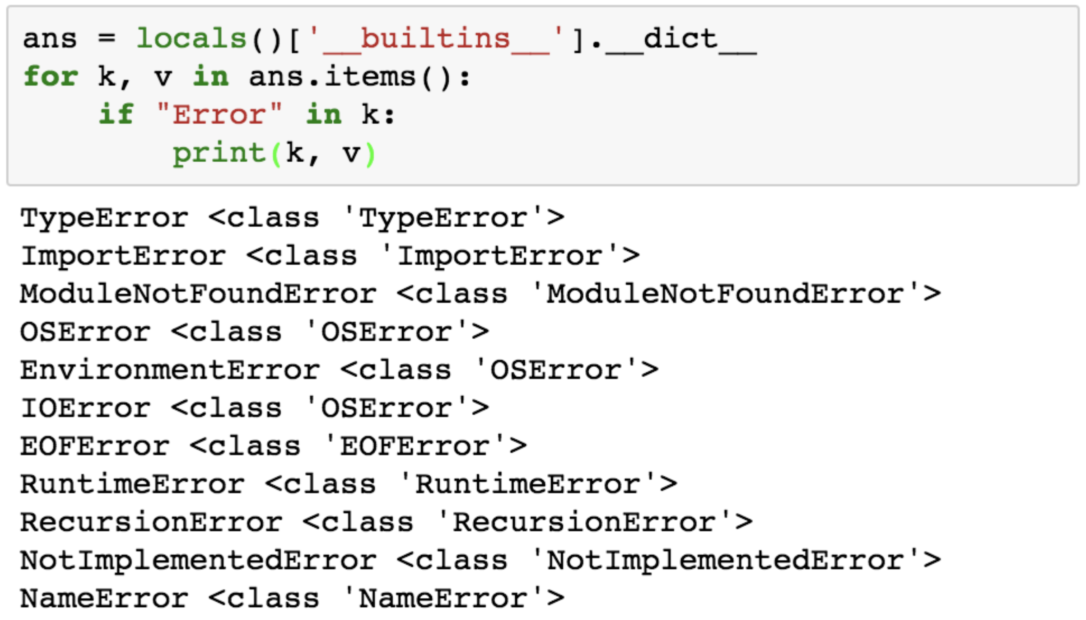


```
# FileNotFoundError
open("imaginary.txt")
```

```
---------------------------------------------------------------
------------
FileNotFoundError                         Traceback (most recen
t call last)
<ipython-input-7-1f07e636ec19> in <module>()
----> 1 open("imaginary.txt")

FileNotFoundError: [Errno 2] No such file or directory: 'imagin
ary.txt'
```

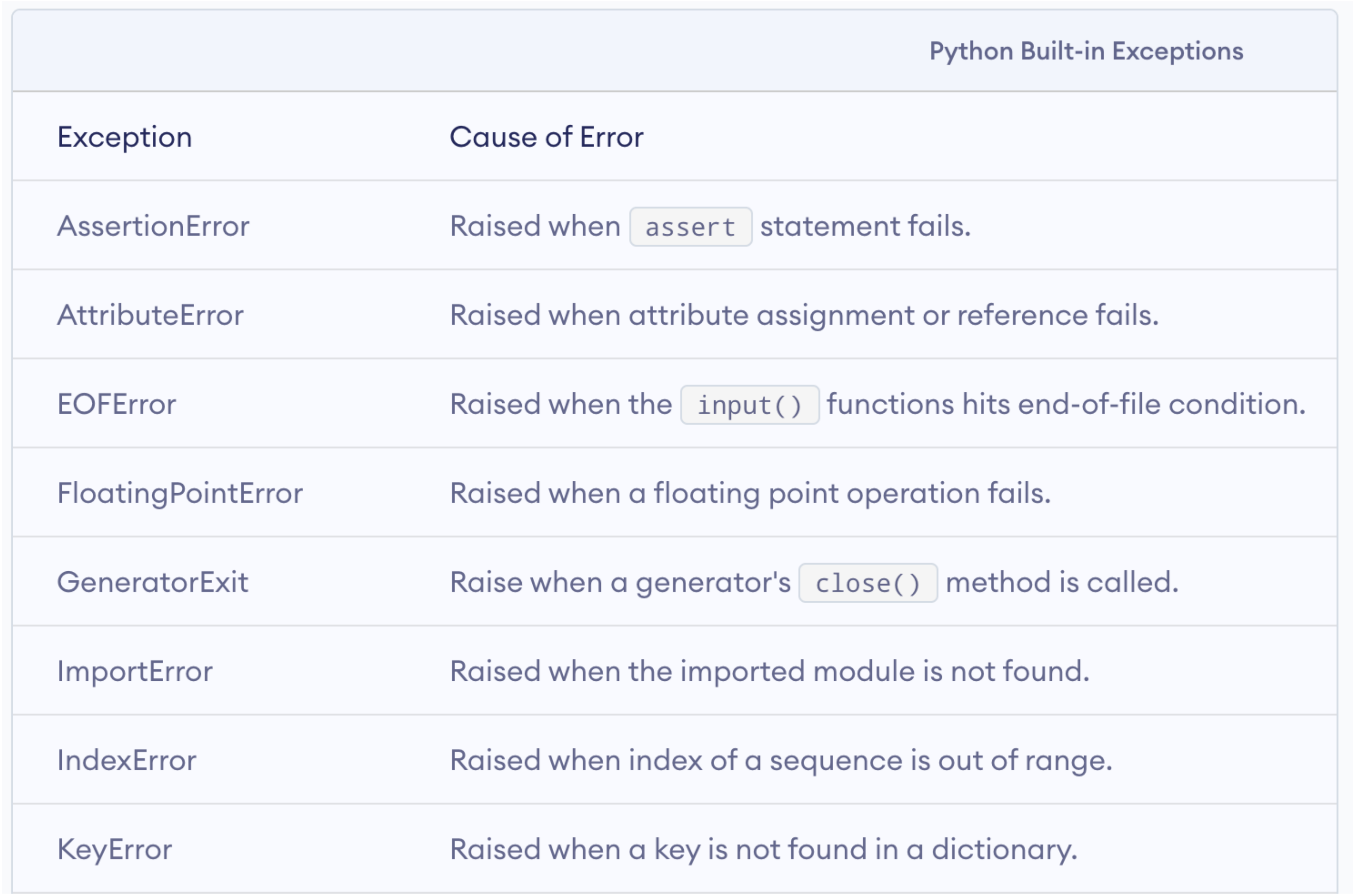
Python Built-in Exceptions

- Illegal operations can raise exceptions. There are plenty of built-in exceptions in Python that are raised when corresponding errors occur. We can view all the built-in exceptions using the local() built-in functions as follows.
- 非法操作会引发异常。当发生相应的错误时， Python 中会引发许多内置异常。我们可以使用 local （）内置函数来查看所有内置异常，如下所示。



```
ans = locals()['__builtins__'].__dict__
for k, v in ans.items():
    if "Error" in k:
        print(k, v)
```

```
TypeError <class 'TypeError'>
ImportError <class 'ImportError'>
ModuleNotFoundError <class 'ModuleNotFoundError'>
OSError <class 'OSError'>
EnvironmentError <class 'OSError'>
IOError <class 'OSError'>
EOFError <class 'EOFError'>
RuntimeError <class 'RuntimeError'>
RecursionError <class 'RecursionError'>
NotImplementedError <class 'NotImplementedError'>
NameError <class 'NameError'>
```

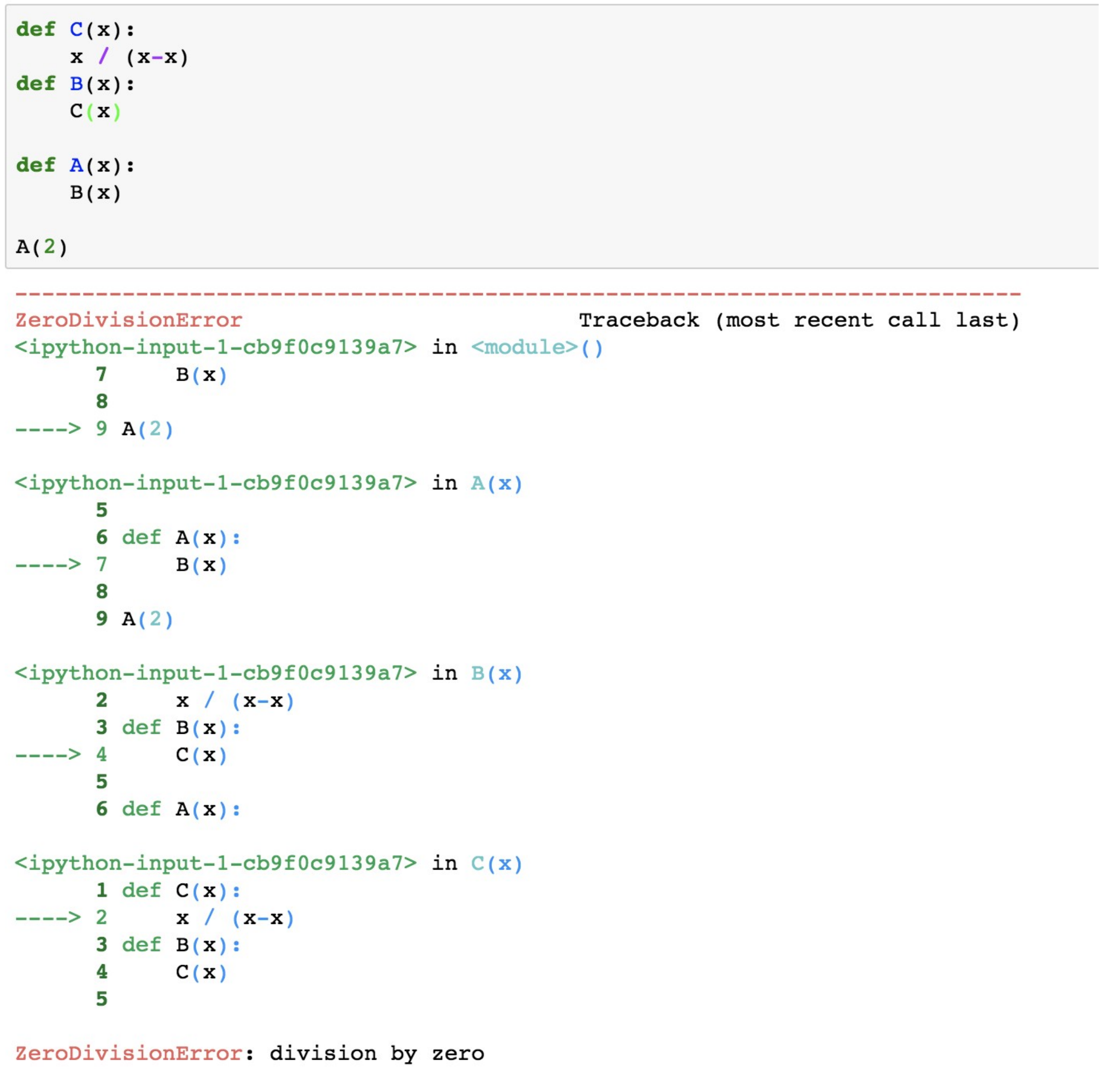


Python Built-in Exceptions

Exception	Cause of Error
AssertionError	Raised when `assert` statement fails.
AttributeError	Raised when attribute assignment or reference fails.
EOFError	Raised when the `input()` functions hits end-of-file condition.
FloatingPointError	Raised when a floating point operation fails.
GeneratorExit	Raise when a generator's `close()` method is called.
ImportError	Raised when the imported module is not found.
IndexError	Raised when index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.

Python Exception Handling

- When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled.
- 当有异常出现时，它会使当前的进程停止，并且将异常传递给调用进程，直到异常被处理为止。
- For example, if function A calls function B which in turn calls function C and an exception occurs in function C. If it is not handled in C, the exception passes to B and then to A.



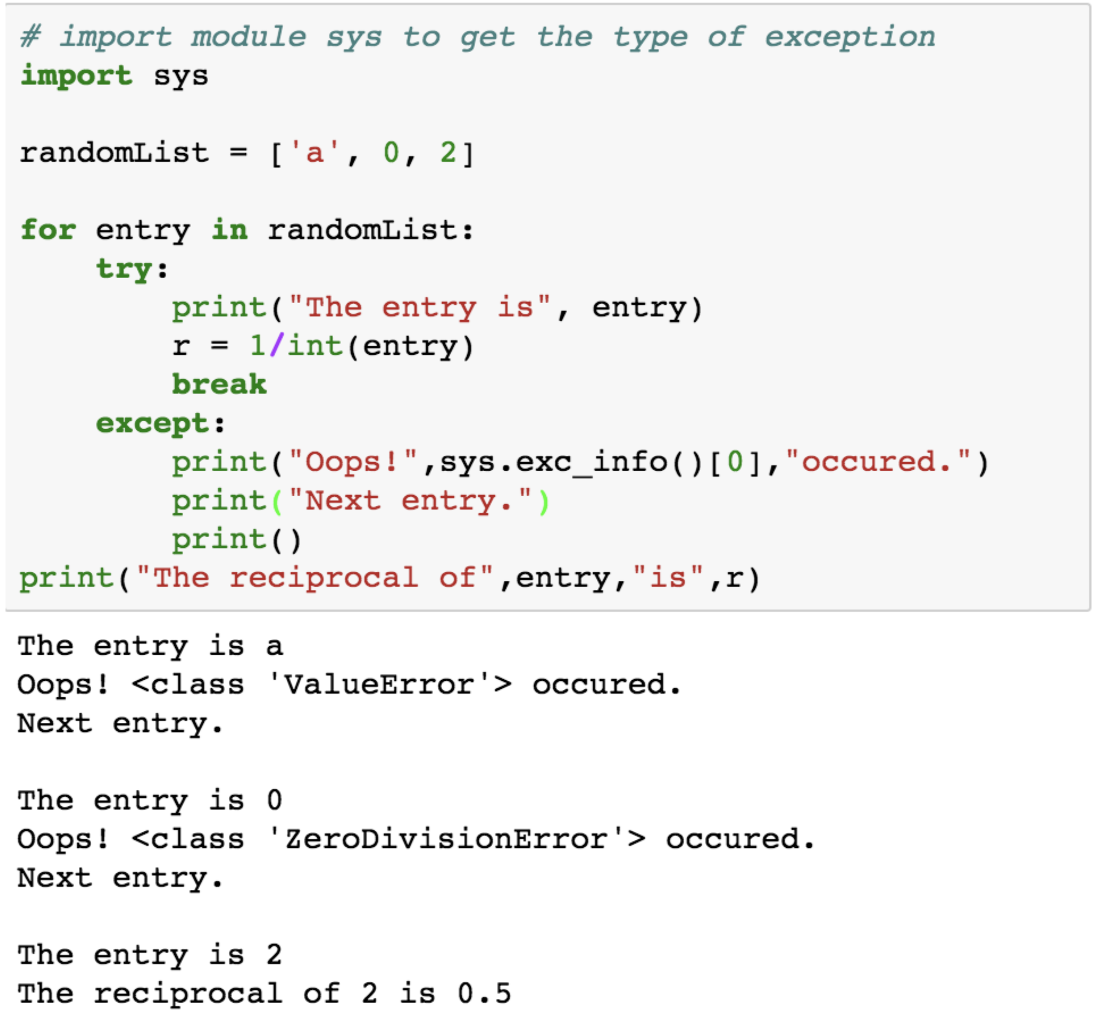
```python
def C(x):
    x / (x-x)
def B(x):
    C(x)

def A(x):
    B(x)

A(2)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-cb9f0c9139a7> in <module>()
      7     B(x)
      8
----> 9 A(2)

<ipython-input-1-cb9f0c9139a7> in A(x)
      5
      6 def A(x):
----> 7     B(x)
      8
      9 A(2)

<ipython-input-1-cb9f0c9139a7> in B(x)
      2     x / (x-x)
      3 def B(x):
----> 4     C(x)
      5
      6 def A(x):

<ipython-input-1-cb9f0c9139a7> in C(x)
      1 def C(x):
----> 2     x / (x-x)
      3 def B(x):
      4     C(x)
      5

ZeroDivisionError: division by zero
```

Catching Exceptions in Python

- In Python, exceptions can be handled using a try statement.
- 在 Python 中，可以使用 try 语句处理异常。
- A critical operation which can raise exception is placed inside the try clause and the code that handles exception is written in except clause.
- 可能引发异常的关键操作放在 try 子句中，并且将处理异常的代码编写在 except 子句中。
- If no exception occurs, except block is skipped and normal flow continues. But if any exception occurs, it is caught by the except block
- 如果没有异常发生，则跳过 Except 的内容，并继续正常流程。但是，如果发生任何异常，它将被 Except 捕获

Catching Exceptions in Python



```python
# import module sys to get the type of exception
import sys

randomList = ['a', 0, 2]

for entry in randomList:
    try:
        print("The entry is", entry)
        r = 1/int(entry)
        break
    except:
        print("Oops!",sys.exc_info()[0],"occured.")
        print("Next entry.")
        print()
print("The reciprocal of",entry,"is",r)
```

```
The entry is a
Oops! <class 'ValueError'> occured.
Next entry.

The entry is 0
Oops! <class 'ZeroDivisionError'> occured.
Next entry.

The entry is 2
The reciprocal of 2 is 0.5
```

Catching Specific Exceptions in Python

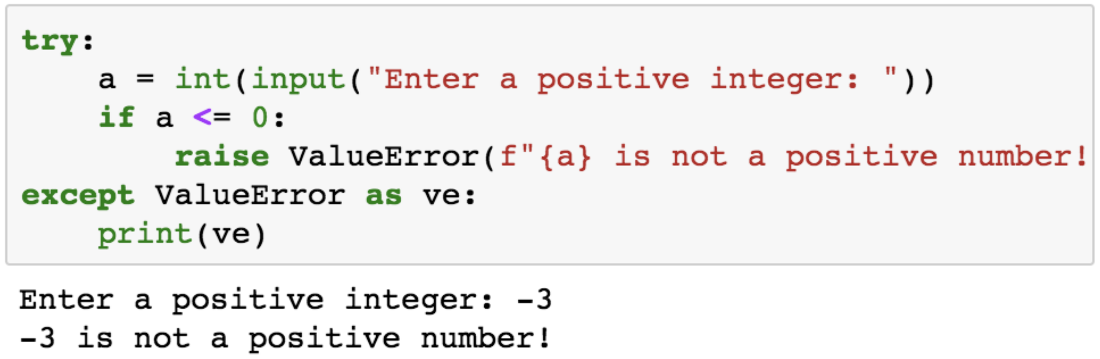
```python
try:
   # do something
   pass

except ValueError:
   # handle ValueError exception
   pass

except (TypeError, ZeroDivisionError):
   # handle multiple exceptions
   # TypeError and ZeroDivisionError
   pass

except:
   # handle all other exceptions
   pass
```

```python
# import module sys to get the type of exception
import sys

randomList = ['a', 0, 2]

for entry in randomList:
    try:
        print("The entry is", entry)
        r = 1/int(entry)
        break
    except ValueError:
        print("Value Error")
    except (ZeroDivisionError):
        print("ZeroDivision Error")
print("The reciprocal of",entry,"is",r)
```

```
The entry is a
Value Error
The entry is 0
ZeroDivision Error
The entry is 2
The reciprocal of 2 is 0.5
```

Rasing Exceptions

- 触发异常
- In Python programming, exceptions are raised when corresponding errors occur at run time, but we can forcefully raise it using the keyword raise.
- 在 Python 编程中，当运行时发生相应的错误时会引发异常，但是我们可以使用关键字 raise 强制引发它。
- We can also optionally pass in value to the exception to clarify why that exception was raised.
- 我们还可以选择将值传递给异常，以阐明引发该异常的原因。

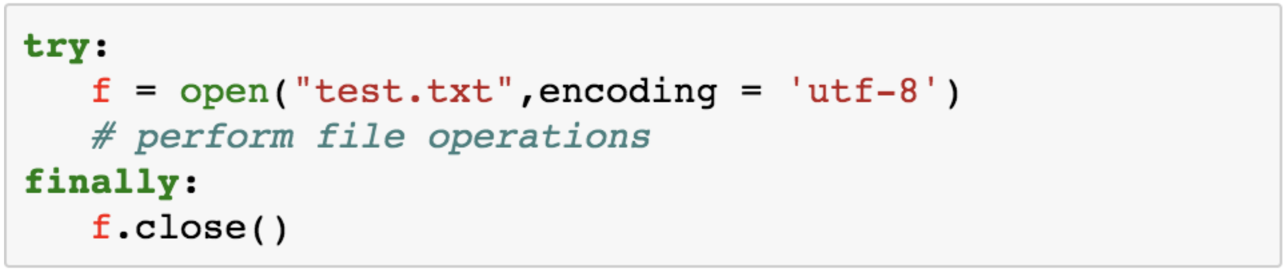


```
try:
    a = int(input("Enter a positive integer: "))
    if a <= 0:
        raise ValueError(f"{a} is not a positive number!
except ValueError as ve:
    print(ve)
```

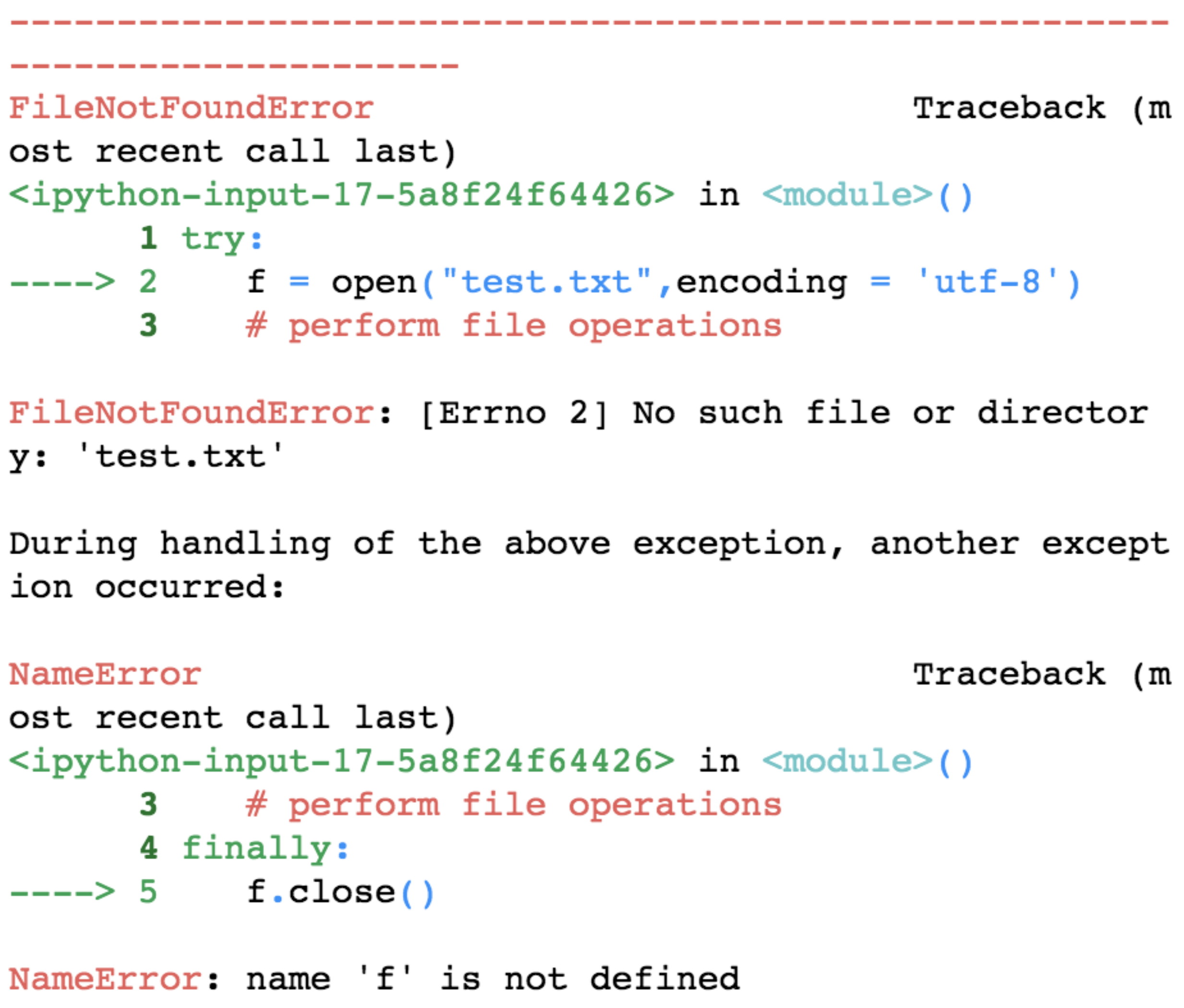
```
Enter a positive integer: -3
-3 is not a positive number!
```

Try...finally 语句

- The try statement in Python can have an optional finally clause. This clause is executed no matter what, and is generally used to release external resources.
- Python 中的 try 语句可以有一个可选的 finally 子句。该子句无论如何执行，通常用于释放外部资源。
- For example, we may be connected to a remote data center through the network or working with a file or working with a Graphical User Interface (GUI).
- 例如，我们可能通过网络或使用文件或使用图形用户界面（GUI）连接到远程数据中心。
- In all these circumstances, we must clean up the resource once used, whether it was successful or not. These actions (closing a file, GUI or disconnecting from network) are performed in the finally clause to guarantee execution.
- 在所有这些情况下，无论资源是否成功，我们都必



```
try:
    f = open("test.txt",encoding = 'utf-8')
    # perform file operations
finally:
    f.close()
```



```
---------------------------------------------------------------------
---------------------
FileNotFoundError                         Traceback (m
ost recent call last)
<ipython-input-17-5a8f24f64426> in <module>()
      1 try:
----> 2     f = open("test.txt",encoding = 'utf-8')
      3     # perform file operations

FileNotFoundError: [Errno 2] No such file or director
y: 'test.txt'

During handling of the above exception, another except
ion occurred:

NameError                                 Traceback (m
ost recent call last)
<ipython-input-17-5a8f24f64426> in <module>()
      3     # perform file operations
      4 finally:
----> 5     f.close()

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

Reference

- https://www.programiz.com/python-programming/exception-handling