Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

EXCEPTION HANDLING - 2

Guido Van Rossum

Agenda:

1. Default Exception Handling and Exception Hierarchy
2. Customized Exception Handling by using 'try – except' blocks
3. Control Flow in try-except
4. How to print exception information to the console?

3. Default Exception Handling and Exception Hierarchy

i. Default Exception Handling in Python

- Every exception in Python is an object. For every exception type the corresponding classes are available. Whenever an exception occurs, PVM will create the corresponding exception object and will check for handling code. If handling code is available then it will be executed and the rest of the program will be executed normally.
- If handling code is not available then Python Virtual Machine will terminates the program abnormally and prints corresponding exception information to the console. The rest of the program won't be executed.
- To prevent this abnormal termination, we should handle exceptions explicitly (By using try and except blocks).

Eg:

```
In [1]:
```

```
print('Hello')
print(10/0)
print('Hi')
```

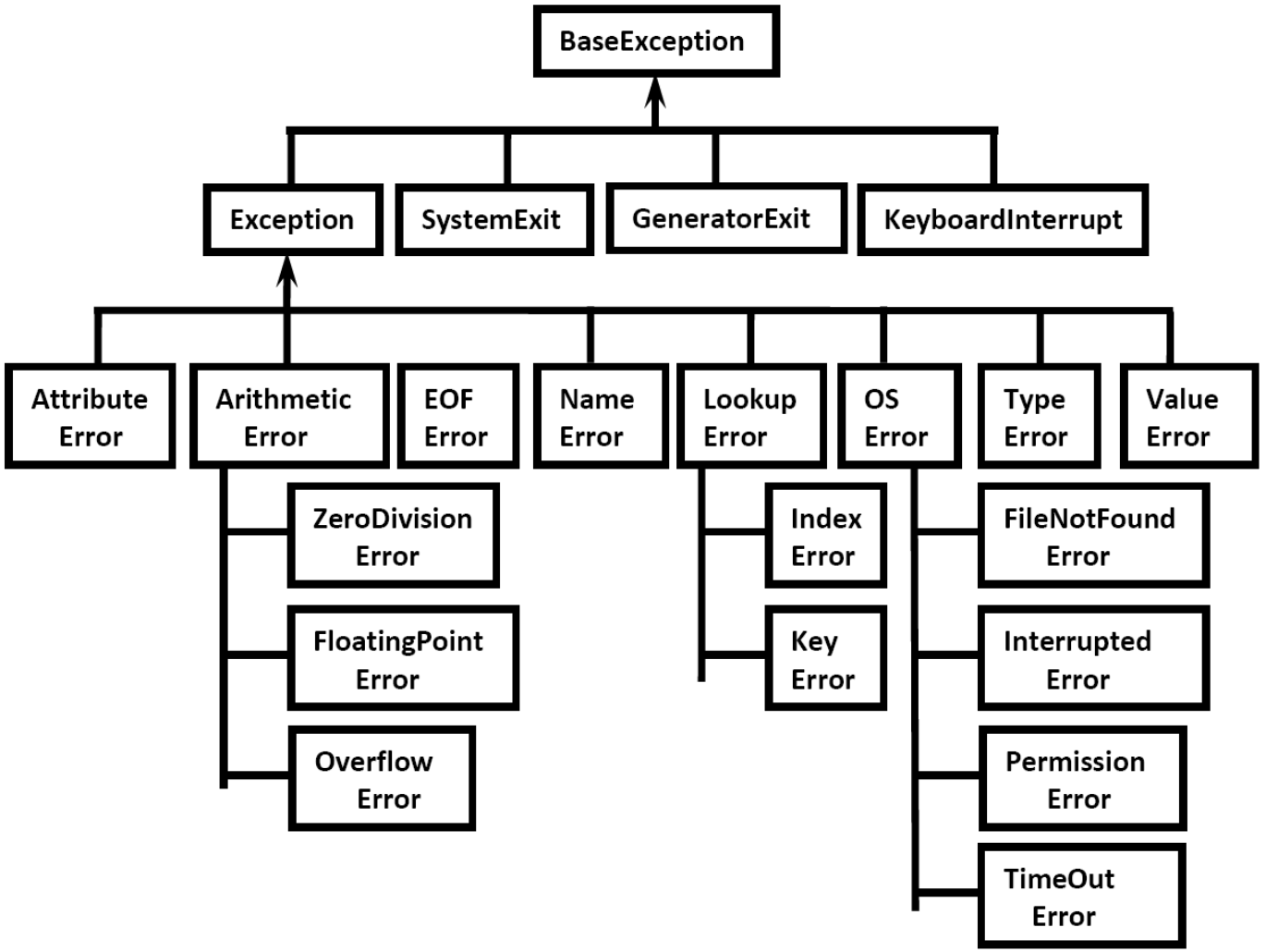
```
Hello

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-9fcbb5ba5ff8> in <module>
      1 print('Hello')
----> 2 print(10/0)
      3 print('Hi')

ZeroDivisionError: division by zero
```

This is nothing but **Abnormal Termination**.

ii. Python's Exception Hierarchy



Key Points:

- Every Exception in Python is a class.
- All exception classes are child classes of BaseException. i.e., every exception class extends BaseException either directly or indirectly. Hence BaseException acts as root for Python Exception Hierarchy.
- Most of the times being a programmer we have to concentrate Exception and its child classes.

4. Customized Exception Handling by using 'try - except‘ blocks

- It is highly recommended to handle exceptions.
- The code which may raise exception is called risky code and we have to take risky code inside try block.
- The corresponding handling code we have to take inside except block.

Syntax:

```
try:
        Risky Code                              ====>       Main code
except XXX:
        Handling code/Alternative Code          ====>       Alternative Code
```

Examples:

I. Without try-except:

```
print("stmt-1")
print(10/0)
print("stmt-3")
```

```
stmt-1

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-4-99f7ae7b0df9> in <module>
      1 print("stmt-1")
----> 2 print(10/0)
      3 print("stmt-3")

ZeroDivisionError: division by zero
```

The above code results into **Abnormal termination/Non-Graceful Termination.**

With try-except:

```
print("stmt-1")
try:
    print(10/0)
except ZeroDivisionError:
    print(10/2)
print("stmt-3")
```

```
stmt-1
5.0
stmt-3
```

The above code results into Normal termination/Graceful Termination.

5. Control Flow in try-except

Let us consider the below code,

```
try:
        stmt-1
        stmt-2
        stmt-3
except XXX:
    stmt-4
stmt-5
```

Case-1: If there is no exception

- 1,2,3,5 and Normal Termination.

Case-2: If an exception raised at stmt-2 and corresponding except block matched

- 1,4,5 Normal Termination.

Case-3: If an exception raised at stmt-2 and corresponding except block not matched

- 1, Abnormal Termination.

Case-4: If an exception raised at stmt-4 or at stmt-5 then it is always abnormal termination.

Important Conclusions:

1. Within the try block if anywhere exception raised then rest of the try block won't be executed even though we handled that exception. Hence we have to take only risky code inside try block and length of the try block should be as less as possible.
2. In addition to try block, there may be a chance of raising exceptions inside except and finally blocks (we will discuss later) also.
3. If any statement which is not part of try block raises an exception then it is always abnormal termination.

Note:

- except block is acts as a guard to the try block statements.

6. How to print exception information to the console?

Eg:

```
try:
    print(10/0)
except ZeroDivisionError as msg:
    print('The type of Exception :',type(msg))
```

```
The type of Exception : <class 'ZeroDivisionError'>
```

Here, **msg** is the reference variable to exception object.

Eg:

```
try:
    print(10/0)
except ZeroDivisionError as karthi:                    # you can use 'msg' or'karthi'
    print('The type of Exception :',type(karthi))
```

```
The type of Exception : <class 'ZeroDivisionError'>
```

Another Way:

```
try:
    print(10/0)
except ZeroDivisionError as msg:
    print('The type of Exception :',type(msg)) # Way 1
    print('The type of Exception :',msg.__class__) # Way 2
```

```
The type of Exception : <class 'ZeroDivisionError'>
The type of Exception : <class 'ZeroDivisionError'>
```

Eg:

```
try:
    print(10/0)
except ZeroDivisionError as msg:
    print('The type of Exception :',type(msg)) # Way 1
    print('The type of Exception :',msg.__class__) # Way 2
    print('The exception class name :',msg.__class__.__name__)
    print("Exception raised and its description is:",msg)
```

```
The type of Exception : <class 'ZeroDivisionError'>
The type of Exception : <class 'ZeroDivisionError'>
The exception class name : ZeroDivisionError
Exception raised and its description is: division by zero
```

Another Example:

```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print('The Result is :',x/y)
except BaseException as msg:
    print('The type of Exception :',type(msg)) # Way 1
    print('The type of Exception :',msg.__class__) # Way 2
    print('The exception class name :',msg.__class__.__name__)
    print("Exception raised and its description is:",msg)
```

```
Enter First Number: 10
Enter Second Number: 2
The Result is : 5.0
```

Eg:

```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print('The Result is :',x/y)
except BaseException as msg:
    print('The type of Exception :',type(msg)) # Way 1
    print('The type of Exception :',msg.__class__) # Way 2
    print('The exception class name :',msg.__class__.__name__)
    print("Exception raised and its description is:",msg)
```

```
Enter First Number: 10
Enter Second Number: 0
The type of Exception : <class 'ZeroDivisionError'>
The type of Exception : <class 'ZeroDivisionError'>
The exception class name : ZeroDivisionError
Exception raised and its description is: division by zero
```

Eg:

```
try:
    x=int(input("Enter First Number: "))
    y=int(input("Enter Second Number: "))
    print('The Result is :',x/y)
except BaseException as msg:
    print('The type of Exception :',type(msg)) # Way 1
    print('The type of Exception :',msg.__class__) # Way 2
    print('The exception class name :',msg.__class__.__name__)
    print("Exception raised and its description is:",msg)
```

```
Enter First Number: 10
Enter Second Number: two
The type of Exception : <class 'ValueError'>
The type of Exception : <class 'ValueError'>
The exception class name : ValueError
Exception raised and its description is: invalid literal for int() with base
10: 'two'
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You