Exception Handling

Pradipta Kumar Pattanayak

Department of Computer Science

Silicon Institute of Technology

ppattanayak@silicon.ac.in

What is an Exception

The errors also occur at runtime, and we know them as exceptions.

An exception is an event which occurs during the execution of a program and disrupts the normal flow of the program’s instructions.

Usually, the script handles the exception immediately. If it doesn’t do so, then the program will terminate and print a traceback to the error along with its whereabouts.

try:

Run this code

except:

Execute this code when there is an exception

else:

No exceptions? Run this code.

finally:

Always run this code.

Keyword used for exception handling

try: It will run the code block in which you expect an error to occur.

except: Here, you will define the type of exception you expect in the try block (built-in or custom).

else: If there is not any exception, then this block of code will be executed (consider this as a remedy or a fallback option if you expect a part of your script to produce an exception).

finally: Irrespective of whether there is an exception or not, this block of code will always be executed.

Example-1

```
try:
    a=10
    b=int(input("Enter a number: "))
    c=10//(a-b)
    print("The value of C=",c)
except:
    print("Value Error")
print("Program will continue")
```

Output:

```
Enter a number: 10
Value Error
Program will continue
```

Example-2

```
#Runtime Error....
a = [11, 22, 33]
try:
    print( "Second element ", a[1] )
    # Throws error since there are only 3 elements in list
    print ("Fourth element is : ", a[3])

except IndexError:
    print ("An Runtime error occurred")
print("Normal execution of the program")
```

Example-3

```
try:
    a=10
    b=a/0
    print("Hello")
except ZeroDivisionError:
    print("Division by zero")
print("Normal execution...")
```

output:

```
Division by zero
Normal execution...
```

finally block

- This block of code always execute.

```
try:
    x=10
    y=0
    c=x//y
except:
    print("Divisionby Zero error")
finally:
    print("Finally Block inside the function...")
print("Smooth execution...")
```

Output:

```
Divisionby Zero error
Finally Block inside the
function...
Smooth execution...
```

finally cluse

```
def fun(x,y):
    try:
        c=x//y
        return c
    finally:
        print("Finally Block inside the function...")
```

```
try:
    c=fun(10,20)
    #print"The value of C is ",c)
    print(c)
    c=fun(10,0)
    print(c)
except:
    print("Divisionby Zero error")
print("Smooth execution...")
```

```
Output:
Finally Block inside the function...
0
Finally Block inside the function...
Divisionby Zero error
Smooth execution...
```

else clause

else clause must be present after all the except clauses. The code enters the else block only if the try clause does not raise an exception.

Example:

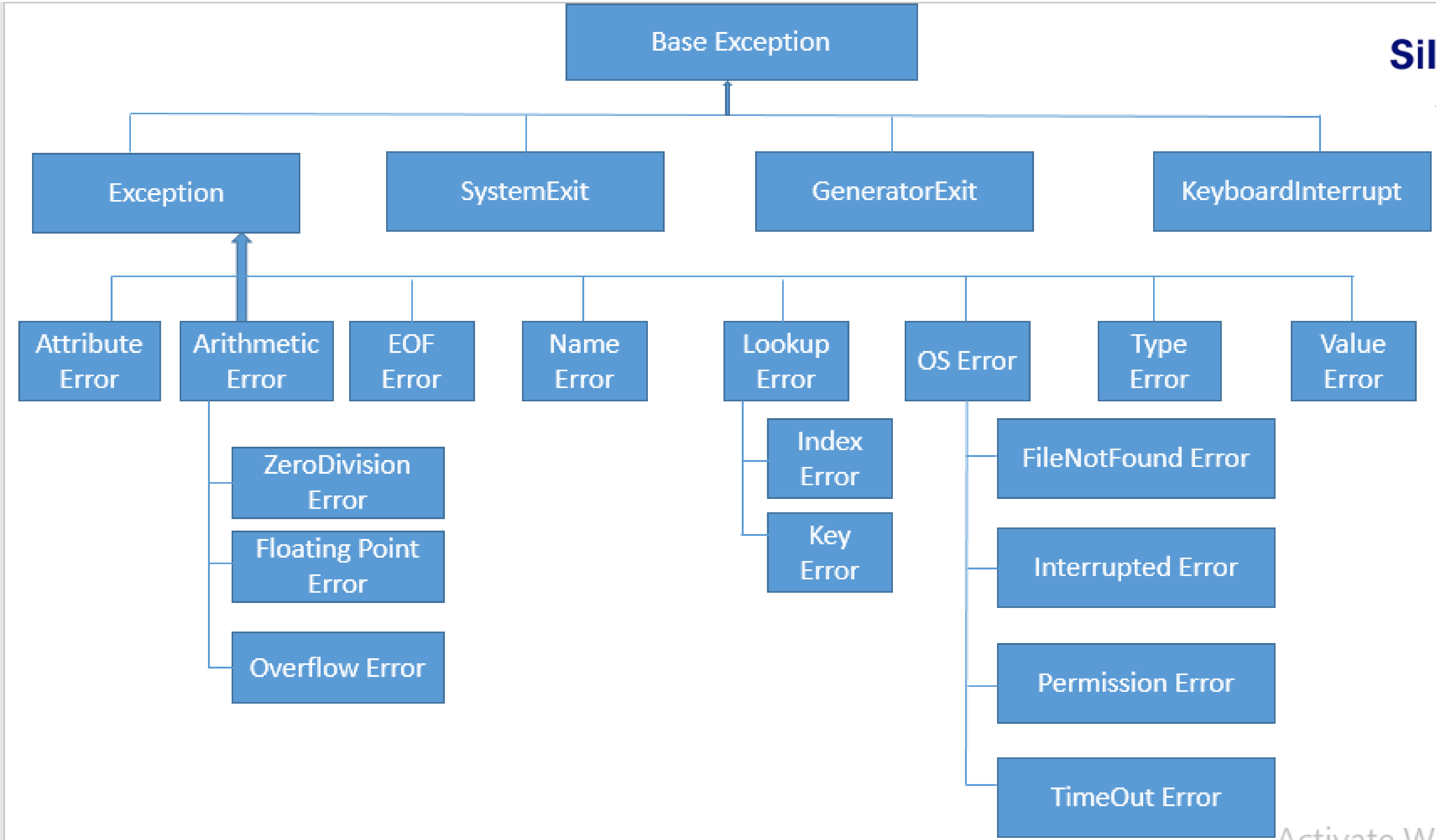
```
def myFun(a , b):
    try:
        c = ((a+b) / (a-b))
    except ZeroDivisionError:
        print( "a/b result in 0")
    else:
        print("The value of c=", c )
# Driver program to test above function
myFun(2.0, 3.0)
myFun(3.0, 3.0)
print("Normal Execution...")
```

Note: If we placed finally block, it should be placed after else clause.

Program to to handle multiple exception

```
try:
    f = open('fil.txt')
    print(f.read())
    f.close()
except ValueError:
    print('It failed')
except FileNotFoundError:
    print('File not found')
```

Output: File not found



Silicon
...beyond teaching