Exception Handling

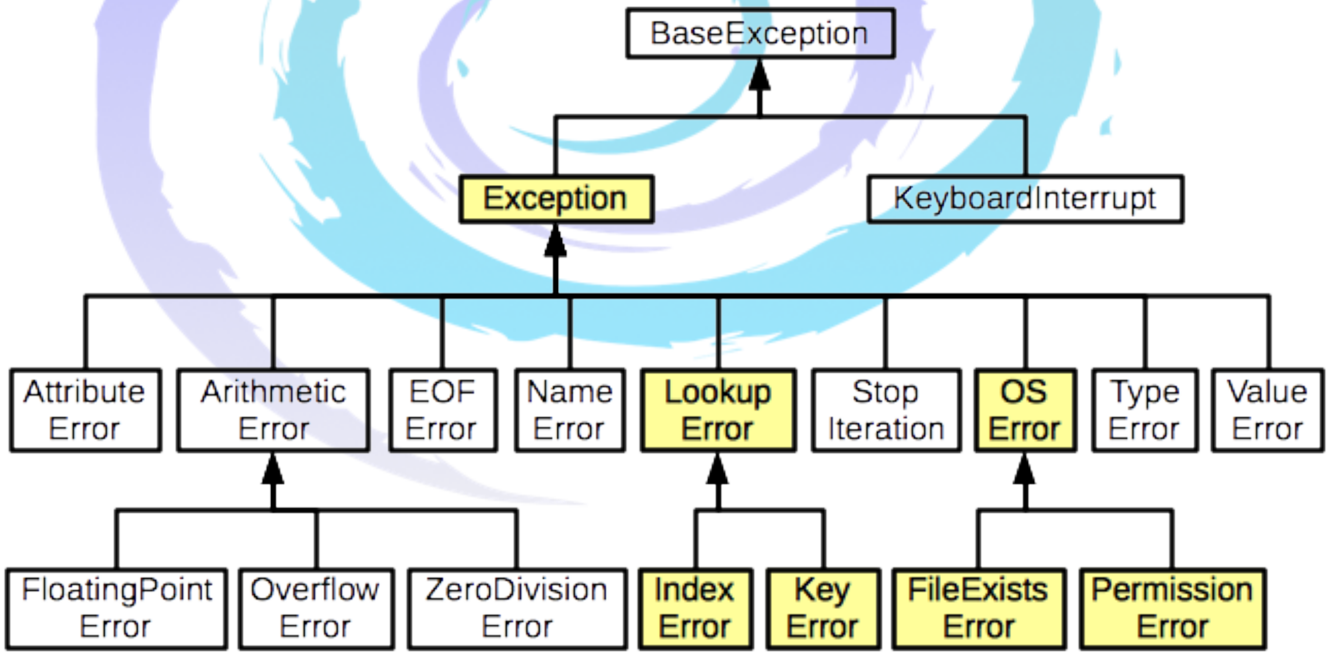
Exception :
An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program’s instructions.

Error vs Exception :

Error: An Error indicates serious problem that a reasonable application should not try to catch.

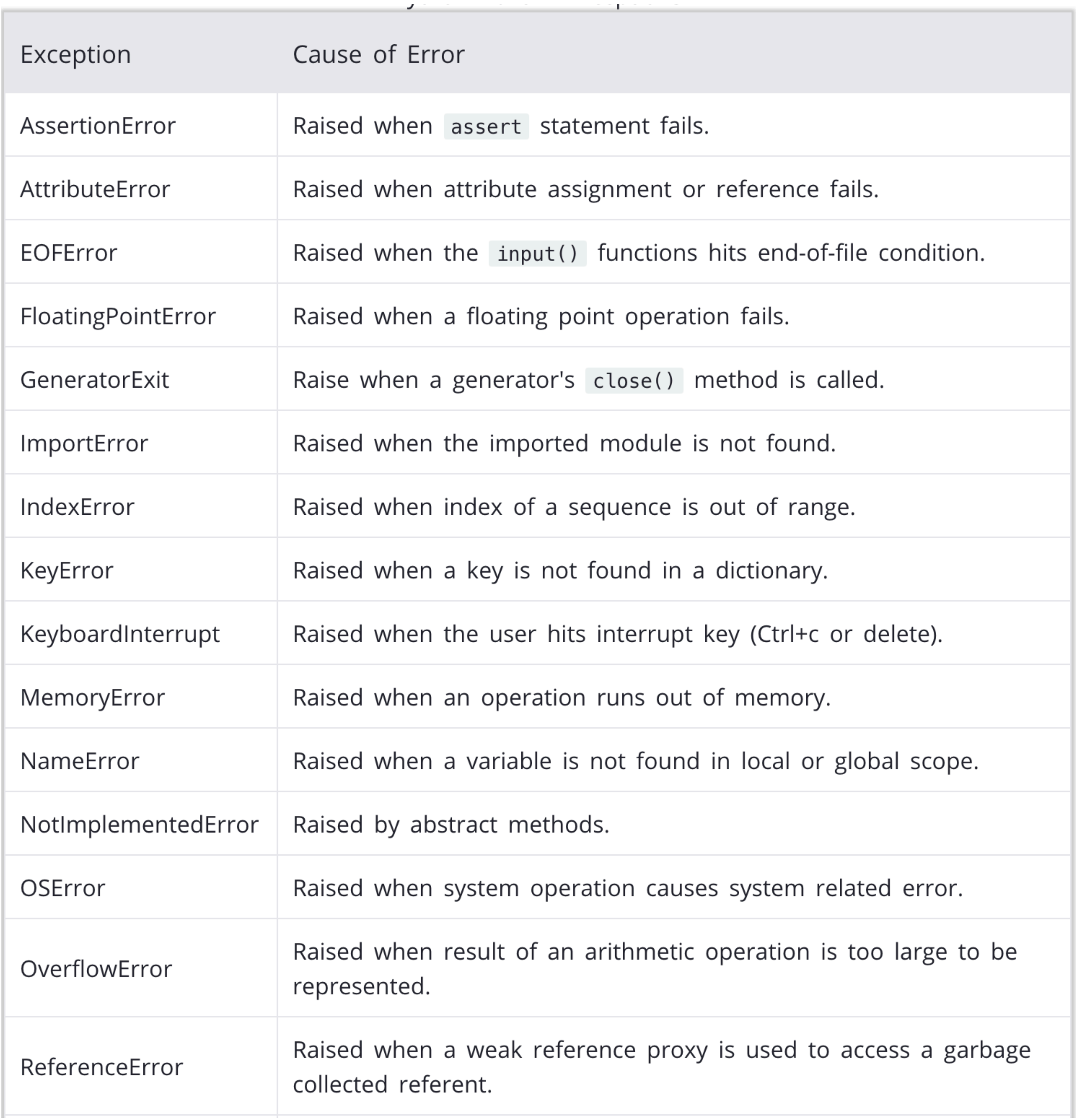
Exception: Exception indicates conditions that a reasonable application might try to catch.

Exception Hierarchy :
All exception and errors types are sub classes of class Exception, which is base class of exception hierarchy.
One branch is headed by Exception.
This class is used for exceptional conditions that user programs should catch.



Built in Exceptions in Python

According to above diagram there are multiple inbuilt exceptions as



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.
KeyboardInterrupt	Raised when the user hits interrupt key (Ctrl+c or delete).
MemoryError	Raised when an operation runs out of memory.
NameError	Raised when a variable is not found in local or global scope.
NotImplementedError	Raised by abstract methods.
OSError	Raised when system operation causes system related error.
OverflowError	Raised when result of an arithmetic operation is too large to be represented.
ReferenceError	Raised when a weak reference proxy is used to access a garbage collected referent.

RuntimeError	Raised when an error does not fall under any other category.
StopIteration	Raised by `next()` function to indicate that there is no further item to be returned by iterator.
SyntaxError	Raised by parser when syntax error is encountered.
IndentationError	Raised when there is incorrect indentation.
TabError	Raised when indentation consists of inconsistent tabs and spaces.
SystemError	Raised when interpreter detects internal error.
SystemExit	Raised by `sys.exit()` function.
TypeError	Raised when a function or operation is applied to an object of incorrect type.
UnboundLocalError	Raised when a reference is made to a local variable in a function or method, but no value has been bound to that variable.
UnicodeError	Raised when a Unicode-related encoding or decoding error occurs.
UnicodeEncodeError	Raised when a Unicode-related error occurs during encoding.
UnicodeDecodeError	Raised when a Unicode-related error occurs during decoding.
UnicodeTranslateError	Raised when a Unicode-related error occurs during translating.
ValueError	Raised when a function gets argument of correct type but improper value.
ZeroDivisionError	Raised when second operand of division or modulo operation is zero.

Internal working of PVM to handle an Exception

- Default Exception Handling : Whenever inside a python application, if an exception has occurred, the PVM creates an Object known as Exception Object and hands it off to the run-time system(PVM).
- The exception object contains name and description of the exception, and current state of the program where exception has occurred.
- Creating the Exception Object and handling it to the run-time system is called throwing an Exception.
- There might be the list of the methods that had been called to get to the method where exception was occurred.
- This ordered list of the methods is called Call Stack.

Now the following procedure will happen.

• The run-time system searches the call stack to find the method that contains block of code that can handle the occurred exception. The block of the code is called Exception handler. ie except block
• If it finds appropriate handler then it passes the occurred exception to it. Appropriate handler means the type of the exception object thrown matches the type of the exception object it can handle.
- If run-time system searches all the methods on call stack and couldn't have found the appropriate handler then run-time system handover the Exception Object to default exception handler , which is part of run-time system. This handler prints the exception information and terminates the program abnormally.

For Exception handling in Python we have to use three keyword as

try :
The code which is written inside try block is considered as an exception prone code means which may generate exception.

except :
This block is called as exception handler which gets executed when exception is occurred.

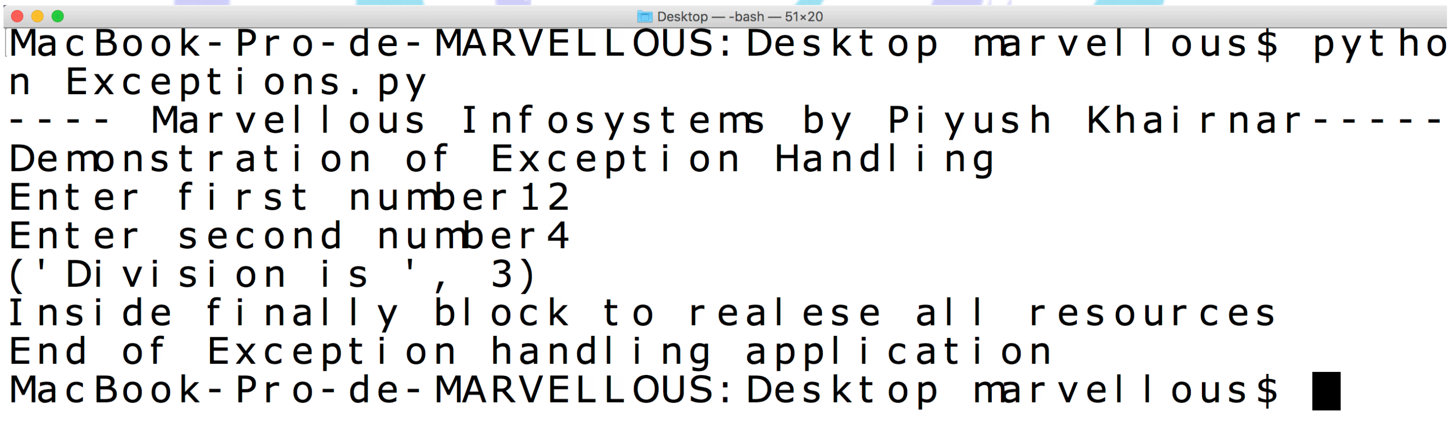
finally :
This block gets executed always irrespective of exception. Generally this block is used to release all resources.

In short the error handling is done through the use of exceptions that are caught in try blocks and handled in except blocks. If an error is encountered, a try block code execution is stopped and transferred down to the except block.
In addition to using an except block after the try block, you can also use the finally block.
The code in the finally block will be executed regardless of whether an exception occurs.

Consider below application which demonstrate concept of Exception Handling

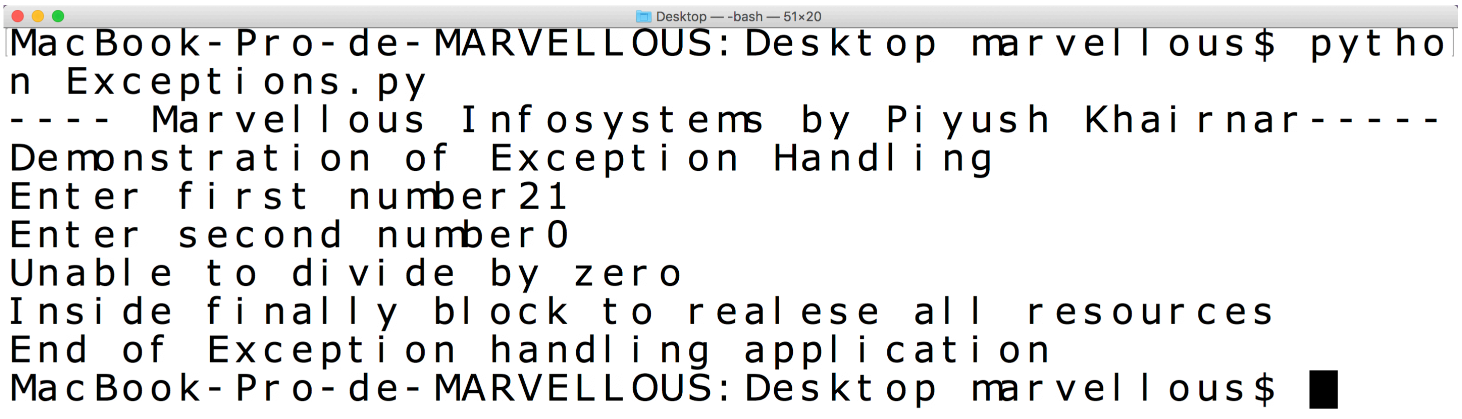
```
print("---- Marvellous Infosystems by Piyush Khairnar-----")

print("Demonstration of Exception Handling”)

no1 = int(input("Enter first number"))
no2 = int(input("Enter second number"))

try:
    ans = no1/no2
    print("Division is ",ans)

except ZeroDivisionError:
    print("Unable to divide by zero")

finally:
    print("Inside finally block to realese all resources")

print("End of Exception handling application”)
```

Output of Above application if we provide proper input



```
MacBook-Pro-de-MARVELLOUS:Desktop marvellous$ pytho
n Exceptions.py
---- Marvellous Infosystems by Piyush Khairnar-----
Demonstration of Exception Handling
Enter first number12
Enter second number4
('Division is ', 3)
Inside finally block to realese all resources
End of Exception handling application
MacBook-Pro-de-MARVELLOUS:Desktop marvellous$ █
```

Output of above application if we enter second number as Zero



```
MacBook-Pro-de-MARVELLOUS:Desktop marvellous$ pytho
n Exceptions.py
---- Marvellous Infosystems by Piyush Khairnar-----
Demonstration of Exception Handling
Enter first number21
Enter second number0
Unable to divide by zero
Inside finally block to realese all resources
End of Exception handling application
MacBook-Pro-de-MARVELLOUS:Desktop marvellous$ █
```