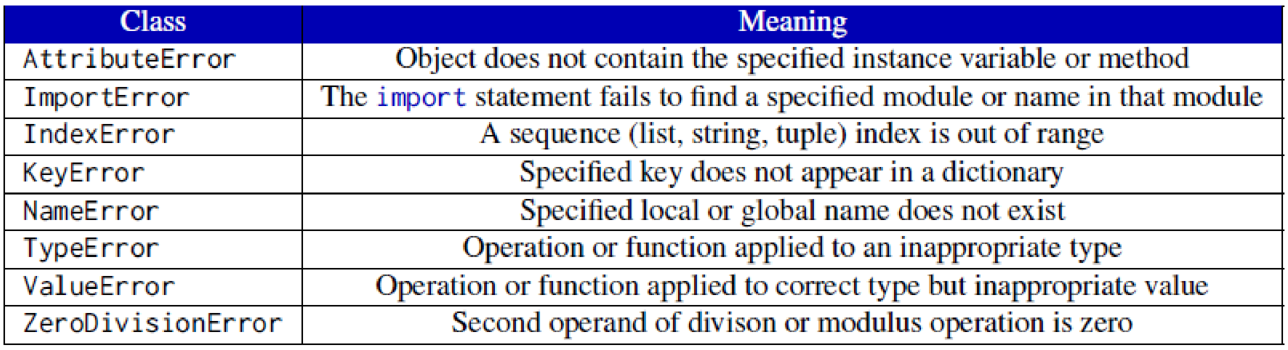
Exceptions II

Handling Different Types of Exceptions

This notebook demonstrates how we can handle different types of exceptions differently.



Class	Meaning
AttributeError	Object does not contain the specified instance variable or method
ImportError	The import statement fails to find a specified module or name in that module
IndexError	A sequence (list, string, tuple) index is out of range
KeyError	Specified key does not appear in a dictionary
NameError	Specified local or global name does not exist
TypeError	Operation or function applied to an inappropriate type
ValueError	Operation or function applied to correct type but inappropriate value
ZeroDivisionError	Second operand of divison or modulus operation is zero

There are many types of exceptions that can occur. The table above lists some of the common standard exception classes that have been predefined in Python. All the exception classes listed above inherit from the base Exception class.
Hence all of these exceptions can be handled in the same manner using a single generic except block. However, we can handle each exception type separately by writing specific exception handlers for each type.
For example, suppose we want to handle only the division by zero error. We can use that specific exception type in the except block.

In [1]:

```python
'''
  Use of ZeroDivisionError exception
'''

try:
    n1 = int(input('Enter the numerator:'))
    n2 = int(input('Enter the denominator: '))
    quo = n1/n2
    print('The quotient when dividing', n1, 'by', n2, 'is:', quo)
except ZeroDivisionError:
    print('ZeroDivisionError!! Division by zero is not allowed!!')
```

```
Enter the numerator:10
Enter the denominator: 0
ZeroDivisionError!! Division by zero is not allowed!!
```

But sometimes, the same code can generate multiple exceptions. For example, the code in the previous notebook, has the potential to generate two types of errors:

1. if the user enters zero for the denominator, and,
2. if the user enters a non-numeric value.

In the code in the cell above, any attempts to divide by zero will result in a ZeroDivisionError exception and will be handled by the except block. But any other type of exception, including the case where the user enters a non-numeric value, will not be handled and the program will terminate abruptly.

When multiple exceptions are possible and we want to treat each separately, we use more than one except block. The except blocks are executed in the order they are written. While it is not applicable in this example, suppose the same error belongs to two different categories, the first applicable except block will be executed.

In [2]:

```python
'''
  Use of multiple Except blocks
'''
try:
    n1 = int(input('Enter the numerator:'))
    n2 = int(input('Enter the denominator: '))
    quo = n1/n2
    print('The quotient when dividing', n1, 'by', n2, 'is:', quo)
except ZeroDivisionError:
    print('Division by zero is not allowed!!')
except ValueError:
    print('You need to enter a numeric value')
```

```
Enter the numerator:10
Enter the denominator: abc
You need to enter a numeric value
```