# Exceptions VII

## Creating Our Own Exception Classes

**In this notebook, we study how we can create and use our own exception classes.**

**All user defined exception classes are derived from the base class called `Exception`. To create your own user defined class, you need have nothing but the class heading.**

In [2]:

```
'''
 User defined exception class
'''
class DivisionByTwoError(Exception):
    pass #This statement is only required for syntactic purposes.
```

In [3]:

```
'''
 User defined exception class
'''
class DivisionByThreeError(Exception):
    '''
    The string returned by the redefined __str__ method, will be printed when we print the excepti
on object.
    '''
    def __str__(self):
        return 'Numbers divisible by three are not acceptable'
```

**In this example, we raise an DivisionByThree exception if the user enters a numerator value that is divisible by three. We raise a `DivisionByTwo` exception if the numerator value is divisible by two. A `ZeroDivisionError` exception will be automatically raised, if the user enters zero for the denominator.**

In [4]:

```
'''
  Raising user defined and automatic exceptions
'''
try:
    n1 = int(input('Enter the numerator:'))
    n2 = int(input('Enter the denominator: '))
    if n1 % 3 == 0:
        raise DivisionByThreeError
    if n1 % 2 == 0:
        raise DivisionByTwoError
    quo = n1/n2
    print('The quotient when dividing', n1, 'by', n2, 'is:', quo)
except ZeroDivisionError:
    print('Division by zero is not allowed!!')
except DivisionByThreeError as e:
    # Note that the string variable of the exception object is also printed
    print('Numerator cannot be divisible by Three!!\n', e)
except DivisionByTwoError:
    print('Numerator cannot be divisible by Two!!\n')
except Exception:
    print('Error in input!')
```

```
Enter the numerator:15
Enter the denominator: 12
Numerator cannot be divisible by Three!!
 Numbers divisible by three are not acceptable
```

**This example is similar to the one above, except that it now includes an else block which contains within it another `try/except` block**

In [6]:

```
'''
  Nested try/except blocks
  First try/except block: guards against errors generated by first number (DivisionByTwo,
DivisionByThree, Other)
  Second try/except block: guards against errors generated by second number (ZeroDivisionError, Ot
her)
'''
try:
    n1 = int(input('Enter the numerator:'))
    if n1 % 3 == 0:
        raise DivisionByThreeError
    if n1 % 2 == 0:
        raise DivisionByTwoError
except DivisionByTwoError:
    print('Numerator cannot be divisible by Two!!\n')
except DivisionByThreeError as e:
    print('Numerator cannot be divisible by Three!!\n',e)
except Exception:
    print('Error!')
else:
    try:
        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 Exception:
        print('Error!')
```

```
Enter the numerator:7
Enter the denominator: 0
Division by zero is not allowed!!
```