Exceptions I

This series of notebooks discusses Exception Handling in Python

This notebook demonstrates the need for exception handling. We begin with an example which has the potential to result in an error and stop the program abnormally. Then we introduce the *Look Before You Leap* concept.
Finally, since LBYL may not work well in all situations, we introduce Exception Handling. Note that when an exception is raised, an exception object is created and passed on to the `except` clause.

Consider the following program that accepts two integers and returns the quotient. There is no way of controlling for the fact that the user may enter 0 for the denominator.

In [2]:

```
'''
  Erroneous input
'''
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)
```

```
Enter the numerator:12
Enter the denominator: 0


---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-6635fbc25c76> in <module>()
      6 n2 = int(input('Enter the denominator: '))
      7
----> 8 quo = n1/n2
      9 print('The quotient when dividing', n1, 'by', n2, 'is:', quo)

ZeroDivisionError: division by zero
```

We can solve the problem described above in the following manner (referred to as *Look Before You Leap (LBYL)*:

In [3]:

```
'''
  Erroneous input
'''
n1 = int(input('Enter the numerator:'))
n2 = int(input('Enter the denominator: '))

#Before dividing the numerator by the denominator, make sure that the denominator is not zero.
if n2 == 0:
    print('Division by zero is not allowed!!')
else:
    quo = n1/n2
    print('The quotient when dividing', n1, 'by', n2, 'is:', quo)
```

```
Enter the numerator:21
Enter the denominator: 0
Division by zero is not allowed!!
```

However, what if the user enters a non-numeric value. The LBYL policy does not work well in such situations. Hence the need for - *Easier To Ask for Forgiveness than Permission (EAFP)* - or in other words, *Exception Handling.*

If the user enters an illegal value (either 0 or a non-numeric value), an exception is raised and an exception object is generated. Control of the program then goes to the except block. Exception handling involves the writing of code that responds when exceptions are raised, and prevents the program from crashing.
In Python exception handling is achived through the `try/except` statement block. The `try` block includes the code that could

potentially raise an exception while the `except` block contains the code to handle such execptions

In [ ]:

```
'''
  First Exception Handling Program
'''
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 Exception:
    print('Error in input!!')
```