Exceptions - III

The Untyped Except block. The exception object

Sometimes your code can generate exceptions that you did not anticipate. In such cases, you may want to include an untyped (generic) except block, like we used in the third example in the first notebook.

Note: an untyped except block has to be the last except block.

Write a program to read data from a file. Each line is two integer values. For each pair of integers, write a line to a new text file containing the two numbers together with the quotient. We desire to handle `ValueError` and `ZeroDivisionError` exceptions separately. All other exceptions can be handled using an untyped except block.

In [16]:

```
'''
  Specific (typed) and generic (untyped) exception handling
'''
try:
    f = open('input.txt', 'r')
    f2 = open('output.txt', 'w')
    for line in f:
        n_lst = line.split(',')
        n1 = int(n_lst[0])
        n2 = int(n_lst[1])
        quo = n1/n2
        f2.write('{}, {}, {0.2f}\n'.format(str(n1), str(n2), str(my_quo)))
    f.close()
    f2.close()
except ZeroDivisionError: #This block of code will handle ZeroDivisionError exceptions
    print('Division by zero is not allowed!!')
except ValueError: #This block of code will handle exceptions due to non-numeric values being ente
red by the user
    print('You need to enter a numeric value')
    '''
    The block below is called an untyped except block.  This block captures all exceptions, other
than the ZeroDivisionError
    and ValueError exceptions, that the program may generate. An untyped except block must be the
last except block.
    It cannot appear before a typed except block.
    '''
except Exception:
    print('Some other error')
```

```
Some other error
```

In the previous example, the message 'Some other error' is not very explanatory. We may want more information about the type of error that occurred. In this case, we can "catch" the exception object that is generated and print out the information that is contained inside.

In [17]:

```
'''
 Specific (typed) exception handling
 Catching Generic (untyped) exceptions
'''
try:
    f = open('input.txt', 'r')
    f2 = open('output.txt', 'w')
    for line in f:
        n_lst = line.split(',')
        n1 = int(n_lst[0])
        n2 = int(n_lst[1])
        quo = n1/n2
        f2.write('{0}, {1}, {0.2f}\n'.format(str(n1), str(n2), str(my_quo)))
    f.close()
    f2.close()
except ZeroDivisionError:
```

```
    print('Division by zero is not allowed!!')
except ValueError:
    print('You need to enter a numeric value')
except Exception as e: # To catch the exception object
    print('Some other error', e) #Here we print the contents
```

```
Some other error name 'my_quo' is not defined
```