RAISING EXCEPTIONS

You can *raise* exceptions using the `raise` statement by providing the name of the error/exception and the exception object that is to be *thrown*.

The error or exception that you can raise should be a class which directly or indirectly must be a derived class of the `Exception` class.

Example (save as `exceptions_raise.py`):

```
class ShortInputException(Exception):
    '''A user-defined exception class.'''
    def __init__(self, length, atleast):
        Exception.__init__(self)
        self.length = length
        self.atleast = atleast

try:
    text = raw_input('Enter something --> ')
    if len(text) < 3:
        raise ShortInputException(len(text), 3)
```

```
    # Other work can continue as usual here
except EOFError:
    print 'Why did you do an EOF on me?'
except ShortInputException as ex:
    print ('ShortInputException: The input was ' + \
           '{0} long, expected at least {1}')\
          .format(ex.length, ex.atleast)
else:
    print 'No exception was raised.'
```

Output:

```
$ python exceptions_raise.py
Enter something --> a
ShortInputException: The input was 1 long, expected at least 3


$ python exceptions_raise.py
Enter something --> abc
No exception was raised.
```

How It Works

Here, we are creating our own exception type. This new exception type is called ShortInputException. It has two fields -length which is the length of the given input, and atleast which is the minimum length that the program was expecting.

In the except clause, we mention the class of error which will be stored as the variable name to hold the corresponding error/exception object. This is analogous to parameters and arguments in a function call. Within this particular except clause, we use the length and atleast fields of the exception object to print an appropriate message to the user.

Source: http://www.swaroopch.com/notes/python/