Error Handling and Exceptions

Objectives

To try out Python exception handling within a module environment

Reference Material

Chapter 12 Error Handling and Exceptions.

Questions

1. Recall the **mytimer** module we worked on after Chapter 10 Modules and Packages. There were two functions, **start_timer()** and **end_timer()**, which should be called in that order. What if **end_timer()** was called without a **start_timer()** before it? We need to raise an exception in our timer module if that happens.

   Use your **mytimer.py** or the one from the solutions directory. You will have to detect if **start_timer()** was called previously from the **end_timer()** function. We suggest that you reset your global time to zero in **end_timer()** after a successful run, and test that. Which exception would be appropriate to raise?

   Test it using **Ex12.py**.

2. Now, in **Ex12.py**, handle the error elegantly with an appropriate error message.

If time allows…

Ex12.py opens and reads the words file. What happens if that file does not exist? Handle that exception in an elegant manner as well.

3. In a previous optional exercise we created a class called **File**. If you did not complete that exercise then take file.py from the solution directory for 11 Classes and OOP.

   Handle an `IOError` in the constructor for **File**. Create a new attribute called **`_error`**, which should be False if the file is created successfully, but set to the exception arguments if there was an `IOError`.

   In the **`size`** method, return the file size (as before) if the object was created without an error, otherwise return **`None`**.

   Define a new property which returns the value of the **`_error`** attribute.

   Test your code. We suggest you create a directory and use that directory name for creating a file. Output an error message if there is an error with the file.

Solutions

1. Choosing which exception is not so easy. The nearest we could think of is SystemError. Given more time we might invent our own exception subclass. Raising the exception is fairly easy:

```
def end_timer(txt='End time'):
    """...
    """
    global start_time
    if start_time == 0:
        raise SystemError(
            "end_timer() called without a start_timer()")
    (utime,stime) = os.times()[0:2]
    end_time = utime+stime
    print "%-12s: %05.3f seconds" % \
          (txt,end_time-start_time)

    start_time = 0
```

2. Detecting the error is also fairly straightforward:

```
try:
    mytimer.end_timer()
except SystemError, err:
    print >> sys.stderr,"end_timer error:",err
```

If time allows:

```
try:
    for row in open ("words"):
        lines += 1
except IOError as err:
    print >> sys.stderr,"Could not open:", \
           err.filename, err.args[1]
```

3.

```
import os.path
import struct

class File(object):
    def __init__(self,filename):
        self._filename = filename
        self._error = False

        # If the file does not exist, create it
        if not os.path.isfile(filename):
            try:
                open(filename,'w')
            except IOError as err:
                self._error = err.args

    @property
    def size(self):
        if self._error:
            return None
        else:
            return os.path.getsize(self._filename)

    @property
    def error(self):
        return self._error

# Text file
class TextFile(File):
    @property
    def contents(self):
        """ Return the contents of the file """
        return open(self._filename,'rt').read()

    @contents.setter
    def contents(self,value):
        """ Append to the file """
        if not value.endswith("\n"):
            value += "\n";
        open(self._filename,'at').write(value)

# Binary file
class BinFile(File):
    @property
    def contents(self):
        """ Return the contents of the file """
        value = open(self._filename,'rb').read()
        return value

    @contents.setter
    def contents(self,value):
        """ Append to the file """
        if isinstance(value,int):
            out = struct.pack('i',value)
            open(self._filename,'ab').write(out)
        else:
            open(self._filename,'ab').write(value)
```

```
if __name__ == '__main__':
    import sys

    # Test constructor error handling
    if not os.path.isdir:
        os.mkdir('Dummy')

    dummy = TextFile('Dummy')
    print "Size of Dummy:",dummy.size

    if dummy.error:
        print >> sys.stderr,"Dummy error:",dummy.error
    else:
        print >> sys.stderr,"No error detected!"
```