Exercise 12 – Error Handling and Exceptions

Objective

To try out Python exception handling within a module environment.

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 initialise and reset your global time to **None** in **end\_timer()** after a successful run, and test that. We use **None** because zero is a valid time. Which exception would be appropriate to raise?

Test it using **Ex12.py**.

1. 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 and elegant manner as well.

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

Handle an IOError in the constructor for **MyFile**. 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 \_\_**len\_\_** 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

Here are our versions of these exercises, remember that yours can be different to these, but still correct. If in doubt, ask your instructor.

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 easy:

def end\_timer(txt='End time'):

"""...

"""

**global start\_time**

**if start\_time is None:**

**raise SystemError(**

**"end\_timer() called without a start\_timer()")**

**end\_time = os.times()[:2]**

print("{0:<12}: {1:01.3f} seconds".

format(txt, end\_time - start\_time))

**start\_time = None**

**return**

1. Detecting the error is also fairly straightforward:

try:

mytimer.end\_timer()

except SystemError as err:

print("end\_timer error:", err, file=sys.stderr)

**If time allows:**

try:

for row in open("words"):

lines += 1

except IOError as err:

print("Could not open:",

err.filename, err.args[1], file=sys.stderr)

**Question 3**

import os.path

import struct

class File:

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

def \_\_len\_\_(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)

return

# Binary file

class BinFile(File):

@property

def contents(self):

""" Return the contents of the file """

value = open(self.\_filename, 'rb').read()

return value.decode()

@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.encode())

if \_\_name\_\_ == '\_\_main\_\_':

import sys

# Test constructor error handling

if not os.path.isdir():

os.mkdir('Dummy')

dummy = TextFile('Dummy')

print('Size of Dummy:', len(dummy))

if dummy.error:

print('Dummy error:', dummy.error, file=sys.stderr)

else:

print('No error detected!', file=sys.stderr)