Assignment 2

Create a very simple inheritance hierarchy of three classes that write to text files.

**LogFile(WriteFile):** its instance writes a date and message to a log file:

2015-01-21 18:35 this is a log message

**DelimFile (WriteFile):** its instance writes values separated by a delimeter:

a,b,c,d

**WriteFile(object):** the parent class to both **LogFile** and **DelimFile**, does work that is common between them. Not intended to be instantiated.

Here is calling code you can use to test:

log = LogFile('log.txt') # passes the filename to write to

mydelim = DelimFile('data.csv', ',') # passes the filename to write to

# and a delimeter

log.write('this is a log message') # writes a message to the file

log.write('this is another log message') # same

mydelim.write(['a', 'b', 'c', 'd']) # writes a list of values separated

# by comma to the file

mydelim.write(['1', '2', '3', '4']) # same

Here's what the two files look like when we're done:

# text of log.txt

2015-01-21 18:35 this is a log message

2015-01-21 18:35 this is another log message

# text of data.csv

a,b,c,d

1,2,3,4

As you can see, both of these classes write to files (so they are similar), but they write to them in different ways (so they are different).

**LogFile:**

* constructor takes a log filename as an argument
* when we call **write()** on the LogFile object and pass a string argument, it writes a log message, preceded with the date and time
* this class would be useful in any application in which you want to keep track of what is happening as a script executes

**DelimFile:**

* constructor takes a filename and a delimeter as arguments
* when we call write() on the DelimFile object and pass a list, it then writes to the file by joining the elements of the list with the delimeter.
* delimited files are often used to store and transfer data; for example, Excel can output its data in a CSV format, or comma-separated values format. Really, any character can be used for a delimeter, but commas, colons and tabs are the most common.

**WriteFile:**

* is not designed to be instantiated (constructed)
* is the parent class of both LogFile and DelimFile
* handles tasks that both classes need

This is a relatively uncomplicated assignment, but the basic challenge in designing your three classes will be in avoiding duplication of code, and in producing a solution in as few lines as possible. Obviously there are some tasks that are common between the two classes, and some that are specific. Your assignment is to create a parent class, **WriteFile**, that holds the common functionality, which may include part of the constructor tasks, and have **LogFile** and **DelimFile** inherit from **WriteFile**, drawing upon its methods as necessary. The key then would be to figure out the shortest possible solution using inheritance.

You'll want to make sure that the **write()** method is implemented in eacb of **DelimFile** and **LogFile**, and that it is an abstract class in the parent, **WriteFile**.

For the log file date, we'll use the **datetime** module. Just for convenience, here's code for creating the datetimestamp shown in the otuput:

import datetime

dt\_str = datetime.datetime.now().strftime('%Y-%m-%d %H:%M')

Print **dt\_str** to see the result.

Optionlly, you can allow for the delimeter to be included in any field value. Note the quotation marks around the second field in the first line of **text.csv** in this example:

c.write(['a', 'this, that', 'c', 'd'])

$ cat text.csv

a,b,c,d

1,2,3,4

a,"this, that",c,d

This has been done because the input actually has the delimeter character in it, and without this delineation of the field value it would be mistaken for a delimeter if we tried to parse the fields. This is a convention of the CSV format. Although we are not attempting to match the full spec for a CSV file, this file can be opened as a spreadsheet in Excel.

For extra credit (or just a little more challenge, since we're not really giving credit in this course), solve the same assignment using composition instead of inheritance.

* **WriteFile** should be instantiated with two arguments: a filename and a writer class (either **LogFile** or **DelimFile**)
* the class will open the file as before, but will also instantiate and store an instance of the class passed as argument.
* in this case, have **DelimFile()** use a comma as the default delimeter, so that instantiating the **DelimFile** object doesn't require that we include a delimeter
* when the user calls **write()**, the instance will use the stored instance to do the writing, calling **write()** on it -- and since both **LogFile** and **DelimFile** have a **write()** method, it should work (as long as the argument to **write()** is correct)

Good luck!