Assignment 2

Due Date: This assignment is due at 11:59:59PM on 11 October, 2013.

Submission: Solutions should be uploaded in a single-file attachment to the appropriate Assignment on `http://newclasses.nyu.edu`

Style: This assignment will be graded, in part, by automated testing, so all classes, members, methods, etc. must adhere **precisely** to that specified. Code should be well-commented, logically packaged, and easy to read. Variable names should be self-documenting wherever possible. No line should exceed 80 characters.

This assignment will require the use of all the skills you have acquired up to this point, including shell scripting, error handling, calling modules, and object orientation. If you are lost, make liberal use of the books, online resources, and office hours. The maximum grade, including extra credit, is **150 points**. This Assignment is longer than the previous, and will be worth **1.5x** as much in the final calculation. Please note that **in order to be eligible for extra credit, the assignment must be turned in by the initial deadline - extra credit cannot be used to compensate for lateness.**

You will be building a command-line utility to conveniently rename `.mp3` files according to the data held inside their ID3 tags. ID3 is a standard format used in MP3s to hold information about, among other things, a song's title, album title, artist, and genre.

To achieve this, you will be creating a shell script that contains several interdependent functions and commands that use these functions in order to accomplish your task. Please **read carefully** and **make sure you test each function independently**. Also take care that when submitted, your script does not contain any extraneous code. **No code may make use of globals other than imports.** Significantly, nothing other than global statements and function calls may make use of `sys.argv`; arguments **must** be passed in function calls. All output in function specifications is via `return` statements, all exceptions are thrown via `raise`.

I will assume that all of you have some library of MP3-formatted music; if this is not the case, there are many repositories online.

1. [**0 points**] Download and install Mutagen from `https://code.google.com/p/mutagen/`. Mutagen is a third-party library module that makes dealing with ID3 tags straightforward. Mutagen has no other third-party module dependencies, so installation should be straightforward - see the `README` file in the archive.

All further work will be done inside a script you **must** name `assignment2.py`; failure to do so will result in a **5 point penalty**. The script should, of course, include the appropriate shebang line.

2. [**10 points**] Create a function:

   Name: `find_mp3s`
   Input(s): `path` - a `str`
   Output(s): a `list` of `str` describing absolute paths to mp3 files under `path`
   Throw(s): `ValueError` if `path` doesn't exist and `TypeError` if `path` is not a `str`

   This function should, given a path, use the `os` and `os.path` modules to find all files whose filenames end in `.mp3` underneath the provided `path`. If `path` doesn't exist, this function should `raise` a `ValueError` (but not catch it - this is the job of the caller of the function). This is very similar, but not identical, to Lab #8; in particular it is a function, not a script. Take care to comply with the specified inputs, outputs, and thrown exceptions.

3. [**20 points**] Create a function:

   Name: `load_id3`
   Input(s): `path` - a `str` describing the absolute path to an MP3 file
   Output(s): an `EasyID3` instance from the `mutagen.easyid3` module
   Throw(s): `ValueError` if `path` doesn't exist or if the ID3 initialization fails, and `TypeError` if `path` is not a `str`

   Add `from mutagen.easyid3 import EasyID3` to the top of your script, underneath existing `import` statements. This will bring the EasyID3 class into the local namespace. The ID3 initializer takes a path like so:

   ```
   id3 = EasyID3(path)
   ```

   Ensure that you are using `EasyID3` as the ID3 engine; the default ID3 implementation is significantly more difficult to use. It may be useful to read the mutagen tutorial at `https://code.google.com/p/mutagen/wiki/Tutorial`. The function should take a `path` to an MP3 file and return an ID3 instance representing the ID3 information from the MP3 referred to. If the `path` supplied is not a string (`str`), does not point to an MP3 file, or there is a problem initializing the `ID3` instance, this function should throw, but not catch, a `ValueError`, similar to `find_mp3s`. Please test to ensure that no other exceptions are thrown for other input types or invalid paths.

4. [**20 points**] Create a function:

   Name: `make_filename`
   Input(s): `id3` - an `EasyID3` instance
   Output(s): a `str` representing a proposed new filename for the MP3 file `id3` was extracted from (details below)
   Throw(s): a `TypeError` if `id3` is not an instance of `EasyID3`, or a `ValueError` if `id3` does not contain the necessary data (see below)

   `EasyID3` instances can be used like dictionaries, so to get e.g. an artist name from it, you simply call `id3['artist']`. ID3 tags can contain many different bits of metadata about an MP3 track; for the full list you may import the `EasyID3` module as specified earlier and `print EasyID3().valid_keys.keys()`.

   For our purposes, we are only interested in the artist, album, and name information. The output of this function should be of the form `'artist - album - title.mp3'`, e.g. `'Jonathan Coulton - Thing a Week Three - Code Monkey.mp3'`. A song's artist and title are required, but album is optional; if `id3['album']` is `None` or an empty string, the album name should be replaced by `'Unknown'`, e.g. `'Ylvis - Unknown - The Fox.mp3'`. If either artist or title are missing, throw a `ValueError` stating that the file is missing required data.

5. [**25 points**] Create a function:

   Name: `rename_mp3`
   Input(s): `input_mp3_path`, `output_mp3_path` - two `str` values describing the input MP3 file path and a desired output path (not including filename)
   Output(s): `None` (no `return` statement)
   Throw(s): a `ValueError` if either `input_mp3_path` or `output_mp3_path` does not exist (see below)

   This function will form the core of the script you are writing. For a given MP3 file specified as `input_mp3_path`, use `load_id3` to get its ID3 tags, `make_filename` to create a filename from its ID3 tag, and then use `shutil.copyfile()`[1] to make a copy, probably in a different place. `output_mp3_path` should be a folder/directory, not a file, and the `shutil.copyfile()` should use the output of `make_filename` appended to `output_mp3_path` as a destination (use `os.path.join()`). In order to use `shutil.copyfile()` you will, of course, need to add `import shutil` to the top of your file.

[1] See `http://docs.python.org/2/library/shutil.html` for details.

If you attempt to copy a file to a folder/directory that does not exist, `shutil.copyfile()` will throw an `IOError`. You must catch this exception using a `try/except` block and re-throw a `ValueError` using `raise` if this happens. If you call `make_filename` on an `id3` instance with insufficient information, it will throw a `ValueError` - this should propagate up to the caller, either by re-throwing using `raise` or by **not** catching it. If you call `load_id3` on a `input_mp3_path` that either doesn't exist or is not an MP3 file, it will throw a `ValueError` - this should be handled in the same way as in the previous case.

6. [**25 points**] The final component is to create the actual script. With `find_mp3s` you have a way of locating all `.mp3` files in a folder/directory, and with `rename_mp3` you have a function that will copy an MP3, renamed using its ID3 information, to another place. The script will take two paths, one source and one destination. You should:

   - Check the arguments to ensure they are valid and exit gracefully if they are not (see the Lab #8 solutions for an example)
   - Get a list of all `.mp3` files in the first argument, the source, using `find_mp3s`
   - For each (think for loop) `.mp3` file, use `rename_mp3` to copy it to the second argument, the destination
   - Wrap the inside of the loop, the execution of `rename_mp3` in a `try/except` block pair to print to the console that a particular `.mp3` file cannot be converted; name the file in the error message

7. [**50 points**] Extra Credit: This may prove challenging - ensure that your assignment is perfect up to this point and save it in a safe place for submission in case you cannot complete this part.

   Augment your script so that in addition to the source and destination paths, it takes a third string describing the format of the renaming of the MP3 filenames, comprised of the elements in `EasyID3().valid_keys.keys()`, e.g. `'artist - performer - conductor - title.mp3'`. You may assume that each element is separated from the others by `' - '` (that is, a hyphen surrounded by one space on either side). You will:

   - Remove the `.mp3` from the end of the third argument using either `.split()` from the `.str` class or a slice `[:-4]` or, if it is absent, report an error
   - Use `.split` from the `str` class to get a list of the ID3 elements to use in the filename
   - Alter `make_filename` to take a list of ID3 elements and create a filename based on the keys in the list, throwing an error if any of the keys is invalid (i.e. not in `EasyID3.valid_keys.keys()`)
   - Make `['artist','album','title']` the argument to the new `make_filename` if the third argument to the script is not provided, as a default