Homework 6: Objects & Classes

Due Wednesday, November 5, 3p

- hw6.py program
- hcde310photos.html output from your program

Part 0: Prepare

Part 0.1: Card.py exercises

We *strongly* recommend that you use lab time to work through some of the exercises on Card.py before starting this homework. They aren't related, but working through the exercises with people and then the homework on your own will help make sure that you have a solid handle on Objects & Classes.

Part 0.2: Get the homework files

You'll need the homework files. Get them by typing the following lines in terminal:

```
cd ~
./getHW.sh 6
```

Don't forget that if you already have Eclipse open, you may need to right click on the homeworks project and "refresh" it, to make the new folder hw6 appear.

Part 1: Processing the feed

Download `hw6.py` and `hw6_fbgroup.json`. It contains code and data that will start you off.

1.1 Overview

In this assignment, you will be creating classes that represent Status updates (Posts), Links, and Photos on Facebook. Using these classes, you will create objects corresponding to all of the post dictionaries from the HCDE310 Facebook group JSON stream. The object-based representation of posts will make it easy to print out summaries of posts in the feed, generate a web page with the photos that have been uploaded to the group, and selectively print out summaries of the questions asked on the group.

1.2 Building Blocks

There are three classes that represent things in the Facebook group feed: Post, Link, and Photo. We will start you off with the class definitions, method names, and `__init__()` methods, and you'll fill in the rest. We will also provide you with code in the main block

(code that follows `if __name__ == '__main__'` at the end of hw6.py) that you will use to test your methods.

Note: there are a lot of parts to this section, but don't panic: most of the methods can be implemented in 1-3 lines of code each. Implement each method, and as you go along, uncomment out the code in the main block (starting with Test 1).

1) `Post` is the base class that represents all of the basic data and operations that can be performed on a post. The constructor, `__init__()` will initialize all of the following instance variables using data from a post dictionary (called `post_dict`, which the constructor takes as an argument).

   The following are instance variables that you will use in your methods:

   - `message`: The contents of the post
   - `comments`: If there are comments in the post dictionary (i.e., '`comments`' is a key in `post_dict`), the comments instance variable will be a dictionary containing the number of comments and a list of all the comments. If there are no comments, the value of this instance variable is `None`.
   - `likes:` Same as comments, but for likes.
   - `name:` The short name of the user. You will have to fill in the code to create and initialize this variable yourself.

   In addition, Post contains several methods:

   - `__init__()`: Initializes all of the instance variables used in the methods. We have included most of the code here for you already. (a) fill in code that uses `short_name()` to initialize the name instance variable. Uncomment the test for (a) in the main block to ensure you have properly initialized name.
   - `comments_count()`: (b) Implement comment_count() by uncommenting the method definition in the Post class. Your method should return the number of comments if the post has comments. It should otherwise return 0. Uncomment the test for (b) to make sure the method works correctly.
   - `likes_count()`: (c) Implement `likes_count()` by uncommenting the method definition in the Post class. Your method should return the number of likes if the post has likes. It should otherwise return 0. Uncomment the test for (c) to make sure the method works correctly.
   - `short_message()`: (d) Implement `short_message()` by uncommenting the method definition in the Post class. If the message is less than 140 characters, it should return the message. If it is greater than 140 characters, it should return a string that is the first 140 characters of the message, plus '...'

(e.g. a message like 'this message is more than 140 characters so we should use the string slicing operator to get the first 140 characters and then concatenate that string with an ellipsis' would get returned as 'this message is more than 140 characters so we should use the string slicing operator to get the first 140 characters and then concatenate t...'). Uncomment the test for (d) to make sure the method works correctly.

- `__str__()`: returns the string representation of the Post object. We've given you the complete implementation, but commented out the lines that use name, `comments_count()`, `likes_count()`, and `short_message()`. Uncomment the code in `__str__()` once you are sure that parts (a-d) work correctly. Uncomment out the test for (e) in the main block to test `__str__()`. When you print a `Post` object, the output will follow the following format:

```
--- status update ---------
name: <name>
likes: <number of likes>
comments: <number of comments>
message: <short message>
```

2) `Link` is a subclass of `Post`, so it includes all of the functionality (data and methods) built into Post. In addition, it contains an instance variable called url, which is a string corresponding to the URL being shared in the post. You will implement two methods:
   - `__str()__`: (a) Implement `__str__()` by uncommenting the method definition in the `Link` class. The output should similar to `__str__()` for `Posts`, except it should also include a header that says 'link' and a field for the URL. e.g.,

```
--- link ------------------
name: <name>
likes: <number of likes>
comments: <number of comments>
message: <short message>
linked url: <url being shared>
```

   Uncomment the line corresponding to (2a) in the main block to test your__str__() method.

3) `Photo` is also a subclass of `Post`, so it includes all of the functionality (data and methods) built into `Post`. In addition, it contains two instance variables: `preview_url`, which corresponds to a thumbnail preview of the image being linked to, and `photo_url`, a URL that links back to the original Photo post on facebook.com with the full image. You will implement:
   - `__str()__`: (c) Implement `__str__()` by uncommenting the method definition in the `Photo` class. The output should similar to `__str__()` for

`Posts`, except it should also include a header that says ‘photo’ and fields for the URL of the thumbnail and full photos of the post. e.g.,

```
--- photo ------------------
name: <name>
likes: <number of likes>
comments: <number of comments>
message: <short message>
thumbnail: <url of the thumbnail photo>
photo: <url of the photo post>
```

1.3 Tests

Test 1: Text output

Once you have implemented all of the above methods and uncommented out the test code, your output should look like this:

```
-- TEST 1 --
- elements -
Korn T.
3
1
Anyone know how to force Google Drive to save the graph from HW5 at a higher resolution? I tried maximizing it's size on Google Sheets an...
I have a question for #5 from hw4-exercises. When we save the dictionary to a CSV file, should we manually create a blank CSV file first,...
- short post -
--- status update ---------
name: Korn T.
comments: 3
likes: 1
message: Anyone know how to force Google Drive to save the graph from HW5 at a higher resolution? I tried maximizing it's size on Google Sheets an...

- long post -
--- status update ---------
name: Keting C.
comments: 7
likes: 0
message: I have a question for #5 from hw4-exercises. When we save the dictionary to a CSV file, should we manually create a blank CSV file first,...

--- link ---------
name: Autumn G.
comments: 0
likes: 4
message: If you haven't already clicked the worst cat link that Sean put on our HW3 turn-in, click it now. Unless you hate laughing. If you hate l...
linked url: http://worstcats.tumblr.com/

--- photo ---------
name: Danny C.
comments: 1
likes: 0
message: For question #5 from hw4. Is it okay if I uncheck the comma separator to get the intended output?
thumbnail: https://fbcdn-sphotos-d-a.akamaihd.net/hphotos-ak-xpf1/v/t1.0-9/p130x130/1780890_10205088596832228_5356950455425450937_n.jpg?oh=be21923b931
4eddb44c9cfaa&oe=54ED8827&__gda__=1425119116_c9fbc56663add27a1d89b3ef9ba52b3a
photo: https://www.facebook.com/photo.php?fbid=10205088596832228&set=gm.1555081211392493&type=1&relevant_count=1
```

Test 2: Run gen_photo_page()

Uncomment the line that calls `gen_photo_page()`. Note that we have already defined `gen_photo_page()` for you. You shouldn't change its definition, but if you have problems, you may need to check your implementation of the methods you defined.

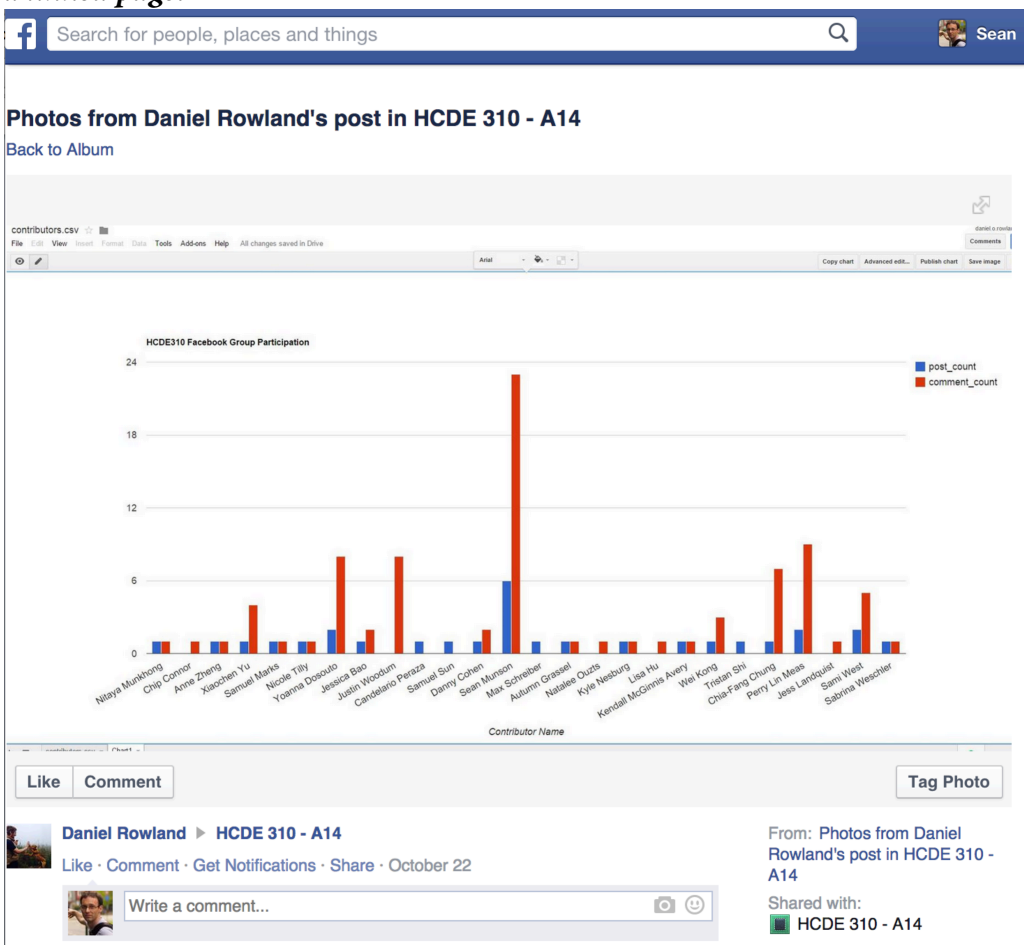
Then check out the file that is generated, hcde310photos.html, which is located in the same directory as hw6.py. You'll want to open it in a browser. One way to do that is to use the browser built into Eclipse. To do that, right click on your hw6 folder in Eclipse and refresh, to make the filename appear. Then right click on the file name, click "open with" and then "web browser". Try clicking on a thumbnail and you should be taken to the

Facebook page (requires logging in to Facebook). Make sure it resembles the following pages, and that the links work.

hcde310photos.html:

& so on

a linked page:



2. Tasks

2.1 Task 1: Finding questions (Required)

Create an `is_question()` method in `Post` that returns `True` if `self.message` contains a question mark, and False otherwise. In the main block, loop through the first 30 post objects, and use the `is_question()` method to only print out the message (a question) contained in the post. Your output should follow the format below:

```
------------
<name> got <num> responses to their question:
<short message>
------------
```

```
;;;;; Task 1: printing questions ;;;;;;;
---------
Yoanna D. got 2 responses to their question:
For P1 when they say data sources the project will use are they talking about the APIs?
---------
---------
Samuel M. got 3 responses to their question:
If I want to create a new file to try out exercises from class notes do I just create a new project, package, and class in the developmen...
---------
---------
Korn T. got 3 responses to their question:
Anyone know how to force Google Drive to save the graph from HW5 at a higher resolution? I tried maximizing it's size on Google Sheets an...
---------
---------
Daniel A. got 3 responses to their question:
Are we supposed to turn in a graph?
---------
---------
Samuel M. got 6 responses to their question:
Can anyone tell me if I'm on the right track? I was expecting this code to read in every line and assign names to keys in the dictionary,...
---------
---------
Danny C. got 1 responses to their question:
For question #5 from hw4. Is it okay if I uncheck the comma separator to get the intended output?
```

(and so on)

2.2 Just for Fun Task 2: Unique Commenters (Optional & Ungraded)

Add a method for the post class that generates the unique commenters for that post, other than the poster him/herself. That is, if three different people commented, print each of their names, but don't print anyone's name twice. Use that method within the `__str__()` method to print out those names in some readable way. The complete list of comments can be found in `self.comments`.

(Hint: use a `set` data structure. Look up the documentation for it. You may find the `.add()` method for sets useful. It's like append for lists, but if the item you add is already in the list, then it is not added again.)