Jasque Saydyk and Tyler Smith

Professor Vanderberg, Swenson, and Trice

CS 126L Section 2

22 February 2017

Lab 05 - Undeadbook

1. Problem Statement

The goal of this program was to make a series of functions that simulated a simple social media website, testing our ability to use fundamental data structures that come with Python. This program needed four functions to run.

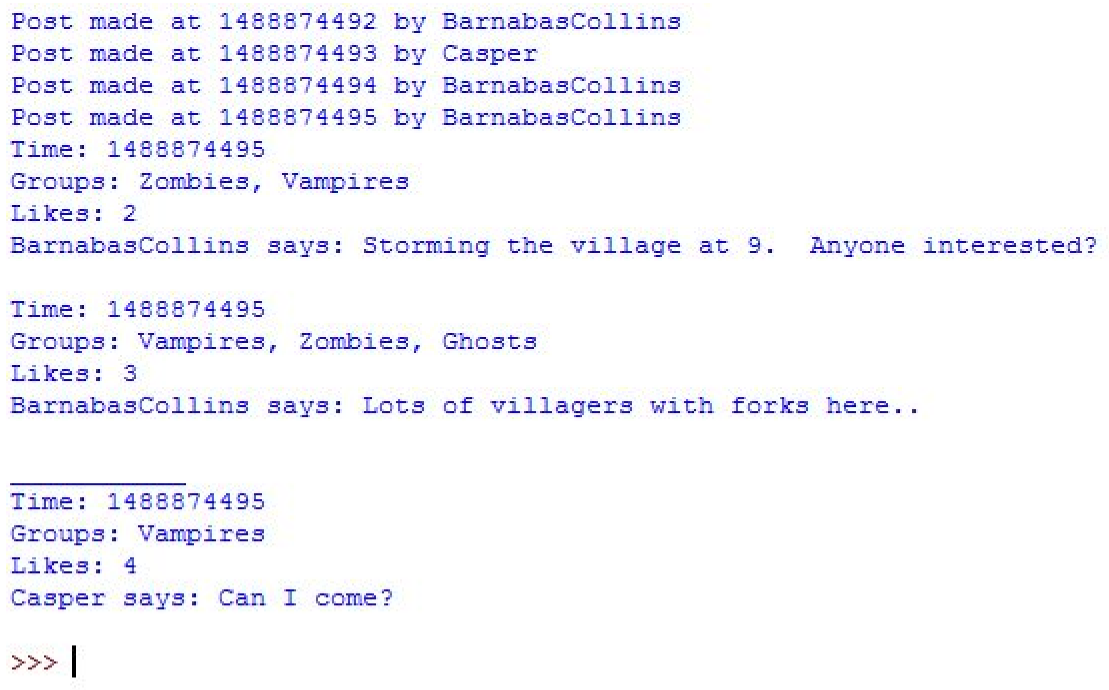
- Update function
    - Parameters
        - book: Data structure to add post to
        - status: The text of a status update
        - audience: A list of audiences that may be interested
        - name: The handle/name of the person posting
    - Return
        - A unique ID for the post that was added to book. Consists of username, then timestamp
    - Output
        - Unique id in following format “Post made at (time) by (name)”
- Like function
    - Parameters
        - book: Data structure to add post to
        - id: ID of the post to like
        - name: The handle/name of the person liking
- Unlike function
    - Parameters
        - book: Data structure to add post to
        - id: ID of the post to unlike
        - name: The handle/name of the person unliking
- Display function
    - Parameters
        - book: Data structure to read post from
        - id: ID of the post to be displayed
    - Output
        - The specific format specified in the lab

2. Planning

To do this lab, we first determined that we would have to type the given test out. After we had the test, we then determined that we would create each of the method headers and a docstring declaring what that method does, so that we get to filling out the methods, we aren't having to look back at the lab to do so. After we had the test and method headers with their docstrings filled out, we then determined that we would use a dictionary data structure for the book variable that the functions call for, and within that dictionary, there will be another dictionary for the Like and Unlike keys and a list for the Audience key within the book dictionary. We also determined we would have to use the time class to get the timestamp needed for the project. With all of this preparation and planning out of the way, we were then able to do the project.

3. Implementation and Testing

Due to the extensive amount of planning, the project was really straight-forward and simple. The only difficulty our team ran into was the idea of accessing a dictionary from within a dictionary was an unwieldy idea that took some getting used to. This project also forced us to brush up on the dictionary's documentation to understand its basic usage. The only other error we ran up against was having to convert any non-String output to String so that it could be properly printed by Python. All in all, this project was relatively simple project that we completed successfully. Below is the output from the program.



```
Post made at 1488874492 by BarnabasCollins
Post made at 1488874493 by Casper
Post made at 1488874494 by BarnabasCollins
Post made at 1488874495 by BarnabasCollins
Time: 1488874495
Groups: Zombies, Vampires
Likes: 2
BarnabasCollins says: Storming the village at 9.  Anyone interested?

Time: 1488874495
Groups: Vampires, Zombies, Ghosts
Likes: 3
BarnabasCollins says: Lots of villagers with forks here..

___________
Time: 1488874495
Groups: Vampires
Likes: 4
Casper says: Can I come?

>>> 
```

4. Reflection

In retrospect, how the project has you approach the problem is not how I would approach the problem. By having you layer dictionaries and lists, you get a confusing data structure whose scope will always have to remain limited, and any future programmer who has to understand this code will have a “fun” time doing so. A better approach to me would be a class based approach, where you have a post class with all of the data members in the dictionary be the data members of the class, then the functions would be the methods of the class. Along with some getters and setters, you have a simple, understandable, Object Oriented solution to this problem, that would have more features and be far more extensible than the current solution.