Undeadbook

1 Overview

A group of vampires, zombies, and ghosts have contracted you and your partner to create a social media site fiocused on the “undead” population. In this lab you will be creating a program that allows the undead (e.g., ghosts, zompies, and vampires) to make, display, like and unlike status updates.

2 Learning Outcomes

By the end of this project students should be able to:

- read and write programs that define simple functions;
- read and write programs using lists and dictionary syntax;
- read and write programs that use comparison operators;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students’ problem solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

- Read Problem Space Chapter 6: Collections.
- Read the full lab instructions and plan your strategy for representing the status updates using lists and dictionaries. Think about the requirements and how your data organization can satisfy the requirements.
- Do the following for your pre-lab and be prepared at the beginning of lab to show it to the lab aide.
  1. Describe your plan (i.e., what data you need to store, how you plan to store each part of the required data).
  2. Write pseudocode for three posts that represents your organization plan using Python list and dictionary syntax.

4 Lab Instructions

Do this part in lab:

You should implement several functions that Undeadbook will need. Keep in mind that "Output" describes what a function should print, while "Returns" describes the value a function should return. You will then be given a test program that uses the API you developed. The lab instructor will then check your final output for correctness.

update function .

The update function will take in a book variable that contains 0 or more posts that have been made. This function should update the book variable to add the status and the audience (both of which are passed into the update function) associated with the handle/name of the undead person making the post (passed in via the name parameter). Posts within the book should be accessible (and therefore stored) using a unique ID made up of the the name of the person that making the post and the time the post was made. For the time you will use the number of seconds since January 1, 1970. Your lab instructors will show you how to get this number.

Parameters

book data structure to add post to. It will be updated, and passed from one function to another.

User-defined: It will be your decision how this variable is structured.

status the text of a status update.

Example: "Storming the village at 9. Anyone interested?"

audience a list of audiences that may be interested.

Example: ["Zombies", "Vampires"]

name the handle/name of the person posting the update.

Example: "BarnabasCollins"

Returns

A unique ID for the post that was just added to the book. The unique ID should be based on the name of the undead person making the post and the timestamp representing when the post was made (e.g., BarnabasCollins1349708829)

Output

Before returing the unique ID, this function must print a confirmation of who made a post and at what time formatted like:

```
Post made at 1349708829 by BarnabasCollins
```

like function

The like function registers a like similar to Facebook. If someone likes a post more than once, subsequent likes should be silently ignored.

Parameters

book data structure containg posts, one of which is being liked via this function.

User-defined: Same structure as book variable in the update function.

id the id of the post to like. This is the unique ID based on the poster's name and timestamp (i.e., the information return from the update function). For example, BarnabasCollins1349708829.

name the handle/name of the undead person likeing the post.

Example: "Casper"

Returns

Nothing

unlike function

The unlike function registers an unlike similar to Facebook. If someone unlikes a post more than once, subsequent unlikes should be silently ignored. If the post has never been liked by the user, they can't unlike it and it should be silently ignored.

Parameters

book data structure to unlike post in

User-defined: Same structure as book variable in the update function.

id the unique ID of the post to unlike (e.g., BarnabasCollins1349708829)

name the handle of the person unliking the post.

Example: "Casper"

Returns

Nothing

display function

The display function takes in a unique ID and shows the corresponding post was made, the audience or groups of undead people that liked the post, the total number of likes for the post, the name of the undead person who made the post, and the status message.

Parameters

book data structure to read post from

User-defined: Same structure as book variable in the update function.

id the unique ID of the post to be displayed (e.g., BarnabasCollins1349708829)

Returns

Nothing

Output

This function should print a tweet formatted like this:

```
Time: 13479708829
Groups: Ghosts, Goblins
Likes: 1
BarnabasCollins (mention with @BarnabasCollins) says:
Ghosts are welcome too! LOL
```

5 Test Code

Below is the test code you should use in the main() function of your program.
Be sure to call the main function to start the program.

```
# Initialize your empty 'book' variable before running the code below.

# BarnabasCollins is adding the first post to the book variable. The posted
#    says'Storming the village at 9. Anyone interested?", and is intended
#    for the Zombie and Vampire audiences.
barnabas_one = update(book,
                      "Storming the village at 9.  Anyone interested?",
                      ["Zombies", "Vampires"],
                      "BarnabasCollins")
# some time goes by
time.sleep(1)

# Casper posts asking the Vampires if he can come along.
casper_one = update(book,
                    "Can I come?",
                    ["Vampires"],
                    "Casper")
time.sleep(1)

barnabas_two = update(book,
                      "Forgot to include the ghosts! LOL",
                      ["Ghosts"],
                      "BarnabasCollins")
time.sleep(1)

barnabas_three = update(book,
                        "Lots of villagers with forks here..",
                        ["Vampires", "Zombies", "Ghosts"],
                        "BarnabasCollins")

# Edmund and Grimm like Barnabas' first post
like(book, barnabas_one, "Edmund")
like(book, barnabas_one, "Grimm")
like(book, barnabas_one, "Edmund")

# 4 differnt undead people like Capser's first post
like(book, casper_one, "Edmund")
like(book, casper_one, "Grimm")
like(book, casper_one, "Harry")
like(book, casper_one, "Count␣Chocula")

# Casper likes Barnabas' second post
like(book, barnabas_two, "Casper")

# more likes..
like(book, barnabas_three, "Casper")
like(book, barnabas_three, "Count␣Chocula")
like(book, barnabas_three, "Boo")

# unlikes...
unlike(book, casper_one, "Edmund")
unlike(book, barnabas_three, "Casper")
unlike(book, casper_one, "Edmund")

# display some of the posts
display(book, barnabas_one)
display(book, barnabas_three)
print "___"
display(book, casper_one)
```

When you have completed the lab run pep8 against your code until all formatting errors have been corrected and your code is PEP 8 compliant. See the Getting Started lab if you need instructions on running the program, or the pep8 documentation found here.

6 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical.

Your lab report should begin with a preamble that contains:

- The lab assignment number and name
- Your name(s)
- The date
- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why.

3. Implementation and Testing

In the third section you should describe how you implemented your plan. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as a .py file)
- a screen shot of your running application / solution
- results from testing

4. Reflection

In the last section you should reflect on the project. Consider different things you could have done to make your solution better. This might include code organization improvements, design improvements, etc.

You should also ask yourself what were the key insights or features of your solution? Were there alternative approaches or techniques you could have employed? How would these alternatives have impacted a different solution?

5. Partner Rating

Every assignment you are required to rate your partner with a score -1, 0 or 1. This should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating of 1 indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 0 indcates that your partner met the class expectations of them. Rating your partner at -1 means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. If a student recieves three ratings of -1 they must attend a mandatory meeting with the instructor to dicuss the situation, and recieving additional -1 ratings beyond that, the student risks losing a letter grade, or even failing the course.

Colophon

This project was developed by Dr. James Dean Palmer of Northern Arizona University. Except as otherwise noted, the content of this document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.