Homework 10 - Object Oriented Programming

CS 1301 - Intro to Computing - Spring 2020

Important

- Due Date: **Monday, April 20th, 11:59 PM.**
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
    - TA Helpdesk
    - Email TA's or use class Piazza
    - How to Think Like a Computer Scientist
    - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Jasmine Chacko (jasminevchacko@gmail.com) & Damian Henry (dhenry35@gatech.edu)

Introduction

Python is one of many coding languages which uses object oriented programming (OOP). In OOP, classes can be created which contain certain attributes and methods which are shared by all objects of that class. This helps you create concise code which you can re-use. The goal of this homework is to understand OOP and its real world applications.

For this assignment, you will be building a Twitter application. There will be three classes working together: a Twitter class, an Account class, and a Tweet class. Each of these classes will have attributes and methods, as described below. You have been provided with a file that has the beginnings of these classes; you are responsible for filling in the rest.

Note: The given HW10.py has __repr__ methods, as well as some other useful methods, that we have already defined for you. Do **not** delete or change these, as these are needed for testing.

Tweet

Attributes:

- message (str): the body of the tweet
- hashtags (tup): the hashtags associated with the tweet
- likes (int): the number of likes the tweet has received
- retweets (int): the number of times the tweet has been retweeted

Methods:

- __init__
    - initializes the following attributes:
        - message (str)
        - hashtags (tuple) - if a tweet does not have hashtags, this attribute should be set to an empty tuple by default
        - likes (int) - every tweet must begin with 0 likes
        - retweets (int) - every tweet must begin with 0 retweets
- is_trending
    - This method determines whether or not a tweet is trending.
    - The method takes in a threshold of likes (int) and and a threshold of retweets (int). If the tweet's likes and retweets are both greater than the thresholds, then the tweet is trending.
- __str__
    - This method does not take in arguments and creates a string representation of a Tweet object.

  - The string representation should be formatted as follows:
    Message: <tweet message>, Hashtags: <tweet hashtags>
- __eq__ (provided)
  - This method checks if two tweets are equal based on their messages, hashtags, likes, and retweets.

Account

Attributes:

- username (str): the username of the account
- password (str): the password of the account
- tweets (list): contains Tweet objects that the user has tweeted

Methods:

- __init__
  - initializes the following attributes:
    - username (str)
    - password (str)
    - tweets (list) - every account must begin with no tweets (empty list)
- change_password
  - This method takes in an old password (str) and a new password (str).
  - First, the old password must match the current password in order for the password of the account to be changed.
    - If the passwords do not match, return "Passwords don't match"
  - If the new password is the same as the current password, return "New password must be different"
  - Otherwise, the account's password should be set to the new password passed to the method, and return "Successfully changed passwords"
- is_famous
  - This method checks to see if an account is famous based on its tweets.
  - If the account has more than 5 trending tweets, then this method should return True. A tweet is trending if it has more than 100,000 likes and more than 75,000 retweets.
  - Otherwise, the method should return False.
  - **Hint:** the is_trending method in the Tweet class is helpful to check if a Tweet is trending
- __lt__
  - This method checks to see if another account is "less than" another one.
  - An account is less than another if the sum of all the likes of the tweets of one user is less than the sum of all the likes of the tweets of the other user.
- __eq__
  - This method checks to see if two accounts are equal to each other.
  - Two accounts are equal if they have the same username and password.

Twitter

Attributes:

- accounts (list): contains Account objects that are registered
- followers (dict): the key is an account username and the value is a list of Account objects following the user
- following (dict): the key is an account username and the value is a list of Account objects the user is following
- trending (list): contains Tweet objects that are trending on the website

Note: the followers and following dictionaries work together! If an account follows another, both dictionaries should change.

Methods:

- __init__ (provided)
  - initializes the following attributes:
    - accounts (list) - A list of accounts on Twitter
    - followers (dict) - A dictionary in which the keys are usernames, and the value is a list of accounts following that user. The followers dict should be empty upon initialization.
    - following (dict) - A dictionary in which the keys are usernames, and the value is a list of accounts that user is following.
    - tweets (list) - A list of all the Tweet objects that have been tweeted
- register
  - This method should take in a username (str) and a password (str) of a user that wants to register a new account.
  - The method should first check to see if a username has already been registered. If the username is taken, the method should return "Username is already taken!"
  - If the username is not taken, the method should do the following:
    - Create the username as a key in the followers dictionary, initializing its value to an empty list.
    - Create the username as a key in the following dictionary, initializing its value to an empty list.
    - Create an Account object using the given username and password.
    - Add the new Account object to the accounts of Twitter.
    - Return the Account object.
- follow
  - This method should take in two Account objects (referred to as account1 and account2 here for ease).
  - Both accounts must already be registered! The accounts list will be useful.
  - account2 will be following account1, which means:

        - If account2 is not already following account1, account2 needs to be added to account1's list in the followers dictionary.
        - If account2 is not already following account1, account1 needs to be added to account2's list in the following dictionary.
        - In other words, both the followers and following dictionaries should reflect that account2 is now following account1.
    - If account2 is already following account1, this method should not do anything.
    - The order of parameters should be: self, account being followed, account that is following the other
    - **Note:** Keys are usernames, values are lists of Account objects
- unfollow
    - This method should take in two Account objects (referred to as account1 and account2 here for ease).
    - Both accounts must already be registered! The accounts list will be useful.
    - account2 will be unfollowing account1, which means:
        - If account2 is following account1, account2 should be removed from account1's list in the followers dictionary.
        - If account2 is following account1, account1 should be removed from account2's list in the following dictionary.
        - In other words, both the followers and following dictionaries should reflect that account2 has unfollowed account1.
    - If account2 is not following account1, then this method should not do anything.
    - The order of parameters should be: self, account being unfollowed, account that is unfollowing the other
    - **Note:** Keys are usernames, values are lists of Account objects
- delete_account
    - This method takes in an Account object.
    - If the account is registered, this method should remove the account from the list of accounts on Twitter.
    - The account should also be removed as a key from the followers and following dictionaries.
    - If the account is not registered, the method should do nothing.
- tweet
    - This method takes in an Account object, a message (str), and hashtags (tup).
    - If the account is not registered, the method should immediately return "You must have a registered account in order to tweet".
    - If the message length is greater than 25, the method should return "The length of the message must be <= 25".
    - Otherwise, the method should create a new Tweet object with the given message and hashtags.
    - The Tweet should be added to the account's list and the Twitter object's tweets list.

- get_famous
    - The method should return a list of the usernames of all the accounts that are famous.
    - **Hint:** The is_famous method within the Account class is helpful here.

Grading Rubric

Function	Points
Tweet: correct instantiation __init__	6
Tweet: correct implementation of is_trending	6
Tweet: correct implementation of __str__	2
Account: correct instantiation __init__	6
Account: correct implementation of change_password	7
Account: correct implementation of is_famous	7
Account: correct implementation of __lt__	6
Account: correct implementation of __eq__	6
Twitter: correct implementation of register	10
Twitter: correct implementation of follow	10
Twitter: correct implementation of unfollow	10
Twitter: correct implementation of delete_account	7
Twitter: correct implementation of tweet	10
Twitter: correct implementation of get_famous	7
Total	**100**

Provided

The `HW10.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW10.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can resubmit this assignment an unlimited number of times until the deadline; just click the "Resubmit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW10.py` on Canvas.