Project: Crime Time

CPE 101 - Fundamentals of Computer Science I
Fall 2019- Due 12/4 @11:55PM

Purpose:

To gain experience writing a class and instantiating objects in a full program**,** as well as using Python I/O functions.

Description:

For this assignment, you will write a program that reads and writes records to a file, which represents one of the most basic forms of persistent data storage.
You are provided two tab-separated value (TSV) files to be read:

1. crimes.tsv contains a one-line header and 155,889 crime descriptions
2. times.tsv contains the time and date information for the 155,889 crimes in crimes.tsv

- Header: **ID Category Description**
- Example: 150011660 ROBBERY "ROBBERY ON THE STREET, STRONGARM"

Note: You may ignore the Description field for this assignment.
For times.tsv contains the time and date information for the 155,889 crimes in crimes.tsv

- Header: **ID DayOfWeek Date Time**
- Example: 150011660 Monday 01/05/2015 02:40

Download the above files from PolyLearn.

Your program will make a new file called **robberies.tsv** with data combined from the provided files, linked together by ID. This file must include:

- Header: **ID Category DayOfWeek Month Hour**
- Example: 150011660 ROBBERY Monday January 2AM
-

To allow your program to produce somewhat meaningful results, all crimes processed will be of category ROBBERY only. All other categories should be filtered out when creating the list.

Implementation

Your program must store each line of data in an object whose type is a class called Crime. Write the definition for this class in your crimetime.py module.
The Crime class should have, at a minimum, the following attributes:

- id as read from crimes.tsv
- category as read from crimes.tsv
- day_of_week as read from times.tsv
- month modified from times.tsv as a full word, e.g. "January"
- hour modified from times.tsv in AM/PM format without minutes,e.g. "1PM"

The constructor for the Crime class need only take two arguments (besides self):

- an ID and
- a category.

Values for the other attributes will be assigned to each Crime object after they are instantiated. Thus, you should initialize these other attributes to None in the constructor.

Add the __eq__ method to your Crime class so that you can test two Crime objects for equality and inequality. Two Crime objects are equal if their IDs are the same.

Add a __repr__ method to the Crime class so that it has a meaningful string representation (i.e. not a memory address), with all attributes combined (using format) into a single string.

When you write a Crime object to a file, you can invoke this string with str (crime), where crime is a variable containing a Crime object. Be sure to use the tab character \t to separate attributes in the string and include the newline character \n at the end.

Finally, add a set_time method to the Crime class with the following parameters:

- day_of_week a string containing a day of the week as read from times.tsv
- month an integer for a month in the interval [1, 12] as read from times.tsv
- hour an integer for an hour in the interval [0, 24) as read from times.tsv

This method will be called when a Crime object needs to be updated with time data and should transform the month and hour integer arguments to their appropriate string representations (see above) before updating the object's attributes.

Minimum Required Program Structure

1) create_crimes(lines)

- Takes as input a list of strings, each representing a line from crimes.tsv.
- Returns a list of Crime objects, one for each unique ROBBERY found.
- There may be duplicate crimes (with the same ID) in the data; your program should only create one Crime object for each unique ID.

3) update_crimes(crimes, lines)

- Takes as input a list of Crime objects and a list of strings, each representing a line read from times.tsv.
- Locates Crime objects using find_crime and updates their time attributes.
- Returns a list of Crime objects with updated attributes.

4) find_crime(crimes, crime_id)

- Takes as input a list of Crime objects and a single integer crime ID.
- return the Crime object matching the given ID.
- It is recommended that you first implement the simpler but slower linear search to get the program working and later return to replace it with binary search.

Print the following (underscores indicate where data must be filled in by your program):
NUMBER OF PROCESSED ROBBERIES: __
DAY WITH MOST ROBBERIES: __
MONTH WITH MOST ROBBERIES: __
HOUR WITH MOST ROBBERIES: __

Testing:

You are required to write at least 3 tests for each function that returns a value (i.e. is not an I/O function). Since we are emphasizing test-driven development, you should write tests for each function first. In doing so, you will have a better understanding as to what the functions take as input and produce as output, which makes writing the function definitions easier.

In addition to module-level functions, you must also create at least 2 Crime objects in tests.py and update them with set_time. You must use assert statements to test these objects with the == and != operators, as well as each object's string representation.

You may also use the diff command to compare your robberies.tsv file against the provided expected-robberies.tsv file.
Make no assumptions about the order of the entries in crimes.tsv and times.tsv. Each project submission will be evaluated using shuffled versions of these files.

Submission
Submit your crimetime.py and tests.py files to the PolyLearn by 12/4 @11PM.