Fall, 2013 CIT 590

Programming Languages and Techniques
Arvind Bhusnurmath

Homework 4

September 20, 2013; Due September 27, 2013, *10:00 am*

This homework deals with the following topics

* TDD

* dictionaries

* using dictionaries to simulate a database

Problem .

Websites like IMDB (Internet Movie Data Base) maintain all the information about movies, the actors(for brevity, I'm going to use actors throughout the assignment to mean actors/actresses) etc. If you search for a movie on the website, a web page showing information about the movie is displayed. It also shows all the actors in it. If you click on an actor's name, you are taken to the actors web page and can find all the info about him/her. i.e. names of movies in which the actor has acted, some other info. This assignment should give you some insight into how the data might be organized in some of these websites.

We will use dictionaries to represent a correspondance between movies and the actors that acted in the movie and try and simulate some of the standard database operations.

The first thing to do would be to create the database. This amounts to doing two things

a) Create a global dictionary variable called **movieInfo**. Obviously it is intially empty. The dictionary has movie names as keys and given that key will provide you with a list of actors

To help you populate the database, we have provided two files as part of the assignment - a) a simple text file that has movies and actor information in some format and b) a python program that takes this files and makes it into dictionary. You should run that program in order to get an initial starting point for this assignment.

Create some utility functions for handling the database. Do all of this as closely as possible in the TDD paradigm i.e. test first. Do not worry about error checking within the internal functions of the assignment. Worry about it only in the main I/O function.

- Write an input function

  def **insertIntoDb(movieTuple)** wherein a movieTuple is a special kind of tuple of the form ('movieName', list of actors).

  So here is a concrete example ('The Dark Knight', ['Christian Bale', 'Gary Oldman', 'Heath Ledger']). Given input like this, the goal of this function is to parse the tuple and then just insert it into a global dictionary. A dictionary will obviously not allow duplicate information, so consider how best to handle that.

  Also assume that 'The Dark Knight' and 'the Dark knight' are one and the same movie - that is we are (sensibly) not being sensitive about case. Be careful about the following set of inserts insertIntoDb('The Dark Knight', ['Christian Bale']) and then insertIntoDb('The Dark Knight', ['Michael Caine']). What I mean by that is that the insertion of Michael Caine should not remove the Christian Bale entry.

  This insert function outputs 1 if a successful insert was made and -1 otherwise.

- Write a delete function def **deleteFromDb(movie)**. Here a movie is just a string and if the movie is in your database (dictionary), that entry is deleted else the user is politely informed that it is not present.

  The delete function outputs 1 if a delete is successully executed and -1 otherwise

- Write a simple 'select' function

  def **selectMovieInfo(movie)**. Given a movie either return a) A message saying no information was present in the database or b) print out the list of actors present in the movie.

  This function returns a list of actors present in the movie.

- Write a more complex 'select' function

  def **selectActorInfo(actorName)**. Given an actor/actress either return a) A message saying no information was present in the database or b) print out the list of movies that you can find them in (as per the database obviously),

  This function either returns an empty list in the case when we have no information pertaining to the actor or it returns the list of movies that this actor has acted in.

Now using these utility functions, try and write functions that answer the following questions

Get a file for this

- Given two movies, find the actors that are common to both movies. That is do the intersection for the two movie casts.

  def **getCommonActors(movie1, movie2)** - returns list of common actors

- Given an actor's name, find all the actors with whom he/she has acted.

  def **getCoActors(actorName)** - returns list of all actors that the actor has ever worked with in any movie

- Find the movie with the largest ensemble cast.

  def **getLargestEnsembleCast()** - goes through the database and picks the movie with the largest cast. In cases where two movies are tied for the first place, it somewhat arbitrarily decides to use alphabetic sorting for the tie breaker. So, for instance if you had entered 3 actors from 'dark knight' into your database and 3 from 'star wars' and there was nothing else in the db, then dark knight would be returned

- Find out the movies in which both actors are present def **getCommonMovie(actor1, actor2)** - goes through the database and returns the movies where both actors were cast. In cases where the two actors have never worked together, it returns just the empty list.

Finally to get this to be somewhat fun and interactive write a main user interaction function that repeatedly provides the user with the following options.

a Insert

b Delete

c Select movie information

d Select actor information

e Find common actors for a given pair of movies

f Get all co-actors of a given actor

g Get largest ensemble cast

h Get common movie

Depending upon the user's input choice, this main function asks for requisite information and then calls into the appropriate function. In the case of insert/delete, just print the word 'Done' after the insertion or deletion has been performed

- For the insert case - first ask for the movie's name. then repeatedly ask for actors' names until the user presses 'q'

- For the delete case - ask for movie's name

- for the select cases - ask for movie's name or actor's name depending upon the situation

- for common actors - ask for the 2 movies' names for which we would like to get common actors
- for all co-actors - ask for the actor's name
- no input required for largest ensemble cast, that is purely a database read.
- for get common movie - ask for the 2 actors' names for which we want a common movie

You are expected to use test driven development as much as possible. So write tests for all the above functions,except for the input/output one. Also, write tests for any helper functions that you come up with as you are trying to write the big functions. .**Submit 2 files, movieDb.py and movieDbTests.py**