CSI Lab 13

Tuesday, April 15th

Video link for hints as you work on this lab: Lab 13 Hints

Goal:

- Continue your practice writing functions.
- Write some code that uses a dictionary.
- Write some code that uses a set.

Part 0: Let's Warm Up With Sets

To complete the following lab, we will need to be using code that implements a set. Recall that a set is a collection of objects of potentially different types *(like a list)*, where no two elements can be identical *(not like a list)*.

Open your book or the presentation slides from session 31 and answer the following questions:

[Q1] How do we create a new, empty set? (In other words, what is the set constructor?)

[Q2] What happens if you try to store duplicate elements in a set?

[Q3] What does the following line do?

```
for element in mySet
```

[Q4] How do we add something to a set?

[Q5] Open up IDLE, and type the following at the interpreter prompt:

```
>>> mySet = set()
>>> mySet2 = set()
```

Please type the following commands and write down the output:

Command	**Output**
print(mySet)	
mySet.add("bacon") print(mySet)	
mySet2.add("cheddar") print(mySet2)	
mySet.union(mySet2)	
mySet | mySet2	
mySet.add("cheddar") print(mySet)	
mySet.intersection(mySet2)	
mySet & mySet2	
mySet2.symmetric_difference(mySet)	
mySet2 ^ mySet	

[Q6] Please briefly explain what the symmetric_difference, union, and intersection set methods are doing.

Introduction

Websites like IMDB (which stands for Internet Movie DataBase) maintain all sorts of info about movies, the actors 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 the hypertext link for an actor, you are taken to the actor's 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 the working of such websites.

In order to complete this lab you need some basic movie/actor data. Rather than giving you a huge database like IMDB uses, you are provided with a relatively small data file - movies.txt The format of this file is very simple. Each line represents one actor and a partial listing of the movies he/she has acted in. The format of one line of this file is:

```
Name of Actor, Movie1, Movie2, Movie3. . .
```

BUT WAIT!!!

Read carefully about BOTH activities before you begin anything. This assignment will be made much easier if, in addition to the two required functions, you also write several helper functions. There are common pieces to these two activities. Rather than cut and paste and duplicating the code, you should provide common code in helper methods and let the two methods use this helper code. The last point or two of credit for this lab will be based on using good helper methods and not duplicating code.

Before you begin, please create a new Python file called lab13.py and write the traditional comments at the top (file, author(s), and description). In addition, copy the following function into lab13.py where you replace my name with your name(s):

```
#Function printName
#Inputs: None
#Outputs: None
#Description: Prints a name.

def printName():
    #TODO: Modify this function to print YOUR name(s)
    print("Name: Sarah Diesburg")
```

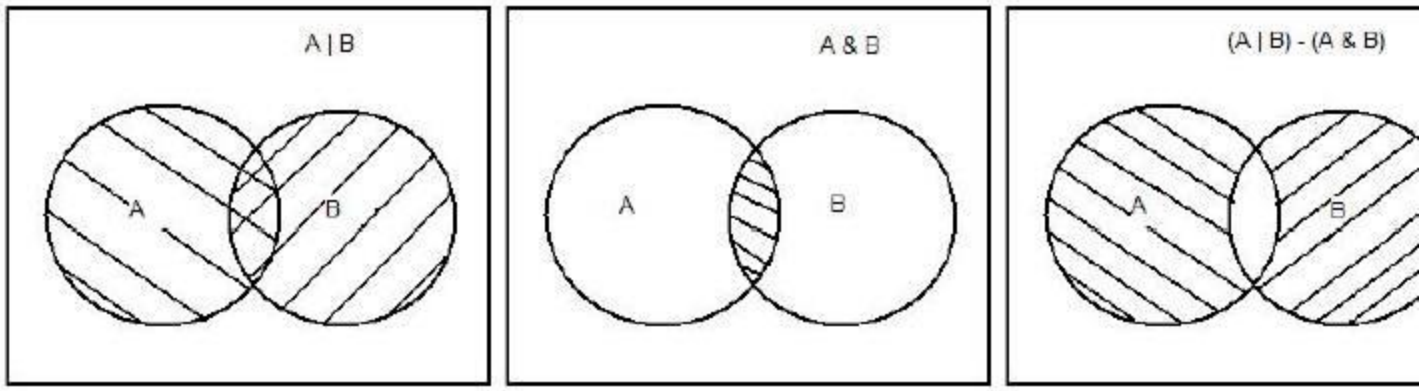
Activity A : Analyzing two movies

Your first task is to write a function called compareTwoMovies() that

- is given two string parameters - assumed to be two unique titles of movies in our database

- will print an error message and stop if either of the movies is not in our database
- will print a well formatted message about all of the actors in the two movies (A union B. Represented as A|B).
- will print a well formatted message about all of the common actors in the two movies (A intersection B. Represented as A&B)
- will print a well formatted message about all of the actors who are in one movie, but not both (A symmetric_difference B. Represented as A^B which is the same as (A|B) - (A&B)

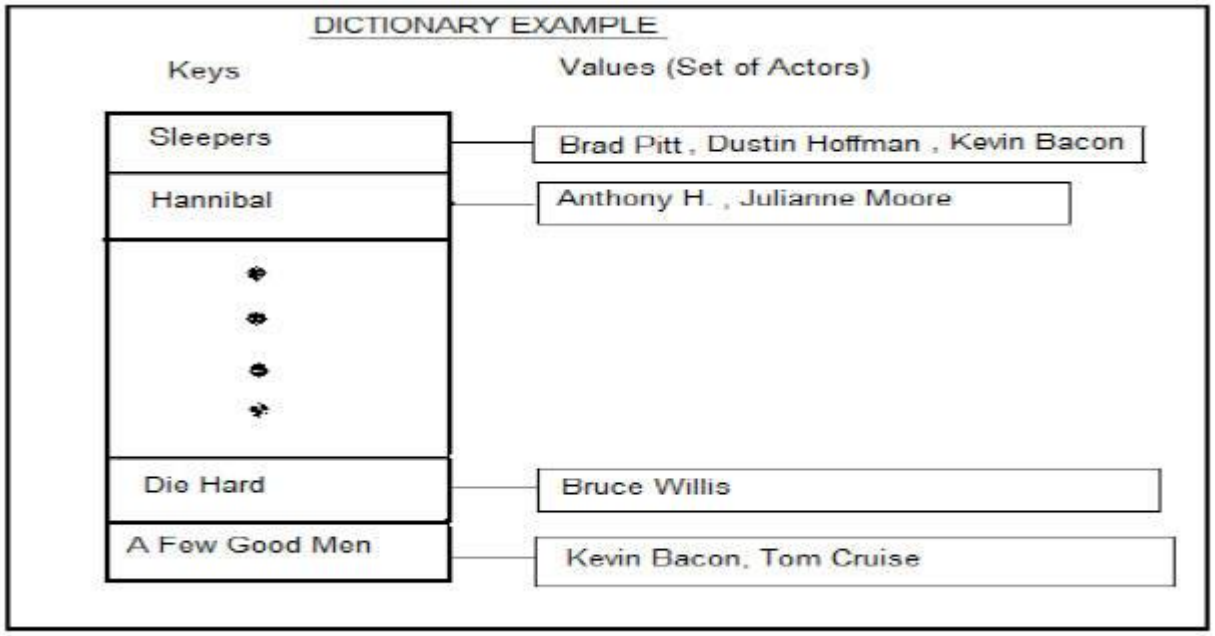
Recall from our discussions about Sets that the three main features described above can be illustrated as follows:



How do I do that???

What is an appropriate data structure for this assignment? Movie names are unique and our aim is to find the actors in the movies subject to some criteria (&, | or ^). In a dictionary, the keys are unique. So, that would make a good choice in this case.

Using the movies.txt file, create a dictionary with movie names as keys and a **SET** of actors as values. A list could also be used instead of a set here, but then you would have to implement the union, intersection and other set operations all by yourself (If you want, you can try that!).



Therefore, the High Level Algorithm for this method looks like this:

1. Read in the lines from the “movies.txt” file.
2. For each line:
    1. Separate out the actor and his movies.
    2. For each movie listed:
       a) If the movie name has not already been entered into the dictionary, add it as a key, and store the name of the actor as a value.
       b) If the movie name exists in the dictionary, add the actors name to the set of actors .i.e. to the value in the dictionary.

3. The dictionary is now ready.
4. Process the input to the function.
    1. Are both of the inputs keys in your dictionary?

        a) If not, then print an appropriate error message and quit.

        b) If so, perform the three "searches" and print their results.

While not required, I suggest that you write a small testing suite to test your code and demo this for one of the instructors before moving on. Here is a good test:

Use movies: Sleeper and Troy

Union of the two movies will be: Brad Pitt, Dustin Hoffman, Kevin Bacon, Diane Kruger

Intersection of the two movies will be: Brad Pitt

Symmetric Difference of the two movies will be: Dustin Hoffman, Diane Kruger, Kevin Bacon

---

Activity B : Finding one actor's co-actors

Your next task is to write a function called coActors() that

- is given a single string parameter - assumed to be the name of an actor in our database
- will print a well formatted message about all of that actor's co-actors. That is, print a message containing all of the actors who appeared in a movie with the first actor.
- if the original actor is not in our database than a well formatted message indicating this should be printed.

This activity starts the same way as Activity A - by building a dictionary/database of movies and actors. The challenge with this activity is discovering how to search that database for a particular actor and use the results of that search to solve this problem.

Again, while not required, I suggest that you write a small testing suite to test your code and demo this for one of the instructors before moving on.

Getting Credit for this assignment

This week I will again ask you to submit your code for electronic grading, using the eLearning submission system.

Follow the directions on the system to select the appropriate course and assignment and submit

- lab13.py

If you worked with a partner, make sure that both you and your partner's names are in the comment header at the top of the file ***and*** in the printName() function.