**Lab 8 - Sets and Functions**

**Overview**

This exercise requires you to use many things we have learned so far. For example, functions, exception handling, strings, tuples, dictionaries, sets and files. You will also need to implement good algorithmic reasoning. So try to design an algorithm first and then implement it in Python. If you manage to finish it properly and following good programming practices, it is a good sign you are becoming an efficient programmer =)

**Problem**

Websites like the Internet Movie Database ([www.imdb.com](http://www.imdb.com)) maintain extensive information about movies and actors. 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 the movie. If you click on the link for an actor, you are taken to an actor’s page, where you can find information about him or her, including the movies the actor has appeared in. This assignment should give you some insight into the working of such websites.

**Question**

**(a) Given two movies, each identified by a tuple (movie, year), and representing the set of actors in that movie:**

**i. Find all the actors in those movies: i.e., A union B (A | B).  
ii. Find the common actors in the two movies: i.e., A intersection B (A & B).  
iii. Find the actors who are in either of the movies but not both: symmetric difference (A - B).**

**Here is a small sample that you can save to a file and use to test your code.**

|  |
| --- |
| **Brad Pitt, Meet Joe Black (1998), Oceans Eleven (2001), Se7en (1995), Mr and Mrs Smith (2005)**  **Angelina Jolie, Bone Collector (1999), Lara Croft Tomb Raider (2001), Mr and Mrs Smith (2005) Denzel Washington, Bone Collector (1999), American Gangster (2007)**  **Julia Roberts, Pretty Woman (1990), Oceans Eleven (2001), Runaway Bride (1999**  **Morgan Freeman, Bone Collector (1999), Se7en (1995), Million Dollar Baby (2004)** |

**Each line starts with the name of an actor, followed by the movies he/she has been in. What is an appropriate data structure to store this information? A dictionary is suggested, as we want to access the movies and actors efficiently, but what should be the key? A key needs to be unique, which rules out actors’ names—they are unique in our sample but not in the whole database. On the other hand, movie titles and production dates form a unique identity that suggests an immutable tuple — perfect as keys. We can arrange our dictionary with (title, year) pairs as keys and have a collection of actors in each movie as the dictionary values. As we will be looking at the intersection and union of actor combinations, that suggests using sets for the collection of actors’ names in each movie. Read in the data and add the data to a dictionary that is structured as described.**

**Repeatedly prompt the user until some sentinel is entered. If two movies “name (year)” are entered, they should be separated by the appropriate operator: &, |,− to indicate the appropriate set operation to be performed (intersection, union, symmetric difference).**

**A larger set of movies is given in the file movie\_list.txt. Test your code with this file once you have managed to test with the smaller sample.**

**Dividing the problem**

**Some functions you might create to help you on dividing your problem:**

1. **get\_tuple function. It receives a string, for example “movie (year)” and returns a tuple (movie, year) that you can use as key in the dictionary**
2. **create\_dictionary function. It reads the text file, stores the file data in the dictionary as suggested, and returns the same dictionary to be used in the main scope;**
3. **menu function: prints the messages to the user and gets the necessary inputs until the user decides to quit.**

**Example input and output (you don’t need to follow the exact same formatting, just have the same results)**

**Open what file: movie\_list.txt**

**Please enter two movies as 'name (year)' separated by the appropriate operator (&, |, -), or enter "." to quit: Oceans Eleven (2001) & Se7en (1995)**

**The set of actors for this operation is:**

**{'Brad Pitt'}**

**Please enter two movies as 'name (year)' separated by the appropriate operator (&, |, -), or enter "." to quit: Oceans Eleven (2001) | Se7en (1995)**

**The set of actors for this operation is:**

**{'Brad Pitt', 'Morgan Freeman', 'Julia Roberts', 'Matt Damon', 'George Clooney'}**

**Please enter two movies as 'name (year)' separated by the appropriate operator (&, |, -), or enter "." to quit: Oceans Eleven (2001) - Se7en (1995)**

**The set of actors for this operation is:**

**{'Julia Roberts', 'Matt Damon', 'Morgan Freeman', 'George Clooney'}**

**Please enter two movies as 'name (year)' separated by the appropriate operator (&, |, -), or enter "." to quit: .**

**Process finished with exit code 0**

def get\_tuple(movie\_year: str) -> tuple:  
 *"""Gets a string in the format 'Movie (Year)' and returns a tuple (Movie, Year)"""* movie, year = movie\_year.split(" (")  
 year = year.strip(")")  
 return movie, year  
  
  
def create\_dictionary(file\_name: str) -> dict:  
 *"""Gets file name with 'Actor, Movie (Year), Movie (Year),...' per line format and  
 returns a dictionary {(Movie,Year):{actor, actor, ...}, (Movie, Year):{actor, actor}...}"""* file = open(file\_name, "r")  
 movies\_dictionary = {}  
  
 for line in file:  
 actor = ""  
 for i, elem in enumerate(line.split(", ")):  
  
 if i == 0:  
 # Keep actor saved to add to next movies  
 actor = elem  
 else:  
 try:  
 movies\_dictionary[get\_tuple(elem)].add(actor)  
 except KeyError:  
 # Movie doesn't exist yet in the dictionary  
 movies\_dictionary[get\_tuple(elem)] = {actor}  
  
 return movies\_dictionary  
  
  
def menu():  
 file\_name = input("Open what file: ")  
 movies\_dictionary = create\_dictionary(file\_name)  
  
 while True:  
 movies = input("Please enter two movies as 'name (year)' separated by"  
 "the appropriate operator (&, |, -), or enter '.' to quit:")  
  
 if movies == ".":  
 break  
  
 if "&" in movies:  
 (movie1, year1) = get\_tuple(movies.split(" & ")[0])  
 (movie2, year2) = get\_tuple(movies.split(" & ")[1])  
  
 try:  
 print(movies\_dictionary[(movie1, year1)].intersection(movies\_dictionary[(movie2, year2)]))  
 except KeyError:  
 print("Movies not in the dictionary")  
  
 if "|" in movies:  
 (movie1, year1) = get\_tuple(movies.split(" | ")[0])  
 (movie2, year2) = get\_tuple(movies.split(" | ")[1])  
  
 try:  
 print(movies\_dictionary[(movie1, year1)].union(movies\_dictionary[(movie2, year2)]))  
 except KeyError:  
 print("Movies not in the dictionary")  
  
 if "-" in movies:  
 (movie1, year1) = get\_tuple(movies.split(" - ")[0])  
 (movie2, year2) = get\_tuple(movies.split(" - ")[1])  
  
 try:  
 print(movies\_dictionary[(movie1, year1)].difference(movies\_dictionary[(movie2, year2)]))  
 except KeyError:  
 print("Movies not in the dictionary")  
  
  
menu()