**Lab 14 - Sets**

**Questions:**

**1.** If my\_set = set('bcd') and your\_set = set('abcde')

(a) What is the value of my\_set.issubset(your\_set)?  
(b) What is the value of your\_set.issubset(my\_set)?

**2.** Given a string S of lower case characters check whether this string is Heterogram or not. A heterogram is a word, phrase, or sentence in which no letter of the alphabet occurs more than once. Hint: remember that sets by definition do not have repeated elements. So try to somehow use sets to solve the problem. Check whether the strings below are Heterograms or not:

string1 = 'the big dwarf only jumps'  
string2 = 'given string is Heterogram'

**3.** Given a string S of lower case characters check whether this string is a Pangram or not. Pangram is a sentence using every letter of a given alphabet at least once. Check whether the string below is a Pangram or not:

string = 'The quick brown fox jumps over the lazy dog'

**4.** Given the following sets, predict the outputs and then check them in your shell. (Hint: Remember which is independent of the order—i.e., when it matters that a\_set is first versus b\_set being first.)

(a)  
a\_set = {"the", "coat", "had", "many", "colors", "red", "blue", "yellow"}  
b\_set = {"my", "coat", "had", "two", "main", "colors", "red", "blue"}  
x= a\_set.intersection(b\_set)  
y= b\_set.intersection(a\_set)  
print(x)  
{coat, had, colors, red, blue}  
print(y)  
{coat, had, colors, red, blue}

(b)  
w = a\_set.union(b\_set)  
v = a\_set.union(a\_set)

print(w)  
print(v)  
(c)  
t = a\_set.difference(b\_set)  
u = b\_set.difference(a\_set)  
print(t)  
print(u)

(d)  
r = a\_set.symmetric\_difference(b\_set)  
s = b\_set.symmetric\_difference(a\_set)  
print(r)  
print(s)

**5.** Write a function that takes a person’s first and last names as input and

(a) uses lists to return a list of the common letters in the first and last names (the  
intersection).

(b) uses sets to return a set that is the intersection of the characters in the first and last names.

(c) uses sets to return the set that is the symmetric difference between the first and last names.

**6.** Websites like the Internet Movie Database (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.

Here is what we’d like to do with the data:

(a) Given two titles of a movie as (movie, year), each 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 (also available on Brightspace) that you can work with for this exercise:

Brad Pitt, Meet Joe Black (1998), Oceans Eleven (2001), Se7en (1995), Mr and Mrs Smith (2005)

Tom Hanks, Sleepless in Seattle (1993), Catch Me If You Can (2002), You’ve got mail (1998)

Meg Ryan, You've got mail (1998), Sleepless in Seattle (1993), When Harry Met Sally (1989)

Anthony Hopkins, Hannibal (2001), The Edge (1997), Meet Joe Black (1998), Proof (2005) Alec Baldwin, The Edge (1997), Pearl Harbor (2001)

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)

Gwyneth Paltrow, Shakespeare in Love (1998), Bounce (2000), Proof (2005)

Russell Crowe, Gladiator (2000), Cinderella Man (2005), American Gangster (2007)

Leonardo Di Caprio, Titanic (1997), The Departed (2006), Catch Me If You Can (2002)

Tom Cruise, Mission Impossible (1996), Jerry Maguire (1996), A Few Good Men (1992)

George Clooney, Oceans Eleven (2001), Intolerable Cruelty (2003)

Matt Damon, Good Will Hunting (1997), The Departed (2006), Oceans Eleven (2001)

Ben Affleck, Bounce (2000), Good Will Hunting (1997), Pearl Harbor (2001)

Morgan Freeman, Bone Collector (1999), Se7en (1995), Million Dollar Baby (2004)

Julianne Moore, Assassins (1995), Hannibal (2001)

Salma Hayek, Desperado (1995), Wild Wild West (1999)

Will Smith, Wild Wild West (1999), Hitch (2005), Men in Black (1997)

Renee Zellweger, Me-Myself and Irene (2000), Jerry Maguire (1996), Cinderella Man (2005)

Each line starts with the name of an actor, followed by the movies he/she has been in. What is an appropriate data structure? 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 are entered, they should be separated by the appropriate operator: &, |,− to indicate the appropriate set operation to be performed (intersection, union, symmetric difference).

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 =)