Lab Exercise #11

Assignment Overview

This lab exercise provides practice with sets in Python.

You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

Part A: Basic Set Operations

Section 4. of the Python Library Reference Manual describes the available set operations:

http://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset

1. Examine the Python program below and predict the values which will be displayed.

```
S = set()

S.add( 900 )
S.add( 400 )
S.add( 700 )
S.add( 800 )

print( "len(S):", len(S) )        # len(S):

print( "S:", S )                  # S:

S.add( 900 )
print( "S:", S )                  # S:

S.discard( 400 )
print( "S:", S )                  # S:

item = 300
S.discard( item )
print( "S:", S )                  # S:

print( "len(S):", len(S) )        # len(S):

A = 700 in S
print( "A:", A )                  # A:

item = 200
B = item not in S
print( "B:", B )                  # B:
```

2. After completing (1) above, download and execute the program ("lab11.parta.py") to check your predictions. If any of your answers are incorrect, re-work the appropriate questions.

Part B: Set Operations and Methods

1. Examine the Python program below and predict the values which will be displayed.

```
S = { 10, 20, 30, 40 }
T = { 30, 40, 50 }
U = { 10, 30 }

A = S.union( T )
print( "A:", A )                # A:

B = S | T
print( "B:", B )                # B:

C = S.intersection( T )
print( "C:", C )                # C:

D = S & T
print( "D:", D )                # D:

E = S.symmetric_difference( T )
print( "E:", E )                # E:

F = S ^ T
print( "F:", F )                # F:

G = S.difference( T )
print( "G:", G )                # G:

H = S - T
print( "H:", H )                # H:

I = S.issuperset( S )
print( "I:", I )                # I:

J = S.issuperset( T )
print( "J:", J )                # J:

K = S.issuperset( U )
print( "K:", K )                # K:

L = S > U
print( "L:", L )                # L:

M = U.issubset( S )
print( "M:", M )                # M:

N = U < S
print( "N:", N )                # N:
```

2. After completing (1) above, download and execute the program ("lab11.partb.py") to check your predictions. If any of your answers are incorrect, re-work the appropriate questions.

Part C: Programming with Sets

Consider the file named "lab11.partc.py". That file contains the skeleton of a Python program to do a simple analysis of two files: it will display the number of unique words which appear in the two files (the union of those two sets of words), as well as the number of unique words which are common to both files (the intersection of those two sets of words).

Case does not matter: the words "pumpkin", "Pumpkin" and "PUMPKIN" should be treated as the same word. Only unique words should be counted: if a word appears more than once in a file, it should only be counted once.

a. Replace the comments labeled "YOUR COMMENT" in function "build_word_set" with meaningful comments to describe the work being done in the next statement. Use more than one comment line, if necessary.

b. Revise function "compare_files" to accomplish the work described in the comments.

c. Test the revised program. There are two sample documents available: "document1.txt" (The Declaration of Independence) and "document2.txt" (The Gettysburg Address).

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named "lab11.partc.py") for grading via the CSE handin system.**

Part D: Programming with Dictionaries and Sets

Consider the file named "lab11.partd.py". That file contains the skeleton of a Python program to display information about the words in a document.

Function "main" is complete. It handles the interaction with the user and calls other functions to perform the appropriate tasks.

Function "print_word_index" is complete. It receives a dictionary, where each element is a word and a set of line numbers where that word appears in a document. It displays all of the words (in alphabetic order), along with the lines numbers for each word (in ascending order).

Function "build_word_index" is incomplete. It receives an input file and builds a dictionary containing the unique words which appear in the input file, along with the line numbers where each word appears. The first line of the input file should be considered to be line 1.

a. Revise function "build_word_index" to accomplish the specified work. You may wish to review function "build_word_set" (above) for ideas about how to handle upper and lower case letters, as well as punctuation.

b. Test the revised program using the sample documents.

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named "lab11.partd.py") for grading via the CSE handin system.**