Lab Exercise #10

Assignment Overview

This lab exercise provides practice with dictionaries (maps) 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 Dictionary Operations

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

```
M = { 200: "EE", 100: "ME", 500: "CPS" }

A = 100 in M
print( "A:", A )                # A:

B = "ME" in M
print( "B:", B )                # B:


print( "M[100]:", M[100] )      # M[100]:

M[500] = "CS"
print( "M[500]:", M[500] )      # M[500]:

M[400] = "AE"
print( "M[400]:", M[400] )      # M[400]:


print( "M.keys():" )            # M.keys():
for key in M.keys():
    print( key )

print( "M.values():" )          # M.values():
for value in M.values():
    print( value )

print( "M.items():" )           # M.items():
for key, value in M.items():
    print( key, value )

print( "M:" )                   # M:
for X in M:
    print( X )
```

2. After completing (1) above, download and execute the program (“lab10.parta.py”) to check your predictions. If any of your answers are incorrect, re-work the appropriate questions.

Part B: Dictionary Operations and Methods

Section 4.10 of the Python Library Reference Manual describes the available dictionary operations:

http://docs.python.org/3/library/stdtypes.html#mapping-types-dict

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

```
M = dict()
M[ "Joyce" ] = 7
M[ "Mike" ] = 12
M[ "Bea" ] = 9

print( "M:", M )                # M:

M[ "Mike" ] = 33
M[ "Bea" ] = M[ "Bea" ] * 5

print( "M:", M )                # M:

if "Bea" in M:
    A = M[ "Bea" ]
    print( "A:", A )            # A:

if "Will" in M:
    B = M[ "Will" ]
    print( "B:", B )            # B:

C = M.get( "Bea" )
print( "C:", C )                # C:

D = M.get( "Will" )
print( "D:", D )                # D:

if "Joyce" in M:
    del M[ "Joyce" ]

if "Will" in M:
    del M[ "Will" ]

print( "M:", M )                # M:

E = M.pop( "Mike", None )
print( "E:", E )                # E:

F = M.pop( "Will", None )
print( "F:", F )                # F:

print( "M:", M )                # M:
```

2. After completing (1) above, download and execute the program (“lab10.partb.py”) to check your predictions. If any of your answers are incorrect, re-work the appropriate questions.

Part C: Programming with Dictionaries

Consider the file named “lab10.partc.py”. That file contains the skeleton of a Python program to do a simple analysis of a text file: it will display the number of unique words which appear in the file, along with the number of times each word appears. Case does not matter: the words “pumpkin”, “Pumpkin” and “PUMPKIN” should be treated as the same word.

Execute the program (which currently uses “document1.txt” as the data file) and inspect the output.

a. Replace each of the lines labeled “YOUR COMMENT” with meaningful comments to describe the work being done in the next block of statements. Use more than one comment line, if necessary.

b. Add doc strings to each function to describe the work being done in the function.

c. The program currently processes the empty string as a word. Revise the program to exclude empty strings from the collection of words.

d. The program currently processes words such as “The” and “the” as different words. Revise the program to ignore case when processing words.

e. The program currently always uses “document1.txt” as the input file. Revise the program to prompt the user for the name of the input file.

f. The program displays the words sorted by frequency of occurrence. Revise the program to also display the words sorted alphabetically.

g. Revise the program to display the collection of words sorted by greatest frequency of occurrence to least frequency, and sorted alphabetically for words with the same frequency count. Hint: since the “sorted” function and the “sort” method are stable sorts, you can first sort the words alphabetically, then sort them by reverse frequency.

h. 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 “lab10.partc.py”) for grading via the CSE handin system.**