Lab Exercise #8

Assignment Overview

This lab exercise provides practice with lists and functions 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: List Operations

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

```
A = 3 * [25]
print( "A:", A )            # A: _____________________________________

B = 3 * list( "25" )
print( "B:", B )            # B: _____________________________________

C = 3 * list( range(3) )
print( "C:", C )            # C: _____________________________________

D = ["a"] + ["bc"] * 2
print( "D:", D )            # D: _____________________________________

E = ["A","BC"] * 2
print( "E:", E )            # E: _____________________________________

F = "a" in ["aardvark"]
print( "F:", F )            # F: _____________________________________

G = "a" in ["a","n","t"]
print( "G:", G )            # G: _____________________________________

Z = ["Arthur", "King", "of", "the", "Britons"]

H = Z[:4]
print( "H:", H )            # H: _____________________________________

I = Z[-3:]
print( "I:", I )            # I: _____________________________________

J = Z[-6:-2]
print( "J:", J )            # J: _____________________________________

K = Z[1][2]
print( "K:", K )            # K: _____________________________________
```

2. After completing (1) above, download and execute the program ("lab08.parta.py") to check your predictions. If any of your answers are incorrect, re-work the appropriate questions.

Part B: List Methods

Section 5.1 of the Python Tutorial describes the available list methods:

http://docs.python.org/3/tutorial/datastructures.html#more-on-lists

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

```
A = []
A.append( 45 )
for n in range(4):
    A.append( 10*n-3 )

print( "A:", A )            # A: _____

B = [ 10, 11, 12, 13 ]
B.insert( 2, 99 )
B.insert( 0, 88 )

print( "B:", B )            # B: _____

C = [ 10, 20, 30, 40, 20 ]
C.remove( 40 )
C.remove( 20 )

print( "C:", C )            # C: _____

D = [ 10, 20, 30, 40, 20 ]
E = D.pop( 2 )
F = D.pop()

print( "D:", D )            # D: _____

print( "E:", E )            # E: _____

print( "F:", F )            # F: _____

Z = [ 10, 20, 30, 40, 20 ]
G = Z.index( 40 )
H = Z.count( 20 )

print( "G:", G )            # G: _____

print( "H:", H )            # H: _____

I = [ 15, -7, 12, -4, 14 ]
I.sort()

print( "I:", I )            # I: _____
```

2. After completing (1) above, download and execute the program ("lab08.partb.py") to check your predictions. If any of your answers are incorrect, re-work the appropriate questions.

Part C: Split and Join

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

```
Z = " Hello, World! "

A = Z.split()
print( "A:", A )            # A: _____

B = Z.split( "," )
print( "B:", B )            # B: _____

C = Z.split( ",! " )
print( "C:", C )            # C: _____

Z = "\tLine 1\n\tLine 2\n"

D = Z.split()
print( "D:", D )            # D: _____

E = Z.split( "\n" )
print( "E:", E )            # E: _____

F = Z.strip().split( "\n" )
print( "F:", F )            # F: _____

Z = "Mississippi River"

G = Z.split( "i" )
print( "G:", G )            # G: _____

H = Z.split( "s" )
print( "H:", H )            # H: _____

I = Z.split( "si" )
print( "I:", I )            # I: _____

W = [ "One", "Two", "Three" ]

J = " ".join( W )
print( "J:", J )            # J: _____

K = ", ".join( W )
print( "K:", K )            # K: _____

Z = "One Two Three"

L = ", ".join( Z.split() )
print( "L:", L )            # L: _____
```

2. After completing (1) above, download and execute the program ("lab08.partc.py") to check your predictions. If any of your answers are incorrect, re-work the appropriate questions.

Part D: Programming with Lists and Tuples

Develop a Python program which will calculate and display information about exam scores for the students in a class, as described below.

The program will prompt the user to enter the name of an input file. If that file cannot be opened, the program will prompt the user to re-enter the name of the input file.

Each line of the input file will represent one student and will have the following format:

Name (string, maximum of 20 characters)
Exam #1 score (integer, range 0 to 100)
Exam #2 score (integer, range 0 to 100)

The student's name will be in the first 20 characters of the line. The two exam scores will be separated by one or more blanks. For example:

```
Hopper, Grace        100   98
Knuth, Donald           82    88
Goldberg, Adele       94   96
Kernighan, Brian     89  77
Liskov, Barbara          87 97
```

The program will assume that the input file contains no erroneous data.

The program will read the contents of the input file and store the data set in a list of tuples, where each tuple will represent one student and will contain the following information:

Name (str)
Exam #1 score (int)
Exam #2 score (int)
Exam average (float)

The type of each field within the tuple is listed: the student's name will be type `str`, the two exam scores will be type `int`, and the exam average will be type `float`.

After reading and storing the data set, the program will display the following:

A table representing the data set, sorted alphabetically by name
The average of all scores on Exam #1
The average of all scores on Exam #2

The table will be aligned in columns and will be appropriately formatted. All averages will be displayed with one fractional digit of accuracy.

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named "lab08.partd.py") for grading via the CSE handin system.**

Suggestions:

1. Use the sample data file ("`scores.txt`") for your initial development, but then create a longer data file which contains a wider variety of test cases.

2. Develop the program incrementally:

a) Start by assuming that the user enters a valid file name; add error-checking later.

b) Read and display each line of the input file to make sure you are reading the data set correctly.

c) Use slicing to get the student's name and the `split` string method to get the two exam scores. Display those items to prove this step is working correctly.

d) Convert the exam scores to type `int` and calculate the student's average. Display those items to prove this step is working correctly.

e) Create a tuple containing the four items for each student (name, exam scores, exam average). Display the tuples to prove this step is working correctly.

f) Append each tuple to a list. Display the list to prove this step is working correctly.

g) Use the `sort` list method to re-order the tuples in the list. Display the list to prove this step is working correctly.

h) Use a `for` statement to display the contents of the list as a table (with appropriate formatting).

i) Use a `for` statement to calculate the average of all scores on Exam #1, then display the results.

j) Add the logic to calculate the average of all scores on Exam #2, then display the results.

k) Add the logic to handle problems with the user's selection of the file name.