CS 224 Introduction to Python | Spring 2022

Closed Lab 5

University of Wisconsin – La Crosse | *March 4*

Functions (and other assorted things)

Description: For this assignment, you will generate data (randomly) and sort it.

Details:

- Write a function that generates a list of experiment numbers. The list has length `num`. Each experiment number is an integer value in the range `min` to `min + num - 1`, where `min` and `num` are parameters to the function with default values of 0 and 50, respectively. The values should be in random order with no repeats or gaps. Think about how to do this before you begin coding – you can do much better than the brute force method. The function returns the list generated.
- Write a function that generates lists of data values. There are two types of data: integer-valued and real-valued. The function should take a parameter that indicates which type of data to generate (with a default value indicating integers). There should also be parameters for the number of values to generate, the minimum value, and the maximum value. These have default values of 50, 100, and 1000, respectively. The function should return the list generated.
- Using the functions you've written, write a function called `main` that generates a list of experiment numbers and three lists of data: one real-valued list and two integer-valued lists. All of the lists should contain the same number of elements. The experiment numbers and data lists are related by position. That is, the experiment number and data values at index i in each list represent the results for an experiment. Test your program.
- Write a function that takes the four lists as parameters as well as an optional parameter called `rev_order` with a default value of `False`. The function will sort the data by the real-valued data element, maintaining the relationship described in the previous sentence. In other words, after sorting the real-values will be in order and each real-valued datum will still be "grouped" with the same experiment number and integer data values as before the sort. For example, if the real-value in position i before the sort is in position j after the sort, then all of the other values that were in position i before the sort must also be in position j after the sort. `rev_order` indicates whether the data should be in non-decreasing order (`False`) or non-increasing order (`True`).

- The final result of your program is the four lists, sorted as indicated above. Print the data before and after sorting to verify that your program works correctly.