New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Introduction to Python
Homework Discussion, Session 6

6.1 Sorting: given the lines showing Open and Close stock prices in stock_prices.csv, attempt to discover the "biggest one-day gainers" for the year by sorting the file by the difference between Close and Open prices (i.e., Close minus Open, indicating how much higher the Closing price was over the Opening price) on a given line.

This problem may be simpler than you might imagine. We're just sorting a list of strings, and the only code required of you is to tell Python by what criteria each line should be sorted.

A "sort modification function" doesn't work with all of the elements or try to sort them. Rather, this function tells Python how a single element should be sorted. In brief, it does this by allowing Python to send an element to be sorted to the function, and the function then returns the value by which that element should be sorted. See the slide examples and also several of our exercises this week for details on this principle.

Therefore your by_closelessopen(line) function will expect a line from the file -- any line from the file -- and return a float value that is the Close stock price (as a float) minus the Open stock price (also as a float).

So what we are saying in effect is "sort this line (or any line) by the difference between its open and close prices"; this will result in Python's sort() function sorting the lines by their 1-day price movement from Open to Close.

So: the function will accept a string line, split the line into its fields, and do the calculation above by converting Open and Close into floats and subtracting Open from Close.

6.2 Read a config from a multidimensional structure and print out in a readable format.

Starting with this code:

```
from config import conf
```

conf is a list of dicts containing the config data. Loop through each dict and print out the info as indicated. Use spaces and empty print statements to organize elements as shown.

The purpose of this assignment is to help you disambiguate the nested braces and brackets that are found in a complex structure, and to use subscript syntax to access "inner" structures. You will also need to loop through the "inner" plugin lists.

This structure is a list of dicts, with one of the dict keys pointing to another "inner" dict, and another of the dict keys pointing to an "inner" list. All of the 3 dicts are uniform in structure, as is usually the case with config files -- they may be arbitrarily structured, but the structure is reliable.

As discussed in the slides and with some of the exercises, an easy way to explore these structures is to travel from "out" to "in" by accessing an element of the outer structure, and seeing what you have. So considering conf, notice that it's a list of dicts. If you loop through this list, each element will be one of the dicts. In a small-step fashion, try learning things about this dict. You can print the len() of the dict. You can print the keys of the dict. For example, here's my attempt at printing the keys of the dict:

```
for entry in conf:
    print(list(entry.keys()))
```

You'll see a list of the keys in each dict. It becomes clear that entry is each dict, and you should be able to print out the 'domain' value very easily.

The 'database' key points to another dict, however. Now you have to treat entry['database'] as a dictionary. You can call keys() on this dict as well:

```
list(entry['database'].keys())
```

So it should be relatively clear how to print the values in the "inner" dict.

The list associated with entry['plugins'] can also be treated as a list. You can loop through this list, take its len(); whatever you'd like to do with a list you can do it with entry['plugins'].

Again, some of the exercises cover multidimensional structures, so if anything is unclear make sure to do those as well.

good luck!