CSC108H/A08H Dictionary Lab

To earn your lab marks, you must actively participate in the lab. As usual, pick a partner and do pair programming (take turns driving and navigating). At the end of the lab, please show your work to your TA and return this handout. We will post the handout on the course website at the end of the week.

1 Objectives

- Understand dictionaries and how they store information.
- Use dictionaries to solve complex tasks.

For this lab, you will write down the inputs you will use to verify the functions and the expected outputs *before* you implement them! You won't get credit for your work unless you do.

2 Dictionaries

For the following exercises, you will be working with a data set that contains changes in the daily spot exchange rate of the US dollar to the British pound:

`http://www.stat.duke.edu/~mw/data-sets/ts_data/exchange-rates`

Each number indicates the amount that the exchange rate changed on a given day. You may use `urlopen` from module `urllib` to open this file directly from the web. Then, you can get data from the file just like in last week's lab.

When testing your code, it will be convenient to have a smaller data set. One is available on the course website. You should also create smaller data sets of your own. For example, a one-line file is a good starter test for `file_to_dict`.

Your tasks

1. Download the starter code, `dict.py`, from the labs page.
2. Read the docstring for function `dict_to_str`. Make sure that you understand what it is supposed to do in all circumstances.
3. Write down a good set of test cases for function `dict_to_str`: a set of argument values to the function, and the return value that you would expect to get in each case. A good set of test values will allow you to be confident that the function works for *any* possible argument it might be given.

   This is a hard problem, so think about it: there are a lot of possibilities! Make your tests as simple and specific as possible. Pick argument values to represent an entire category of possibilities, and look for inputs that cause your code to produce different behavior. Here's a starting point for your list of test cases: an empty dict, a dict with 1 entry, and a dict with more than 1 entry.
4. Write a main block that calls your function once for each test case. (No, you don't have the function yet!) Note that, for this particular function, you won't be able to use `assert` statements to check the result of each function call. Why not?
5. Now implement `dict_to_str`, and verify that the output is correct for each test case.
6. Write down a good set of test values for function `dict_to_str_sorted` and implement them in your main block.
7. Now implement `dict_to_str_sorted`, and verify your code to make sure it passes the tests.

8. Look at the sample data sets and design tests for function `file_to_dict`. First, make sure you understand the structure of the dictionary it produces. Try defining one with that structure "by hand" to make sure. Then implement and test the function.

9. Work through the rest of the functions, always designing your test values first.