L1 PROBLEM 3 (3/3 points)

In this problem and the next, we will explore how to make sense of a collection of data by first parsing the data, and then producing a meaningful plot that visually explains what the raw numbers say.

The file julyTemps.txt contains the daily maximum and minimum temperatures for each day in Boston for the 31 days of July 2012. Open the file in a text editor and look at how it is formatted. The goal of this exercise is to produce a plot that shows the difference between the high and the low temperature for each day.

First, we will write a function to load the data out of this file (for a review of file loading and processing, see Problem Set 1 - Review of IO. You may also wish to read up on the Python docs on file objects, particularly how to read multiple lines in a file).

We want this function to return a tuple of 2 lists: one for high temperatures and one for low temperatures.

The next set of questions will help guide you in writing a function that opens the datafile and parses the data into Python data structures.

1. Write the line of code that would load the file data into a variable called 'inFile'.

   inFile = open('julyTemps.txt', 'r')

2. Once the file is open, we'll parse the data line by line.

   After initializing two empty lists to hold the high and low temperatures, you'll want to loop through the lines of the file and split up each line into words (each line will be a string).

   Write the line of code which splits a line `line` into a list of elements by spaces and stores the result in a variable called `fields`.

   fields = line.split(' ')

3. In the loop, we need to make sure we ignore all lines that don't contain the relevant data. Be sure that you have looked through the raw data file and that you understand which lines do and do not contain relevant data.

   Which set of conditions would capture all non-data lines (ie, provide a filter that would catch anything that wasn't relevant data)? `fields` is defined as it was in Question 2 - that is, `fields` is the variable that contains a list of elements in a line.

   - ○ `if len(fields) != 3:`
   - [✓ `if len(fields) != 3 or 'Boston' == fields[0] or 'Day' == fields[0]:`] ✓
   - ○ `if '-' == fields[0] or 'Boston' == fields[0] or 'Day' == fields[0]:`
   - ○ `if '-' == fields[0] or 'Boston' == fields[0] or 'Day' == fields[0] or ' ' == fields[0]:`
   - ○ `if len(fields) < 3 or not fields[0].isdigit():`
   - ○ `if not fields[0].isdigit() or len(fields) < 3:`

Within the loop that you've written, append the relevant data to the lists you initialized earlier for high and low temperatures. Consider casting them to integers as you do this. Finally, your function should return a tuple of the two lists.

You will create a graph of the data in the next problem.

Spoiler: What Does the Function Look Like?

Check Show Answer

Show Discussion New Post