CSE 231 **Fall 2015**

Computer Project #6

Assignment Overview

This assignment focuses on the design, implementation and testing of Python programs to process data files using lists and tuples, as described below.

It is worth 50 points (5% of course grade) and must be completed no later than 11:59 PM on Monday, October 26.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj06a.py` – your source code program for Part A
`proj06b.py` – your source code program for Part B
`proj06c.py` – your source code program for Part C

Be sure to use the specified file names and to submit them for grading via the **handin system** before the project deadline.

Assignment Background

NASA (National Aeronautics and Space Administration) has a website with information about global climate change:

http://climate.nasa.gov/

That website has a discussion about changes in global surface temperatures over the past 130+ years and includes a summary of the data used in the analysis:

http://climate.nasa.gov/vital-signs/global-temperature/

In this assignment, you will develop three programs to help manage and visualize data on global temperature changes.

Assignment Specifications

1. You will develop the three Python programs described below. Each program will be in a separate source code file (the names are given under "Assignment Deliverables").

2. Your programs may not use sets or dictionaries.

3. Your programs may use any built-in or standard library functions.

Part A

1. The program in "`proj06a.py`" will read the contents of a user-selected data file and will call function "`draw_graph`" to plot a graph of the data from that file.

a) The program will prompt the user for the name of the input file. If it is unable to open that file, the program will display an appropriate message and halt.

b) The program will read the contents of that file and store the values in two lists, then call function "`draw_graph`" (which is supplied by the instructors).

2. The data file will contain zero or more lines, where each line contains a year (4 characters), a space, and a temperature deviation (4 characters). The file named "`data_avgs.txt`" is an example of a properly formatted data file; the first and last lines of that file are:

```
1880  -20
2014   75
```

Each value in the file is a deviation from a baseline temperature, in hundredths of a degree Celsius. That is, in 1880, the global mean temperature was 0.20 degrees Celsius cooler than the baseline value; in 2014, the global mean temperature was 0.75 degrees Celsius warmer than the baseline value.

3. Function "`draw_graph`" accepts two arguments: a list of X values (years) and a list of Y values (temperature deviations). That function uses those two lists to plot the change in global mean temperature over the series of years.

The instructors have supplied function "`draw_graph`" in the file named "`draw_graph.py`"; you will copy the lines from that file into your program in "`proj06a.py`" (they should be the first lines in your source code file, after the comments for the source code header).

Part B

1. The program in "`proj06b.py`" will produce data files which can serve as the input file for the program in Part A.

a) The program will always read from "`data_full.txt`" (it will not prompt the user for the name of the input file). If it is unable to open that file, the program will halt.

b) The program will prompt the user for the name of the output file. If that file does not exist, the program will create it and continue. If that file does exist, the program will discard the current contents of the file and continue.

c) The program will allow the user to select the subset of the data which is to be processed and written into the output file.

2. The file named "`data_full.txt`" contains monthly global mean temperature deviations. For each year, the file contains one line which identifies the year and has the global mean temperature deviation for each of the 12 months. The first four lines of that file are:

```
Year   Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
1880   -30  -20  -18  -28  -14  -29  -23   -7  -17  -16  -19  -21
1881    -9  -13    1   -3   -4  -29   -6   -2   -8  -19  -26  -15
1882    10    9    2  -20  -17  -25  -10    4   -1  -22  -21  -24
```

Each value in the file is a deviation from a baseline temperature, in hundredths of a degree Celsius. That is, in January of 1880, the global mean temperature was 0.30 degrees Celsius cooler than the baseline value.

3. The program will allow the user to select the subset of lines in "`data_full.txt`" which are to be processed. The program will prompt the user to enter a year, and it will then prompt the user to enter an integer count. The year will identify the first year that the user wants to include in the subset, and the integer count will identify the number of years that the user wants to include in the subset. For example, if the user enters "`1920`" and "`15`", the program will use the 15 years starting with 1920 and ending with 1934 as the subset.

If the user enters an integer count which is greater than the number of years available, the program will include as many years as possible in the subset. For example, if the user enters "`2010`" and "`90`", the program will use the 5 years starting with 2010 and ending with 2014 (the last year in the data set) as the subset.

If the user enters "`all`" (any mix of upper and lower case letters) as the year, the program will include the entire data set as the subset to be processed; it will not prompt the user to enter an integer count.

4. The program will calculate the average temperature deviation for each year in the selected subset: it will calculate the sum of the 12 monthly temperature deviations for that year, divide by 12, and round the resulting value to 0 decimal places.

An example: the sum of the monthly temperature deviations for 1881 is -133. That value divided by 12 is -11.083333333333334; rounded to 0 decimal places, the result is -11.

Another example: the sum of the monthly temperature deviations for 1882 is -115. That value divided by 12 is -9.583333333333334; rounded to 0 decimal places, the result is -10.

The program will then write the year and average temperature deviation to the output file using the format described in Part A: each line will contain a year (4 characters), a space, and a temperature deviation (4 characters).

5. The program will display appropriate messages to inform the user about any unusual circumstances.

Part C

1. The program in "`proj06c.py`" will read the file named "`data_full.txt`" and will display a list of the N warmest months in the data set, where N is an integer number selected by the user.

a) The program will always read from "`data_full.txt`" (it will not prompt the user for the name of the input file). If it is unable to open that file, the program will halt.

b) The program will read the contents of "`data_full.txt`" and will create a list of tuples, where each tuple contains a year, a month and a temperature deviation.

c) The program will prompt the user to enter the number of months (N) which should be included in the list of warmest months. If the user does not enter a positive integer value for N, the program will repeatedly display an appropriate message and prompt the user again (until the user enters a positive integer value).

2. The program will display a list of the N warmest months in the data set (based on the monthly temperature deviations). The list will be displayed in sorted order, from largest to smallest. Each item in the list will include the year, month name and temperature deviation. For example, the 5 warmest months in the data set are:

```
2007 Jan   97
2010 Mar   93
2002 Mar   91
2014 Sep   90
1998 Feb   89
```

The report will include a title and will be appropriately formatted.

3. The program will display appropriate messages to inform the user about any unusual circumstances.

Assignment Notes

1. The data in "`data_full.txt`" is from NASA's website and is used with permission.

2. Items 1-8 of the Coding Standard will be enforced for this project.

3. Function "`draw_graph`" in the file named "`draw_graph.py`" is complete as-is. You should copy the source code for that function into your file named "`proj06a.py`".

Suggested Procedure

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step may be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python statements, you must work on your own.

- Complete the program for Part A before working on the remainder of the project.

- Develop a simple version of the program for Part A, then run the program and track down any errors.

- Use the **handin system** to turn in the first version of your solution.

- Cycle through the steps to incrementally develop your program:

    - Edit your library program to add new capabilities.
    - Run the program and fix any errors.
    - Use the **handin system** to submit the current version of your solution.

- Use the **handin system** to submit the final version of your solution.

- Once Part A is complete, work on Part B in the same way: incrementally develop your program and use the **handin system** to submit your work. Then, work on Part C.

- You would be wise to back up your files on your H: drive, also.

- Be sure to log out when you leave the room, if you're working in a public lab.