CSCI 1100 — Computer Science 1 Homework 7
Dictionaries

Homework Overview

This homework is worth **110 points** toward your overall homework grade and is due **Thursday November 10, 2016 at 11:59:59 pm**. It consists of one single program called `hw7.py` that assumes the presence of a data file called `tempdata.json`.

The data set for this homework comes from NOAA (National Oceanic and Atmospheric Administration). It contains temperature data from Troy, NY from 1956 until June of 2015, measured at Hudson River Lock and Dam (you can see it for yourself here: `http://www.google.com/maps/place/42.75,-73.68333`).

The data set is given to you in a file called `tempdata.json` in JSON format, which can be read in its entirety with the following three lines of code:

```
import json

if __name__ == '__main__':
    data = json.loads(open("tempdata.json").read())
```

You might remember the JSON format from our Lecture 13 notes, `http://www.cs.rpi.edu/academics/courses/fall16/cs1/lecture_notes/lec13_files_web.html`. After making the above call, the variable `data` is a list of dictionaries. Each dictionary in the list contains climate values for a single month and year. For example:

```
>>> print data[0]
{u'MMNT': -9999, u'TPCP': -9999, u'MMXT': -9999, u'DX32': 0, u'MNTM': 0, u'DT90': 0,
 u'EMNT': 51, u'MXSD': -9999, u'DT32': 0, u'month': 7, u'DP05': -9999, u'year': 1956,
 u'DSNW': -9999, u'DP01': -9999, u'TSNW': -9999, u'DP10': -9999, u'DT00': 0, u'EMXT':
    -9999}
>>> print('{}/{}'.format(data[0]['month'], data[0]['year']))
7/1956
```

In order to understand this better you might write a simple `for` loop that goes through all the dictionaries in the `data` list and prints out the month and year of each.

Each attribute of each dictionary is a climate measurement for a given year and month. The `u` before each attribute name indicates that these are strings encoded in Unicode, for example `u'MMNT'`. Your code may or may not print out the `u`, but in either case, you may safely ignore it and treat these strings as any regular string.

Here is the list of attributes each dictionary has and their meaning:

```
month: Month of year (1-12)
year:  Year
EMNT: Extreme minimum daily temperature
EMXT: Extreme maximum daily temperature
MMNT: Monthly Mean minimum temperature
MMXT: Monthly Mean maximum temperature
MNTM: Monthly mean temperature
MXSD: Maximum snow depth
TPCP: Total precipitation
```

```
TSNW: Total snow fall
DT00: Number days with minimum temperature less than or equal to 0.0 F
DT32: Number days with minimum temperature less than or equal to 32.0 F
DT90: Number days with maximum temperature greater than or equal 90.0 F
DX32: Number days with maximum temperature less than or equal to 32.0 F
DP01: Number of days with greater than or equal to 0.1 inch of precipitation
DP05: Number of days with greater than or equal to 0.5 inch of precipitation
DP10: Number of days with greater than or equal to 1.0 inch of precipitation
DSNW: Number days with snow depth > 1 inch.
```

Note that some values are missing, especially for older data. In this case, you have a special value (`-9999`). You are expected to disregard any value that is `-9999`.

Problem Description

Write a program that reads in a year in the interval 1956 through 2015 inclusive (you may assume this input is correct) and finds the data for every month in this year to print out the following information:

- Top 3 months and associated values for:
  - highest maximum temperature (EMXT)
  - lowest minimum temperature (EMNT)
  - highest number of days with maximum temperature above 90 (DT90)
  - highest number of days with maximum temperature below 32 (DX32)
  - highest total precipitation (TPCP)
  - lowest total precipitation (TPCP)
  - highest snow depth (TSNW)
  - lowest snow depth (TSNW)

  All values should be printed as floats with one decimal point `{:.1f}`.

  If there are three or fewer valid values, then print that there is not enough data. (For example, snow depth in 1956.)

  Ties should be broken using the month. For maximum values such as highest maximum temperature, the tie-breaking month will be sorted in descending order from 12 down to 1 and for minimum values such as lowest minimum temperature the tie-breaking month will be sorted in ascending order from 1 up to 12. Note that if you store your value/month pairs as a list of tuples or a list of lists, this is the default sort behavior and you will not have to do any additional work to get the sort order correct. See the example for snow depth in 2014 at the end of this write up for an example of the expected ordering.

- Average of mean temperature across different months in the year (MNTM)
  - Overall average (average of all months for the chosen year for MNTM)
  - Average for the first 6 months of the year (MNTM)
  - Average for the last 6 months of the year (MNTM)

  If there are three or fewer valid data values for any one of these then print `Not enough data`.

- A histogram of overall average temperatures in 3 month intervals, i.e. months 1-3, 4-6, 7-9, and 10-12. If there are fewer than two months in an interval, then print `Not enough data`. A histogram prints a single `*` for each value in the average, excluding partial values. For example, 54 `*`'s will be output for an average temperature of 54.7.

Expected output

Below you can see the expected functioning of this program with the file we gave you (note: we might change the file in the homework submission server):

For the year 2014

```
Enter a year (1956-2015) => 2014
2014
Temperatures
------------------------------------------------------------------------
Highest max value => 7: 92.0, 6: 91.0, 9: 90.0
Lowest min value => 1: -9.0, 2: -7.0, 3: 1.0
Highest days with max >= 90 => 7: 3.0, 9: 2.0, 6: 1.0
Highest days with max <= 32 => 2: 20.0, 1: 18.0, 3: 9.0

Precipitation
------------------------------------------------------------------------
Highest total => 7: 5.8, 6: 5.6, 12: 5.0
Lowest total => 9: 0.9, 4: 1.9, 11: 2.1
Highest snow depth => 2: 24.6, 1: 12.6, 12: 5.8
Lowest snow depth => 5: 0.0, 6: 0.0, 7: 0.0

Average temperatures
------------------------------------------------------------------------
Overall: 48.3
First 6 months: 40.9
Last 6 months: 55.7

Temperature histograms
------------------------------------------------------------------------
01-03: **********************
04-06: ************************************************************
07-09: ********************************************************************************
10-12: ****************************************
```

For the year 1956

```
Enter a year (1956-2015) => 1956
1956
Temperatures
------------------------------------------------------------------------
Highest max value => 8: 92.0, 9: 88.0, 10: 83.0
Lowest min value => 12: 5.0, 11: 15.0, 10: 27.0
Highest days with max >= 90 => 8: 3.0, 12: 0.0, 11: 0.0
Highest days with max <= 32 => 12: 4.0, 11: 2.0, 10: 0.0

Precipitation
```

```
----------------------------------------------------------------------
Highest total => 9: 5.0, 11: 2.4, 12: 2.3
Lowest total => 8: 1.1, 10: 1.4, 12: 2.3
Highest snow depth => Not enough data
Lowest snow depth => Not enough data

Average temperatures
----------------------------------------------------------------------
Overall: 42.3
First 6 months: Not enough data
Last 6 months: 42.3

Temperature histograms
----------------------------------------------------------------------
01-03: Not enough data
04-06: Not enough data
07-09: ****************************************
10-12: ****************************************
```

Hints: (1) There are 70 `'-'s` in every dashed line. (2) Investigate use of the `rjust` string method to print things like the `01` month string.

Suggestions

It appears that there are many things to compute, but it is actually the same thing for many different keys of the same list of dictionaries. The trick is to figure out what functions are needed and how they can be used over and over again to do closely related things by changing the function call parameters. You might write functions to do a number of things:

- Gather values from `data` for a key and a year. (Note that `data` should be passed as a parameter.) Return these values in a list or a dictionary
- Compute various statistics from these gathered values.
- Compute and output average temperatures. Can this be called three times?
- Compute and output a histogram. Can this be called four times?
- Anything else? Some combination of the above?

If you are thoughtful and thorough in thinking through your use of functions you might find yourself only writing 100 lines of Python code or even fewer to solve this problem.