CSSE1001/7030

Semester 1, 2015
Assignment 1
10 marks

Due Thursday 16 April, 2015, 9:30am

1 Introduction

The Bureau of Meteorology (BOM) has many data sets that are freely available. For this assignment you will analyse and display maximum temperature data for a selection of weather stations around Australia. We have downloaded some data from

`http://www.bom.gov.au/climate/change/acorn-sat/index.shtml#tabs=Data-&-network`

to be used in this assignment. Because the data from different stations start at different dates and to reduce the size of the files we have cut out older dates.

We have grouped the stations we will be using in the assignment into two groups given in `stations.txt` and `stations2.txt`.

Within each group all the temperature data for each station starts and ends at the same date. For testing purposes, the groups are of different sizes and the start dates for each group are different.

Each line of each data file consists of a string representing the date and a temperature - for example the following comes from the file `Brisbane.txt`

```
19681229    30.7
19681230    29.4
19681231 99999.9
19690101 99999.9
19690102    27.1
```

The temperature `99999.9` signifies that no recording was made on that day. The date is written using the syntax `yyyymmdd` - i.e. `19681229` is the date

29/12/1968. The nice property about writing dates in this format is that dates are in string order if and only if they are in date order.

For this assignment you will write a program to carry out some simple calculations on the data and display the results. With your text-based interactive program you will be able to enter a command and display results as in the example below.

```
Welcome to BOM Data

Please enter the name of the Stations file: stations2.txt

Command: dm 20110101 20110105

Date        Adelaide       Melbourne      Sydney
20110101     28.6           23.1           29.0
20110102     26.1           21.6           28.0
20110103     26.2           20.9           21.3
20110104     29.6           22.0           22.6
20110105     28.7           21.9           23.4

Command: dd Adelaide Melbourne 20110101 20110105

Temperature differences between Adelaide and Melbourne

Date      Temperature Difference
20110101    5.5
20110102    4.5
20110103    5.3
20110104    7.6
20110105    6.8

Command: ya 2000 2006

Year        Adelaide       Melbourne      Sydney
2000         22.9           21.0           22.7
2001         22.4           20.6           23.1
2002         22.4           20.9           23.0
```

```
2003        22.5        20.7        22.6
2004        22.5        20.4        23.4
2005        22.8        21.4        23.4
2006        23.0        20.8        23.1

Command: bad
Unknown command: bad

Command: q
```

The program first asks for a stations file (that lists the stations of interest) and then it enters a loop prompting for a command.

The command

`dm` *start_date end_date*

displays the maximum temperatures for each station for the supplied start date up to and including the end date.

The command

`dd` *station1 station2 start_date end_date*

displays the differences of the maximum temperatures between the two stations for the given date range.

The command

`ya` *start_year end_year*

(**for CSSE7030 student only**) displays the yearly average maximum temperatures for the given year range.

The `q` command quits the program.

An `Unknown command:` message should appear if the command is not one of the commands discussed above or if the command has the wrong number of arguments. This is the only error handling that is required for this assignment. You may assume that all functions you will need to write will be tested only with 'sensible data'.

2 Assignment Tasks

For each function that you write you need to provide a suitable comment giving a description, the type and any preconditions. You should use the triple-quote commenting style. CSSE7030 students are required to add extra features as described at the end of this section.

2.1 Download files

The first task is to download `assign1.py`, `assign1_support.py` and the data files. The stations files contain lists of station names and for each station name there is a file of that name with a `.txt` extension that contains the data for that station.

We suggest you create a folder in which to write your solution and put these files in that folder.

The file `assign1.py` is for your assignment. **Do not modify the support file or your assignment file outside the area provided for you to write your code**. When you have completed your assignment you will submit the file `assign1.py` containing your solution to the assignment (see Section 4 for submission details).

2.2 Write the code

Finally, write your solution to the assignment making sure you have included suitable comments. There are several functions you need to write and these are described below. **Do not use global variables in your code.** We are not expecting you to deal with exceptions for this assignment and so we will not be testing your code with things like invalid date strings, dates out of order or invalid station names.

2.2.1 Load Dates

`load_dates(Stations)` takes a list of stations (as produced by the `load_stations` function) and returns the list of all the dates in the data set. Given that each stations data file has the same dates then this can be found be processing

any one of the data files. For example (this will produce a long list and so in the example below we have elided much of the list):

```
>>> stations = load_stations('stations.txt')
>>> load_dates(stations)
['19600101', '19600102', ..., '20141230', '20141231']
```

2.2.2 Load Station Data

`load_station_data(station)` takes a station name and returns the list of temperatures (as floats) in the order given. For example (this will produce a long list and so in the example below we have elided much of the list):

```
>>> stations = load_stations('stations.txt')
>>> load_station_data('Brisbane')
[29.2, 29.4, ..., 32.0, 32.0]
>>>
```

2.2.3 Load All Station Data

`load_all_stations_data(stations)` takes a list of stations and returns a list of the data for each station. So the result is a list of lists. Each inner list contains the data for a station. For example, if we execute:

```
>>> stations = load_stations('stations.txt')
>>> data = load_all_stations_data(stations)
```

then `data[0]` will contain the temperature data for Birdsville and `data[1][0]` will contain the first temperature in the Brisbane data.

2.2.4 Display Maximum Temperatures

`display_maxs(stations, dates, data, start_date, end_date)` displays a table of maximum temperatures for the given stations and the given date range. For example:

```
>>> stations = load_stations('stations2.txt')
```

```
>>> data = load_all_stations_data(stations)
>>> dates = load_dates(stations)
>>> display_maxs(stations, dates, data, '20021224','20021228')
Date        Adelaide       Melbourne      Sydney
20021224     24.4           20.0           22.6
20021225     26.2           19.4           24.1
20021226     ----           21.3           21.2
20021227     31.3           22.6           24.9
20021228     37.7           35.9           25.8
```

2.2.5 Temperature Differences

`temperature_diffs(data, dates, stations, station1, station2, start_date, end_date)`
returns a list of pairs of dates and temperatures between the temperatures for `station1` and `station2` on that date. The range of dates of interest is given by `start_date` and `end_date`. If either temperature is unknown (`99999.9`) then the difference should be unknown as well. For example (the layout below is used so that the text will appear on the page - it's different from what you should get in the interpreter. Also the floating point numbers may display differently):

```
>>> stations = load_stations('stations2.txt')
>>> data = load_all_stations_data(stations)
>>> dates = load_dates(stations)
>>> temperature_diffs(data, dates, stations, 'Adelaide', 'Melbourne',
            '20021224','20021228')
[('20021224', 4.4), ('20021225', 6.8), ('20021226', 99999.9),
 ('20021227', 8.7), ('20021228', 1.8)]
```

2.2.6 Display Temperature Differences

`display_diffs(diffs, station1, station2)` displays the temperature differences between `station1` and `station2`. `diffs` is the data produced by `temperature_diffs`. For example:

```
>>> stations = load_stations('stations2.txt')
>>> data = load_all_stations_data(stations)
```

```
>>> dates = load_dates(stations)
>>> diffs = temperature_diffs(data, dates, stations, 'Adelaide', 'Melbourne',
             '20021224','20021228')
>>> display_diffs(diffs, 'Adelaide', 'Melbourne')
Temperature differences between Adelaide and Melbourne

Date      Temperature Difference
20021224    4.4
20021225    6.8
20021226   ----
20021227    8.7
20021228    1.8
```

2.2.7 The Top-Level Interface

`interact()` is the top-level function that defines the text-base user interface as described in the introduction.

2.2.8 Hints

For string processing: `strip, split, partition`
For user interaction: `input`

2.3 Extra Task for CSSE7030 Students Only

The CSSE7030 students are required to add extra features to the program. (**Note:** CSSE1001 students will not gain credit for attempting the CSSE7030 task.)

The added features consists of an extra command in interact and two extra functions.

The extra command is of the form `ya` *start_year end_year*. An example is given in interaction in the Introduction.

The first extra function
`yearly_averages(dates, data, start_year, end_year)`
returns the pair of the list of years and the yearly averages for each station whose data is given in `data` for the given range of years. For example (you

will get a different looking result - the layout will be different and the floats will be more accurate):

```
>>> stations = load_stations('stations2.txt')
>>> data = load_all_stations_data(stations)
>>> dates = load_dates(stations)
>>> yearly_averages(dates, data, '2000', '2002')
(['2000', '2001', '2002'],
 [[22.9, 22.4, 22.4],
  [21.0, 20.6, 20.9],
  [22.7, 23.1, 23.0]])
```

The second function
`display_yearly_averages(years, averages, stations)`
displays the data produced by `yearly_averages`. For example:

```
>>> stations = load_stations('stations2.txt')
>>> data = load_all_stations_data(stations)
>>> dates = load_dates(stations)
>>> years, averages = yearly_averages(dates, data, '2000', '2002')
>>> display_yearly_averages(years, averages, stations)
Year         Adelaide       Melbourne      Sydney
2000          22.9           21.0           22.7
2001          22.4           20.6           23.1
2002          22.4           20.9           23.0
```

3 Assessment and Marking Criteria

In addition to providing a working solution to the assignment problem, the assessment will involve discussing your code submission with a tutor. This discussion will take place in the practical session you have signed up to in week 7. You **must** attend that session in order to obtain marks for the assignment.

In preparation for your discussion with a tutor you may wish to consider:

- any parts of the assignment that you found particularly difficult, and how you overcame them to arrive at a solution;

- whether you considered any alternative ways of implementing a given function;
- where you have known errors in your code, their cause and possible solutions (if known).

It is also important that you can explain to the tutor how each of the functions that you have written operates (for example, if you have used a for loop or a while loop in a function, why this was the right choice).

Marks will be awarded based on a combination of the correctness of your code and on your understanding of the code that you have written. A technically correct solution will not elicit a pass mark unless you can demonstrate that you understand its operation.

We will mark your assignment according to the following criteria.

Criteria	**Mark**
Your code is mostly complete, correct, clear, succinct and well commented. You are able to explain your code.	8 - 10
Your code has some problems OR you have some problems explaining your code.	4 - 7
Your code is clearly incomplete, incorrect, too complex or hard to understand OR you have major problems explaining your code.	1 - 3
Your work has little or no academic merit.	0

A partial solution will be marked. If your partial solution causes problems in the Python interpreter please comment out that code and we will mark that.

Please read the section in the course profile about plagiarism.

4 Assignment Submission

You must submit your completed assignment electronically through Blackboard.

Please read
`http://www.library.uq.edu.au/ask-it/blackboard-assessment`
for information on submitting through Blackboard.

You should electronically submit your copy of the file `assign1.py` (use this name - all lower case).

You may submit your assignment multiple times before the deadline - only the last submission will be marked.

Late submission of the assignment will not be accepted. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on-time, you may submit a request for an extension.

Requests for extensions should be made as soon as possible, and preferably before the assignment due date. All requests for extension must be submitted on the UQ Application for Extension of Progressive Assessment form:
`http://www.uq.edu.au/myadvisor/forms/exams/progressive-assessment-extension.pdf`
no later than 48 hours prior to the submission deadline. The application and supporting documentation (e.g. medical certificate) must be submitted to the ITEE Coursework Studies office (78-425) or by email to `enquiries@itee.uq.edu.au`. If submitted electronically, you must retain the original documentation for a minimum period of six months to provide as verification should you be requested to do so.