**Assignment 4**

**Value:**10%

**Due date:**04-Oct-2013

**Return date:**25-Oct-2013

**Submission method options**

Alternative submission method

**Task**

**Assignment 4**

This assignment is based on the Chapter Programming Exercise 10-8, but is not exactly the same. It also contains aspects from Chapter programming Exercise 7-10, but again not exactly the same

The Wombat Valley Tennis Club stores data about its members in a flat text file with one line representing the record for one member.  Here is the start of that file:

106:Nerk, Fred:fnerk@bigpond.com:0260557642  
110:Jones, Sally:sally\_jones@internode.com.au:0429765123  
113:Li, Christopher:chrisli92@gmail.com:0388776655  
.  
.  
and so on. The first field in each record is the ***Member number***, and the remaining three fields are ***Member name***, ***Email address*** and ***Phone No*** respectively.

If you refer to 'Resources' on the ITC106 Subject Interact Site you will find the full file there named **wvtc\_data.txt**. Copy it to your computer making NO CHANGES AT ALL to it.

Your task for this assignment is to design an algorithm and use it to write a Python program that will read the text file LINE BY LINE and place the data about each member into a dictionary with one **key-value pair**representing each member.  The **key** is to be the member number, and the **value** is to be a list containing the other three fields.  Once the data from the file has been successfully placed into the dictionary, the program displays a menu which looks like the following:

Wombat Valley Tennis Club  
     Member Register  
     ===============  
  
MAIN MENU  
  
<D>isplay member details  
<C>hange member details  
<A>dd a new member  
<R>emove a member  
<W>rite updated details to file  
<Q>uit  
  
Your choice (D, C, A, R, W or Q)?

For this assignment you are to implement ONLY the code needed to:

* read the file and place the data into the dictionary
* <D>isplay member details for a chosen member (user supplies the member number)
* <C>hange member details for a chosen member (but only the logic to alter that member's phone number)
* <W>rite updated details to file, and
* <Q>uit once all processing of the dictionary is complete.  (The <Q>uit option is to be the ONLY way the program may be exited.)

You may of course continue on to write all the rest of the code later, however there are no "brownie points" for doing so, and you are STRONGLY urged to get the required assessed sections of your program working perfectly BEFORE you spend time implementing the rest of the system.

**Notes:**

* Functions with appropriate parameter passing and value returning should be used throughout
* The <C>hange member details function will contain a sub-menu of the three possible changes that can be made to a member's details - i.e. a change to *name*,*email* or *phone number*. (The member number may not be changed since it's the key).
* Bomb-proofing (using loops and/or exceptions) must prevent the program from crashinghif abnormal events occur
* Even though a particular operation (such as the logic to allow an email address to be altered) is not being implemented here, the option must still be available to be chosen - it just does nothing if it's selected EXCEPT display a message saying "Option not yet available", with the program returning to the menu (or sub-menu) for another choice to be made. In this way even though the program is not fully complete it WILL show what is still to be implemented.
* Select a range of test data to demonstrate how your program handles "normal" and "abnormal" situations. Also ensure that you design a run to CLEARLY demonstrate that when the <W>rite updated details to file option is chosen, your program properly writes out the updated data from the dictionary into the data file in the SAME format as that shown in the original data file in 'Resources' . To prove this you must <Q>uit one test run directly after using the <W>riteoption to write out the updated details and then immediately run the program again to prove that the "new" data file is capable of being read in again without it crashing the program.  The data file should ALWAYS preserve the format of the original file - only the data items themselves may change.

Run your program using the test data you have designed and save the output it produces in a text file.

**Submit:**

1. Your algorithm and test data table.
2. Source code for your Python implementation.
3. Output test file demonstrating the results of using the test data.
4. Contents of the ***updated*** data file **wvtc\_data.txt**.

It is important that the output listings are not edited in any way.