***BE 1600***

*Section 901*

***Introduction to***

***Programming and Computation***

***Python***

**Project**

50 points

**Due 11/29/2020 (10:30 A.M.)**

The goal of this project is to:

Give you some experience with dictionaries

Practice with file I/O.

Practice with functions.

**Requirements:**

* In a word file:
  + In your own words, describe the problem including input and output.
  + In your own words, Describe the major steps for solving the problem.
* Using Python IDE software, implement your solution.
* Test the code and verify that the program works; include a sample output for each problem in the word file.

**Restrictions:**

You must work individually. All code must be the work of the individual. Do not share your code or copy from external resources.

**Submission**

Upload all required files (two .py files and one-word file) to the Canvas by the due time. DO NOT Email your files.

**Grading:**

For the main project question:

* each of the five required functions (8 points)
* the main part of the program (10 points)

For the extra credit:

* reading data from a file and store it in a python data structure (4 points)
* finding the states with the best and worst record for each of these indicators (8 points)
* printing the table to a file using the required format (8 points)

All assignments must be submitted by the Canvas. **No email or hard copy** is accepted. You must follow the following format:

1. For non-programming questions, use a word file to type your answers. Don’t use the text box on the Canvas to answer the questions or to write comments, we will not read it. State your answer clearly.
2. For programming questions, include only the source file of each programming problem.
3. Submit your files to the Canvas. You must submit your files on time; otherwise, you will receive zero.
4. Use “Add Another File” feature on Canvas to upload each additional file; do not upload the files as a compressed folder.
5. You can upload your files as many times as you like. Only the last attempt counts because the last files you uploaded are the only files your instructor will see.
6. There will be several modules on the Canvas. You need to upload your files using the correct module on the Canvas.
7. Name each file: *Assignment (assignment number)* for the word file [e.g. Assignment 02] and *Assignment (assignment number) \_ (Question number)* for each programming question [e.g. Assignment 02\_Q03].
8. To upload your file(s):

* In Course Navigation, click the ASSIGNMENTS module.
* Click the title of the assignment.
* Click the **Submit** Assignment button.
* Add **File**. ...
* Add Another **File**. ...
* **Submit** Assignment. ...
* View **Submission**.

*It is your responsibility to make sure that each file is uploaded correctly. If you uploaded a wrong file, you receive zero; files will not be accepted after due date even if you have a prove that the file is created before the due date.*

***Make sure you review the Cheating & Plagiarism policy on Canvas.***

**Name and Email Addresses (50 points)**

Write a program that keeps names and email addresses in a dictionary as key-value pairs.

The program should display a menu that lets the user look up a person’s email address, add a new name and email address, change an existing email address, and delete an existing name and email address. The program should save the data stored a dictionary to a file when the user exits the program. Each time the program starts, it should retrieve the data from the file and store it in a dictionary.

You must create and use at least 5 meaningful functions.

* A function to display a menu
* A function to look up a person’s email address
* A function to add a new name and email address
* A function to change an email address
* A function to delete a name and email address.

Your program should check user input; for example, if the user wants to delete a name that does not exist in the dictionary, you should print something like: the name is not in the database. A sample one run is shown below.

|  |
| --- |
| Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 2  Enter name: John  Enter email address: John@yahoo.com  That name already exists  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 2  Enter name: Jack  Enter email address: Jack@yahoo.com  Name and address have been added  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: Sam  The specified name was not found  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: Jack  Name: Jack  Email: Jack@yahoo.com  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 3  Enter name: John  Enter the new address: John@wayne.edu  Information updated  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: John  Name: John  Email: John@wayne.edu  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 4  Enter name: Sam  The specified name was not found  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 4  Enter name: Jack  Information deleted  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 1  Enter a name: Jack  The specified name was not found  Menu  ----------------------------------------  1. Look up an email address  2. Add a new name and email address  3. Change an existing email address  4. Delete a name and email address  5. Quit the program  Enter your choice: 5  Information saved  >>> |

**Extra Credit: Risk Factors (20 points)**

***Background***

The Centers for Disease Control (CDC) keeps data on what it calls “Winnable Battle Risk Factors and Health Indicators”. These are negative behaviors or incidents that could possibly be avoided by changes in lifestyle. In this problem you will examine some CDC data to discover the states with the best and worst records in regards to a few of these risk factors and health indicators.

***Problem Specifications***

The file “riskfactors.csv” lists data on 20 different risk factors and health indicators for each state. The data is in “comma separated value” format (csv), which means that each entry is separated from the others by a comma. Examining the file in a text editor or a spreadsheet program or both should help you understand the format. We grabbed this file from the CDC web page and have placed a copy in the project directory for you.

Since analyzing 20 different indicators can be a bit confusing, we will only look at five: *Heart Disease Death Rate*, *Motor Vehicle Death Rate*, *Teen Birth Rate*, *Adult Smoking*, and *Adult Obesity*. Your program will read in the data from the csv file and find the states with the best and worst record for each of these indicators (largest and smallest values). It will produce a file called “best\_and\_worst.txt” which lists the states that have the highest and lowest value for each of the indicators, along with their values.

Your program must be general enough to work with a similar file with states as rows and with the same column headers. That is, if the CDC puts out a new file with different values in the cells (e.g. maybe new research changed some values), your program will work correctly.

***Additional Requirements***

1. You must create and use at least 2 meaningful functions (your choice).
2. Your program must format the file into columns (see below). You don’t have to match our formatting exactly, but columns should line up and it should be readable. Use string formatting.

**Output of the file best\_and\_worst.txt :**

**Indicator : Min Max**

**---------------------------------------------------------------------------------------**

**Heart Disease Death Rate (2007): Minnesota 129.8 Mississippi 266.5**

**Motor Vehicle Death Rate (2009): District of Columbia 4.8 Wyoming 24.6**

**Teen Birth Rate (2009) : New Hampshire 16.4 Mississippi 64.2**

**Adult Smoking (2010) : Utah 9.1 West Virginia 26.8**

**Adult Obesity (2010) : Colorado 21.4 Mississippi 34.5**

***Hints***

1. Not all lines of the file contain state data. In particular, the first few lines are irrelevant.
2. Don’t forget to convert strings to numbers where appropriate.
3. Watch out for the fact that some data has a percent sign.
4. To open a file for output, remember:
   1. Open the file with the 'w' mode string.
   2. You can only write strings to a file, so you must convert each output to a string before you write them.
   3. Also, remember that if you want a separate line to occur in your output file, you must specifically output the carriage return/line feed string "\n".
5. Don’t forget to close your file—otherwise the string might not get written. If you find that your file has nothing in it or is missing information, forgetting to close the file is a likely reason.
6. Depending on how you design your program you may find some of the following list functions and methods useful: sort, min, and max. I used some, but not all of those.