Intro to Computer Science Programming Project - Address Book

ASSIGNMENT OVERVIEW

In this assignment, you'll be creating a program called `addressbook.py` which allows the user to manage a list of contact information. The data used by the main program will be stored in a separate file, `contacts.txt`. The main program will allow the user to enter, edit, view, and delete the records in the virtual address book, which should include a person's names, email address, phone numbers, etc.

This assignment is worth 75 points and is due on the crashwhite.polytechnic.org server is due on the crashwhite.polytechnic.org server at 23:59:59 on the date given in class.

BACKGROUND

One strategy for keeping track of data, even when a program isn't running, is by storing it in a database, an external file. The simplest type of database is a "flat-file" database, in which information is stored as strings of text.

In the example `contacts.txt` file shown below, each of the three lines is a "record" of one person, and each section of a line (separated by a "delimiter", a tab in this case) stores a data "field": names, email address, and phone number. The information in this text file will be loaded in to the program `addressbook.py` when that program is run, and a menu system will manipulate contact information as the user desires.

```
Richard White     rwhite@polytechnic.org        626-396-6688
Jill Bush         jbush@polytecnic.org          626-867-5309
Craig Fletcher    cfletcher@polytechnic.org     626-555-1212
```

PROGRAM SPECIFICATION

Create a Python program that has functions to:

a. add an entry into a "list of lists" called `contacts` (`add_contact` function)
b. write the contacts data to the `contacts.txt` data file before quitting the program
c. view all the information in contacts (`view_all_contacts` function)
d. open the `contacts.txt` data file and place them into the `contacts` list (the `initialize` function)
e. let the user to search for a name in `contacts`, and then prints out that person's information (`find_contact` function)
f. let the user delete an entry (`delete_contact` function)
g. let the user edit the information for a specific contact (`edit_contact` function)

Each line of `contacts.txt` will contain the record for a single person, with data fields on that line separated by the delimiter "`\t`" (Tab), as shown in the initial file above.

The program will track at least three of the following pieces of information for an entry:

a. first name
b. last name
c. email address
d. cell phone
e. street address
f. birthdate

DELIVERABLES

`addressbook.zip`

A zipped file (`addressbook.zip`, see below) that contains a directory (folder) called `addressbook`, that in turn contains your `addressbook.py` and `contacts.txt` files. To submit your assignment for grading, upload the zipped file to your directory in `/home/studentID/forInstructor` at `crashwhite.polytechnic.org` before the deadline.

Please be sure to include records for the specified names (Bush, White, Fletcher) from the example above in your database. You should also have at least one or two additional names in your `contacts.txt` file.

ASSIGNMENT NOTES

- This program will actually use two representations of our data to operate. When the program is not running, Contact information will be stored in a text format in a file called `contacts.txt`. There will be one line per record, with tabs separating the fields in that record. Using the sample contacts from above, `contacts.txt` stores data like this:

  ```
  Richard White\trwhite@polytechnic.org\t626-396-6688\nJill Bush\
  tjbush@polytechnic.org\t626-867-5309\nCraig Fletcher\
  tcfletcher@polytechnic.org\t626-555-1212
  ```

  You can see that there are Tab characters (`\t`) in there separating the fields, and newline characters (`\n`) separating each record.

- When the program is running, however, the Contact information is read into a "list of lists" called `myContacts`, where we can easily identify the information:

  ```
  myContacts[0] = ["Richard White","rwhite@polytechnic.org","626-396-6688"]
  myContacts[1] = ["Jill Bush","jbush@polytechnic.org","626-867-5309"]
  myContacts[2] = ...
  ```

  In this "list of lists," the email address for the first person on your contact list would be referred to as contacts[0][1].

- To run through all the **contacts**, then, we could write a loop in one of two ways:

  <u>**Strategy 1: Iterator**</u>

  ```
  for contact in myContacts:
      # do something w/ contact
  ```

  <u>**Strategy 2: index variable**</u>

  ```
  for i in range(len(myContacts)):
      # do something with myContacts[i]
  ```

  To run through all the **fields** in each contact, we would use nested loops in one of two ways:

  <u>**Strategy 1: Iterator**</u>

  ```
  for contact in myContacts:
      for field in contact:
          # do something w/ field
  ```

  <u>**Strategy 2: index variable**</u>

  ```
  for i in range(len(myContacts)):
      for j in range(myContacts[i])):
          # do something with myContacts[i][j]
  ```

- A pseudocode/template version of the program is given here, along with some snippets of code that we'll use to start our program. Please note that the `initialize()` function will not work as written unless there is already a `contacts.txt` file for it to open. Make sure that file exists before running the program the first time, or see the note at the end of this assignment for information on how to get around that restriction.

```
#!/usr/bin/env python3

"""addressbook.py
Program to collect information on contacts and
write it to a text file
@author Richard White
@version 2016-11-28
"""

import os

def initialize():
    # Specify handle for file and open for Reading
    infile = open("contacts.txt","r")

    # Initialize the contacts list to nothing
    contactsLines = []

    # Get one line at a time, and strip off newline
    for line in infile:
        contactsLines.append(line.rstrip("\n"))

    # Now we've got all the data, at least as lines
    infile.close()

    # Break lines up according to tab separators
    contacts = []
    for line in contactsLines:
        # split line up by tab separators and append it to our contacts list
        contacts.append(line.split("\t"))
    return contacts
```

(Continued on next page)

```
def add_contact(contacts):
    print("Adding a contact")
    name = input("Contact name (Last, First): ")
    email = input("Contact email: ")
    phone = input("Contact phone #: ")
    new_contact = [name,email,phone]              # Place fields in a list
    contacts.append(new_contact)                  # Append that list to contacts

def view_contacts(contacts):
    """This function is called by option 2 on the
    main menu, and allows the user to see all their
    contacts."""
    for record in contacts:
        for field in record:
            print(field)
        print("-----")

def find_contact(contacts):
    pass                                          # This needs to be written!

def delete_contact(contacts):
    pass                                          # This needs to be written!

def edit_contact(contacts):
    pass                                          # This needs to be written!

def write_file(contacts):
    # outfile is the "handle" that refers to the external file
    outfile = open("contacts.txt", "w")
    # "w" specifies that we're Writing to the file
    for record in contacts:                        # go through the contacts...
        # Write to outfile a line for each contact
        outfile.write("\t".join(record)+"\n")
    outfile.close()                                # close the external file

def menu():
    # os.system("clear")
    print("+--------------------+")
    print("|    ADDRESS BOOK     |")
    print("|                     |")
    print("| 1. Add contact      |")
    print("| 2. View contacts    |")
    print("| 6. Exit             |")
    print("+--------------------+")
    choice = input("Enter your choice: ")
    return choice

def main():
    myContacts = initialize()
    choice = ""
    while (choice !="6"):
        choice = menu()
        if choice == "1":
            add_contact(myContacts)
        elif choice == "2":
            view_contacts(myContacts)
    write_file(myContacts)
    print("Program closing")

if __name__ == "__main__":
    main()
```

You can use this code, along with code presented in class, as a framework for the program that you'll write.

- When it comes time to upload your files, you're going to be "zipping" your directory into a single file called `addressbook.zip`.
    - In OS X or Ubuntu, find the directory containing your `addressbook` folder, ctrl-click or right-click on the folder, and select "Compress" to create a zipped copy of that directory.
    - If you prefer working in the Terminal, you can use the zip command:

      ```
      $ zip -r addressbook.zip addressbook
      ```

    - In Windows, right-click on the `addressbook` folder and select ***Send to > Compressed (zipped) folder***, then name the new zipped folder `addressbook.zip`
- Once you've created the compressed file—the original directory is left intact—you can upload that file, `addressbook.zip`, to the Dropbox on the server.

GETTING STARTED

1. Create a directory, `addressbook`, that will be used to store all files related to your program.
2. Use a text editor to create your initial blank `contacts.txt` file. (See the *IMPROVEMENTS* section at the end of this document for information on how to do this in a slightly more elegant way.)
3. Use a text editor to begin programming the `addressbook.py` file with the overall structure of the program, including a separate function definition for each address book operation. Most of these functions will be empty for the moment; put a `pass` for each one so that it the program will still compile, even when there's not currently code for each of those function.
4. Write the function that will add information for an entry into the contacts list.
5. Write the function that will write the contacts list into the `contacts.txt` text file just before the program quits, so that any modifications to the list of contacts will be saved.
6. Once the initial three functions have been entered and tested, proceed to writing the other functions as needed.
7. Use one of the zip strategies specified above to compress all your work into a single archive file.
8. Use `scp` to upload your file `addressbook.zip` to the server.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. What distinguishes a flat-file database from a relational database?
2. What is a CSV file? Why did we not use a comma as a delimiter in this program?
3. Most people find a Contacts manager such as this one to be useful. What other digital database utilities do you think people find useful, or would find useful if they existed?

SAMPLE INTERACTIONS

```
$ python addressbook.py
Richard's Address Book!
You may: 1. Add a contact
         2. Lookup a contact
         3. Edit a contact
         4. Delete a contact
         5. View all contacts
         6. Exit
Your selection: 1
Adding a contact
Contact name (First Last): Susan Smith
Contact email: susan.smith@gmail.com
Contact phone #: na
Richard's Address Book!
You may: 1. Add a contact
         2. Lookup a contact
         3. Edit a contact
         4. Delete a contact
         5. View all contacts
         6. Exit
Your selection: 5
Richard White rwhite@polytechnic.org 626-396-6688
Jill Bush jbush@polytechnic.org 626-867-5309
Craig Fletcher cfletcher@polytechnic.org 626-ida-know
Susan Smith susan.smith@gmail.com na
Richard's Address Book!
You may: 1. Add a contact
         2. Lookup a contact
         3. Edit a contact
         4. Delete a contact
         5. View all contacts
         6. Exit
Your selection: 2
Looking up a contact...
Enter contact's name, email address, or phone number: Smith
['Susan Smith', 'susan.smith@gmail.com', 'na']
```

IMPROVEMENTS

In the course of building your program one of the first things that gets created is the `contacts.txt` file. Once that file exists, it's not too difficult to write a function that reads the file. But what if that file doesn't exist in the first place? If you give this program to someone else to run and they don't have a contacts.txt file already on the computer, the program will fail when it tries to read a non-existent file. How do we take care of that?

You can improve your program by using Python's error catching `try-except` statement as follows:

```
def initialize():
    try:
        infile = open("contacts.txt","r")
        contactsLines = []
        for line in infile:
            contactsLines.append(line.rstrip())
        infile.close
        contacts = []
        for line in contactsLines:
            contacts.append(line.split("\t"))
        return contacts
    except:
        contacts = []
        return contacts
```

The `try` block of the function tries to open `contacts.txt` for reading contact information. If that file doesn't exist though, as is the case when the program is run for the very first time, Python will halt execution of the program, and indicate that fact as an error. Here, we are trapping that error. The `except` block tells the program what to do if an error occurs during the `try` block. Here, we'll initialize the contacts list as empty, and return that empty list to the main program.