POLYTECHNIC COMPUTER SCIENCE - SPRING 2012
PROGRAMMING PROJECT #6
ADDRESS BOOK

ASSIGNMENT OVERVIEW

In this assignment, you'll be creating a program called *address_book.py*, which allows the user to manage a list of contact information. The data that you'll be working with is stored in a separate file, and will allow the user to enter, edit, and view a person's names, email address, phone numbers, etc.

This assignment is worth 50 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. Each line is a "record," and each section of a line (separated by a "delimiter") stores a data "field." The program, then, manipulates this data field according to commands from the user.

In this example, the program address_book.py will manipulate contact information stored in the text file *contacts.txt*, which initially has the following content:

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

PROGRAM SPECIFICATIONS

Create a Python program that has functions to:

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

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. home phone
e. cell phone
f. street address
g. birthdate

DELIVERABLES

A directory (folder) called *address_book-lastnameFirstinitial* that contains your *address_book-whiter.py* and *contacts.txt* files. The directory should have a name in the customary format, ie. *address_book-whiter.py* . Upload the file to the /home/rwhite/Public/Dropbox folder on the *crashwhite.polytechnic.org* server.

1. Please be sure to include the specified names (Bush, White, Fletcher), although you may have additional names in your *contacts.txt* file.
2. Save a copy of your file on your hard drive, flash drive, etc..
3. Your site will be graded based on files you upload to the server.

ASSIGNMENT NOTES

- A pseudocode version of the program is given here:

```
# def initalize():
#    pass
# def add_contact(contacts):
#   pass
# def view_contacts(contacts):
#    pass
# def lookup_contact(contacts):
#    pass
# def delete_contact(contacts):
#    pass
# def edit_contact(contacts):
#    pass
# def get_choice(contacts):
#    pass
# def finish(contacts):
#    pass
# def main():
#    contacts = initalize()
#    get_choice(contacts)
#    finish(contacts)
# if __name__ == "__main__":
#    main()
```

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

- When it comes time to upload your files, although you could use `sftp` to upload multiple files to the Dropbox, you can imagine that multiple copies of `contacts.txt` would be written over each other. What you really want to do is upload a folder that contains all your files. There's just one problem: `sftp` won't allow you to put an entire directory–it only works with files. What to do?

  Use tar to compress your folder of files into a single file:

```
$ tar -czvf address_book-whiter.tar.gz address_book-whiter
```

  This command uses `tar` with the flags `-czvf` to Compress the Zipped files into a single destination File called *address_book-whiter.tar.gz* . It will do so verbosely, meaning it will list files and directories that are being compressed as it goes. The source file for the tar is the directory listed at the end of the command.

This compressed file *address_book-whiter.tar.gz* is what you'll upload to the Dropbox on the server.

- Once your zipped file has been uploaded to the server, the instructor will use a similar `tar` command to expand your uploaded file back into its original contents, as follows:

```
$ tar -xf address_book-whiter.tar.gz address_book-whiter
```

The flags `-xf` indicate that the file should be eXtracted (as opposed to compressed) from the File *address_book-whiter.tar.gz*, and placed in a directory *address_book-whiter*. Then the instructor will be able to examine your files by descending into that directory.

GETTING STARTED

1. Create a folder—*address_book_rwhite* (use your first initial/last name, not mine!)—that will be used to store all files related to your program.
2. Use a text editor to create your initial *contacts.txt* file. (See the *IMPROVEMENTS* section at the end of this document for information on how do this in a slightly more elegant way.)
3. Examine the pseudocode for this program.
4. Use a text editor to begin programming the *address_book.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.
5. Write a function to open the text file for reading, and get the data into a list.
6. Write a function to 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.
7. Using a "bottom-up" design approach, write the function **view** that will allow the user to see all the contacts in the list.
8. Once a function has been written and tested, proceed to writing additional functions as required.
9. Use the `tar` command as specified above to compress all your work into a single archive file.
10. Use `sftp` and the `put` command to upload your file *address_book_whiter.tar.gz* to the server dropbox.
11. Your uploaded files will be what I'll use to evaluate your work.

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

Sample output from running your program might look something like this:

```
$ python address_book.py
Richard's Address Book!
You may: A) Add a contact
         L) Lookup a contact
         E) Edit a contact
         D) Delete a contact
         V) View all contacts
         Q) Quit the Address Book
Your selection: a
```

```
Adding a contact
Contact name (First Last): Susan Smith
Contact email: susan.smith@gmail.com
Contact phone #: na
Richard's Address Book!
You may: A) Add a contact
         L) Lookup a contact
         E) Edit a contact
         D) Delete a contact
         V) View all contacts
         Q) Quit the Address Book
Your selection: v
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: A) Add a contact
         L) Lookup a contact
         E) Edit a contact
         D) Delete a contact
         V) View all contacts
         Q) Quit the Address Book
Your selection: l
Looking up a contact...
Enter contact's name, email address, or phone number: Smith
['Susan Smith', 'susan.smith@gmail.com', 'na']
```

IMPROVEMENTS

It's a bit odd that this program requires you to create an initial *contacts.txt* file using a separate text editor. This certainly isn't the way that most Address Book applications work. 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.