CS 112 Project Assignment: Address Book

Instructor: Dan Fleck

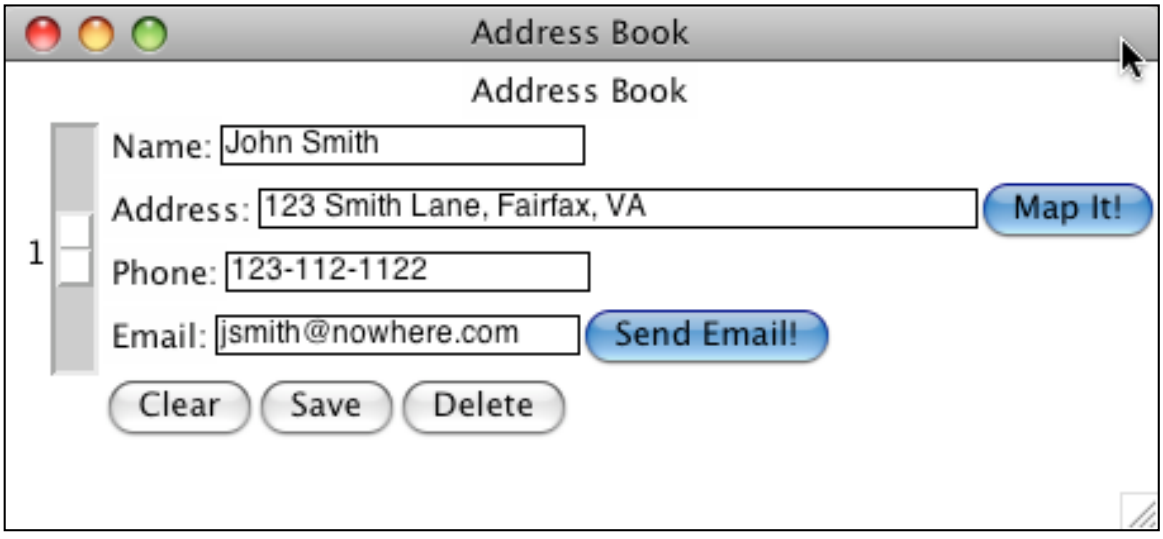
Overview

In this project you will use Python to implement a simple address book. The address book will store multiple data attributes for a person: name, address, phone, email. This information will be easily viewed by scrolling through the entries using a scale widget (vertical slider). During this assignment you will use a class to hold the user information.

You do not need to perform any validation. For this assignment you will be using some basic graphical user interface concepts from the Tkinter library in Python.

Assignment

For CS112 you will need to implement the following parts:



Button Functionality

Clear Button – Removes all the text in the 4 entry boxes (this allows you to easily create a new contact).

Save Button – The save button will save the contact currently shown on screen into the data file. If there is a contact with the exact same name, the save button will update that contact. (The update process is to simply, delete the old contact and then add a new one into the list). If there is no existing contact with the exact same name, you will add a new contact. Make sure to update the maximum value of the slider and resort the data when you add a new contact!)

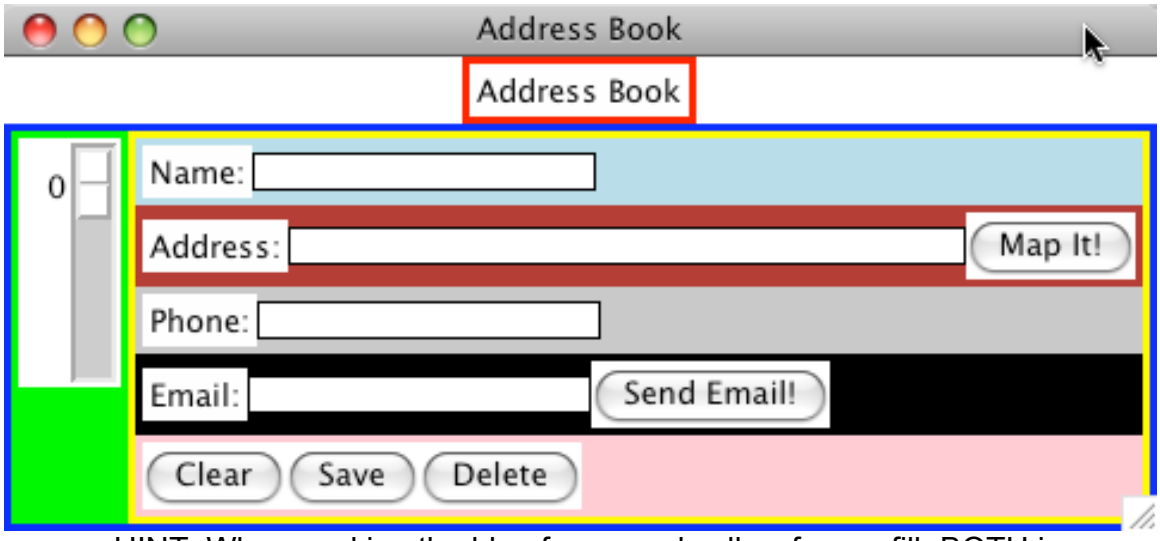
Delete Button – The delete button will delete the contact currently showing on the screen from the file (again you may use the name to find the contact to delete). Then the screen should be updated to show a valid contact (maybe the first one?). Just do not leave the deleted information on the screen! Then, make sure to update the scale maximum if you deleted something.

Send Email – Using the helper code (mail.py) opens an email window with the email address of the contact in it. No validation is needed for email accounts, format, etc… (In a real-world application you would need to validate formats, but in this project you do not.)

Map It – Using the helper code (gLatLong.py) opens Google Maps with the location of the address shown.

Scale Slider – The scale slider on the left should be sized to the total number of address entries in the system. Scrolling through the slider should update the contents of the window as you scroll. If you delete a contact, the maximum slider value should be reduced by one. If you have zero contacts you should disable the scale, by setting it's state to DISABLED. Remember to set it back to NORMAL when you get more contacts!

Window Frames



- HINT: When packing the blue frame and yellow frame, fill=BOTH is helpful.

Some helpful code/information is on the lecture slides and you can also see many Tkinter sites on the web:

http://www.pythonware.com/library/tkinter/introduction/index.htm
http://wiki.python.org/moin/TkInter
http://docs.python.org/lib/module-Tkinter.html

Requirements:

- Contact.py
    - You **must** have a Contact class. The class **must** use only private instance variables and have getters
    - The Contact class **must** implement a comparator function (__cmp__) which will compare the name instance variable so when sorting, the contacts will automatically sort by name. The sorting **must** not be case-sensitive!

- AddressBook.py
    - You **must** have another module (AddressBook.py) that uses the Contact class to create and manage the GUI. The Contact class really just holds the data, the AddressBook.py does most of the work.
    - In AddressBook you will have a list of Contact objects. The AddressBook will store and retrieve the list from a file 'contactlist.txt'. You **must** use pickling to store and retrieve the list of contact! It's easier anyway, so you'll want to. We'll cover this in class.
    - When you first start your program, the contactlist.txt file will not exist. If the file doesn't exist, your list of contacts should just be an empty list. You will need to check for this using 'os.path.isfile'. You **must** not assume the contactlist.txt file is always there!

Starting Hints

- The maximum value of the scale is going to be the length of the contactlist – 1
- To delete item 27 from a list you can use 'del myList[27]'. Of course, in this project you'll need to find the item index first.
- The scale widget has an attribute called "command" that works exactly like the Button's command. The function will be called every time the scale changes.
- Once you have a comparator sorting a list is just 'myList.sort()'
- The save function really performs an update IF the record already exists in the list. To find it, you should search through the list of a

matching name. In my code, I just deleted the matching record and then created a new record from the information in the window. This is simpler in many cases because you already have a function to create a new Contact and a delete function. Combining those two you can do an update.

- You don't need to check for errors anywhere or show error/warning messages. It's okay if you do, but it is not required.
- To update an Entry box, just delete everything from the box and then insert new information.

Bonus Points

+10% bonus – Include a search box that will search for a contact that matches any text in the contact information. So, search should look in all fields (name/address/email/etc…). For example, if my email is bob@coolguy.com and someone searches for "COOL"… they should find this record. Search should not be case sensitive.

+20% bonus – If you do all of the previous bonus and also handle the case where there may be more than one result. How do you present a list of search results? I leave this up to your imagination, but I need some way to see all the results from the search or scroll through only the matches, or something! Note: Doing this you will get a max of 120 total.

My Code

To make your life easier and help you understand what to do, I'll even provide my solution… just download my compiled Python code, and the image needed. Run the code and see what you get!

http://cs.gmu.edu/~dfleck/classes/cs112/spring08/projects/addressbook/AddressBookO.pyc
http://cs.gmu.edu/~dfleck/classes/cs112/spring08/projects/addressbook/ContactO.pyc
http://cs.gmu.edu/~dfleck/classes/cs112/spring08/projects/addressbook/mail.py
http://cs.gmu.edu/~dfleck/classes/cs112/spring08/projects/addressbook/gLatLong.py
http://cs.gmu.edu/~dfleck/classes/cs112/spring08/projects/addressbook/contactlist.txt

(You could spend your time trying to get my source from the compiled code.. but I think you'll be wasting your time… ☺)

What to turn in:

1. Source code (Contact.py and AddressBook.py)
2. A copy of your contactlist.txt which includes 5 contacts.
2. Answers to the following questions:

1. What did you like/dislike about this project?
2. What was the most difficult part of this project?
3. How many hours of work did you do on this project?
4. Did you work with anyone else on the high level issues?

   If so, who (provide first and last name)? (This is okay and will help you learn a lot, but remember this does NOT mean you can share code. If you are talking it's okay… if you are looking at someone else's code that's not okay. If you are typing you should be by yourself.).

   Include in this list any websites you used. (If you're asking for help on a website, the only approved site to post questions about this project is our Blackboard discussion groups.)