Program: Lookup

This program uses many of the things we've discussed in our lectures including: dictionaries, functions, exception handling, optional arguments, methods, looping, conditionals, etc.

The program should ask the user if they'd like to lookup up phone numbers or addresses. Once determined, the program should repeatedly ask the user for a person's name (first and last name) and look up and display the appropriate information. Error conditions should be handled appropriately. An example of the output follows:

```
Lookup (1) phone numbers or (2) addresses: 2
Enter space-separated first and last name: Buck Rogers
error: person not found.
Enter space-separated first and last name: Drew Smith
Street:    285 Andover Lane
City:      Pompano Beach
State:     FL
Zip Code:  33060
Enter space-separated first and last name: Josh Kelly
Street:    7416 Monroe Ave.
City:      Media
State:     PA
Zip Code:  19063
Enter space-separated first and last name:
```

```
Lookup (1) phone numbers or (2) addresses: 1
Enter space-separated first and last name: Drew Smith
Phone:     412-555-2121
Enter space-separated first and last name: Josh Kelly
Phone:     424-555-4053
Enter space-separated first and last name:
```

Data file – address.txt

The program should use the file *address.txt* as the data file. *address.txt* contains a list of names, addresses and phone numbers. *address.txt* has the following comma-separated format, where the first field is the first and last name, the second field is the street, third is the city, fourth is the state, fifth is the zip code, and the last field is the phone number.

```
Drew Smith, 285 Andover Lane, Pompano Beach, FL, 33060, 412-555-2121
.
.
```

To download *address.txt* from within Blackboard, right-click on the file and select "Save link as…" Make sure to save the file to the same folder you plan to write your script in so it will be found when you use the **open** function to open it (otherwise, you will have to provide a full pathname for the file to the **open** function.)

Requirements:

- Name your script <*your-last-name*>*_lookup.py*
- The program should have 3 functions:
  1. An input function to read the comma-separated addresses from file "address.txt" into a dictionary. The function should:
     a. Take no arguments.
     b. Use the first field (first and last name) as the dictionary key. The dictionary value should be the entire line. (I suggest you strip off the newline from each line right after you read it in.)
     c. When you open the file for reading, use a try…catch to detect if the file open fails. If so, print an error end exit the program with an exit code of 1.
     d. return the dictionary as the return value of the function.
  2. A function to format and display the output. Use whatever arguments are appropriate but at least one of them should determine if the function should display just the phone number or the address information. Make this argument an **optional** argument with the default being to print just the phone number. The address output should be formatted neatly, similar to the following:

     ```
     Street:    285 Andover Lane
     City:      Pompano Beach
     State:     FL
     Zip Code:  33060
     ```
  3. A main() function that:
     a. Calls the input function to populate the address/phone dictionary.
     b. Ask the user if they want to lookup phone numbers or addresses, and do so in a loop until they select an appropriate option.
     c. Repeatedly loop, asking for a valid first and last name, and calling the function to format and display the output. If the name does not exist, an appropriate message should be displayed and the user prompted again.
     d. The program should end when the user enters a blank or empty line.

Notes:

1. The program should work if the user enters names in uppercase, lowercase, or a mix of case. Since the names in the file *address.txt* are of mixed case,

for comparisons, you'll want to make use of the string method .**lower()** to convert both the input names and the dictionary keys to lowercase.
2. The dictionary should **not** be a module-level variable.
3. As always, follow proper coding techniques including proper documentation, variable naming, etc.