Lab 2: Phone Contacts
Evan Conley
Mark Huff
April 16, 2018

1. Problem Statement

Create an updatable phone contacts program.

Constraints:

- Constraints use the functions specified in the assignment.
- Can not perform search of contact with matched character case (as it requires all contacts to be stored and handled as lowercase strings)
- Cannot use previous version of Python (i.e. Python 2.x)

Requirements:

- Features:
  - Create a contact
  - Update existing contact (phone, age, email)
  - Display a contact

- Functions:
  - create_contact(contacts, first, last, email, age, phone)
  - update_contact_number(contacts, first, last, phone)
  - update_contact_email(contacts, first, last, email)
  - update_contact_age(contacts, first, last, age)
  - get_contact_email(contacts, first, last)
  - get_contact_age(contacts, first, last)
  - get_contact_number(contacts, first, last)
  - contains_contact(contacts, first, last)
  - display(contacts, first, last)

2. Planning

Steps for implementing test cases:

0. The following steps should be implemented through the main() method
1. Initialize empty ‘contacts’ dictionary
2. Run predefined test cases of functions on ‘contacts’ and print-checks
3. Determine whether output matches expected value
4. If so, then we have successfully test cases

Specification for associated functions:

- create_contact(contacts, first, last, email, age, phone)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the values
    - first: string representing contact's first name
    - last: string representing contact's last name
    - email: string representing contact's email address
    - age: integer representing contact's age in years
    - phone: string representing contact's phone number
- update_contact_number(contacts, first, last, phone)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the values
    - first: string representing contact's first name
    - last: string representing contact's last name
    - phone: string representing contact's phone number
- update_contact_email(contacts, first, last, email)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the values
    - first: string representing contact's first name
    - last: string representing contact's last name
    - email: string representing contact's email address
- update_contact_age(contacts, first, last, age)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the values
    - first: string representing contact's first name
    - last: string representing contact's last name
    - age: integer representing contact's age in years
- get_contact_email(contacts, first, last)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the associated values.
    - first: string representing contact's first name
    - last: string representing contact's last name
    - returns the email string value associated with the contact
- get_contact_age(contacts, first, last)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the associated values.
    - first: string representing contact's first name
    - last: string representing contact's last name
    - returns the integer representing the age of the associated contact
- get_contact_number(contacts, first, last)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the associated values.
    - first: string representing contact's first name

    - last: string representing contact's last name
    - returns the string representing the phone number of the associated contact
- contains_contact(contacts, first, last)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the associated values.
    - first: string representing contact's first name
    - last: string representing contact's last name
    - returns boolean value of True if contact is contained in contacts dictionary; False otherwise
- display(contacts, first, last)
    - contacts: dictionary using contact First and Last Name as a key and contact info for the associated values.
    - first: string representing contact's first name
    - last: string representing contact's last name
    - prints formatted output displaying all the information about the associated contact
        - if the contact does not exist, print a statement informing the user

Will be implementing a combination of lists and dictionaries for this assignment. This allows for ease of access when looking for a specific person.

3. Implementation and Testing

Our first task was to determine how to set up contacts and how to go about storing each individual contact. We ended up settling on having contacts as a dictionary of with each contact having a first and last name as a key tuple. Once the structure of the contacts was settled we went ahead and started to code the functions specified for the assignment. To accomplish these functions we created a helper function that at anytime could access first and last name and convert them into a tuple that can be used in searching out a contact in contacts. Given this helper function it allowed us to streamline the process of searching and updating contacts. After coding the functions we ran tests given to us to make sure all of them worked. All the test cases passed and the program runs as intended.

```
econmang@econDebian:~/Documents/CS/SchoolFiles/cs356/hw/labs/labtwo$ python GroupFiveECMHcontacts.py
Creation of Jim Jones: Passed
Creation of Katie Katz: Passed
Creation of Sarah Sanders: Passed
Updating Sarah Sanders age to 19: Passed
Updating Jim Jones email: Passed
Updating Katie Katz's number: Passed

Katie Katz:
Age: 25
Phone: 907-536-2946
Email: katie.katz@nau.edu

ERROR: Contact does Not Exist
```

4. Reflection and Refactoring

The solution fulfills all the requirements outlined in assignment 2. Our solution to assignment 2 introduced lists and dictionaries to store and update when necessary. Overall, our solution was pretty similar to other group. They also had a helper function that aided in formatting the keys utilized by the defined functions. However, utilizing a dictionary of lists, reliant on knowing which index should hold each data value, they utilized a dictionary of dictionaries, where each value was named. It it our belief that our solution was more memory efficient, as the other team utilized the names of contacts for both keys and values associated with each key. This led to redundant use of names, rather than associating names with only phone, email, and age. As to what could be updated or changed for later iterations of our program, we may wish to implement a solution that allows for a mutable type to be used for names, allowing us to update names of contacts as well as phone, age, and email.