# Assignment 1: Part 2

Contents:

# Phone Book Application

A brief outline of the application:

The main objective of phonebook application is to store and locate phone numbers of a person or an organization in very efficient manner. For example, a user should be able to locate particulars of a person or an organisation by their name.

On initialising the application, the user should be presented with a Menu screen and prompting them to select one of the listed options so that the user can perform the following actions
Add a new contact
Update contact
Search for a contact
Delete a contact
List contacts alphabetically
Exit application

When an option is selected by the user, if required, the application will request for input/s for that relevant function and when required inputs are supplied the function will run and display the outcome on the screen.
On completion the selected task the application will return to the main menu allowing the user to select and perform another activity or exit the application.

Data structure:

A data structure is a data organisation, management, and storage format in computer science that allows for quick access and modification. A data structure, more precisely, is a collection of data values, their relationships, and the functions or operations that can be applied to the data type.
A dictionary data structure is used in this application. A dictionary is a set of key:value pairs separated by commas enclosed in curly brackets. Dictionaries are optimised to retrieve values when the key is known since the keys are indexed.

The "key" will never accept a duplicate value, therefore it will be unique, although the "value" can take practically any value. Dictionaries also have number of built-in operators, methods and functions.

A Lists data structure may also have been used in this case because they are similar in many ways but differ in how their items are accessed. Dictionary entries are accessed by key,

while list members are accessed by number index based on order hence dictionary perform better than list in this scenario. (https://docs.python.org/)

As we are merely trying to store and retrieve names and phone numbers, this makes a dictionary a suitable data structure for an application like a phone book.

**Insertion**:

```
# creating an empty dictionary called phonebook  to store contact details
phonebook  = {}

# inserting a contact name and phone number
phonebook  [name] =  phone number
```

Data inserted as a pair of key and value in dictionaries. A dictionary is a collection of words that is arranged in the order they were typed in, is editable, and does not allow duplicates.

**Update contact's phone number:**

```
# update a contact's phone number
phonebook  [name] = phone number
```

The value of a specific item is altered or updated by referring to its key name when updating.

**Search for a contact:**

```
# look for a name in the phonebook
if  name  in phonebook :
   print (phonebook [name])
else:
  print ("No such name in the dictionary")
```

A dictionary's items can be found by looking up the key name. Keys in dictionaries are indexed, thus if the key is provided, the lookup will be fast..

**Delete a contact**:

```
# remove a particular item
if "name" in phonebook :
   phonebook .pop(name)
else:
  print ("No such name in the dictionary")
```

The item with the supplied key name is removed using the pop() method.

**Sort and List contacts.**

```
# sort the dictionary and prints the list
OD= collections.OrderedDict(sorted(phonebook.items()))
for Name in OD.keys() :
   print("Name = " + str(Name) + " :- Phone Number = " + str(OD [Name]))
```

The dictionary is sorted by “OrderedDict,” which is then looped and prepared to output the list vertically.

Algorithms used for the phone book application:

Pseudocode:

Define main menu function:
Function main_menu
   Pass In: nothing
   Display the following on the screen
        1. Add a new contact
        2. Update contact
        3. Search for a contact
        4. Delete a contact
        5. List contacts alphabetically
        6. Exit application
    Pass Out: nothing
Endfunction

Define add new contact function:
Function add_number
   Pass In: nothing
   Prompt for name
   Check if the name in the phonebook dictionary
      If found, display “name already in the contact” message
   Prompt for phone number
   Check if the input is a number
      If the input is not a number, display "This is not a number. Please enter a valid number"
   Add an entry to phonebook dictionary
   Pass Out: nothing
Endfunctio

Define update contact function:

Function update_number
  Pass In: nothing
  Prompt for name
  Check if the name in the phonebook dictionary
    If there is no matching name, display “name not found” message
  Prompt for phone number
    If the input is not a number, display "This is not a number. Please enter a valid number"
  Update phone number in phonebook dictionary
  Pass Out: nothing
Endfunctio

Define contact search function:

Function search_number
  Pass In: nothing
  Prompt for name
  Check if the name in the phonebook dictionary
    If found, display the name and the phone number
    If not found, display “name was not found” message
  Pass Out: nothing
Endfunctio

Define contact deletion function:

Function del_number
  Pass In: nothing
  Prompt for name to removed
  Check if the name in the phonebook dictionary
  If found, delete the name and the phone number from phonebook dictionary
  If not found, display “name was not found” message
  Pass Out: nothing
Endfunctio

Define list contact function:

Function list_number
  Pass In: nothing
  Loop through the phonebook dictionary
  If no entry found, display “no contact was found” message
  If entries are found, print all the contacts vertically
  Pass Out: nothing
Endfunctio

Initial phonebook application:

declare an empty data dictionary
Crete a variable for menu options value
Call: main_menu and print
while menu option value not equal to 6
  Prompt for menu option between 1 and 6 exclusives
  If menu option is 1
    Call: add_number
  Else if menu option is 2
    Call: update_number
  Else if menu option is 3
    Call: search_number
  Else if menu option is 4
    Call: del_number
  Else if menu option is 5
    Call: list_number
  Else if menu option is not equal to 6
    Call: main_menu and print

Test Plan:

Unit testing:

Tests that validate the functionality of a specific section of code, usually at the function level, are referred to as unit testing.

Test each function:

1. “add_number” function testing.

   input three new entries (name and number)
   check the dictionary for presences of the new record.
   input an existing name and heck for the message notifying that the name already exists.
   Input an invalid number (float or string) and verify that the application displays "This is not a number. Please enter a valid number" message

   Test data:
   Name: Test1
   Number:123456789
   Name: Test2
   Number:234567890
   Name: Test3
   Number:345678901

   Name: Test1

   Name: test20
   Number: 0.2  or Test

2. “update_number” function testing.

   supply an existing name and a new phone number and verify that the record has been updated with new number
   supply a new name and verify that the message displayed indication that name does not exist.
   Input an invalid number (float or string) and verify that the application displays "This is not a number. Please enter a valid number" message

Test data:
Name: Test1
Number:22445566

Name: Test25

Name: test20
Number: 0.2  or Test

3. “search_number” function testing.
   supply an existing name and verify the result (resulting number should be 22445566)
   supply a name that does not exist in the phonebook and verify that a message displayed indication that name does not exist

   Test data:
   Name: Test1

   Name: Test25

4. “del_number” function testing.
   supply an existing name and verify the that is it no longer in the phonebook once deletion completed.
   supply a name that does not exist in the phonebook and verify that a message displayed indication that name does not exist

   Test data:
   Name: Test1

   Name: Test26

5. “list_number” function testing.
   List records and verify the following listed

   Name: Test2
   Number:234567890
   Name: Test3
   Number:345678901

System testing:

System testing examines a fully integrated system to ensure that it fits its specifications. A system test might include, for example, testing a login interface, then writing and updating an entry, as well as deletion (or archiving) of entries, and finally logoff or exiting the application.

Initiate the application

Check the main menu displays the following

1. Add a new contact
2. Update contact
3. Search for a contact
4. Delete a contact
5. List contacts alphabetically
6. Exit application

Check the main menu for a prompt asking user to enter a selection number… (1 - 6)

Enter number 1 and verify that system allows you to add a new record and retunes to selection option on completion

Repeat the above test for options 2,3,4 and 5 and verify that the corresponding functions are working.

Test option 6 and see that the application is closed

test all the scenario tested for the unit testing.

References:

The Python Software Foundation, 2021, Python DOC, viewed28 June 2021, https://docs.python.org/3.9/tutorial/datastructures.html#dictionaries

Readme:

The main objective of phonebook application is to store and locate phone numbers of a person or an organization in very efficient manner.
On initialising the application, the user should be presented with a menu and a dialog.

Here is a brief commentary of the solution implemented.

Application developed and tested on Python 3.9.5.

First a function called ‘menu’ created for displaying the following options.

1. Add a new contact
2. Update a contact
3. Search for a contact
4. Remove a contact
5. View existing contacts
6. Exit Phonebook

Then a function is created (menuchoice) and used for prompting user for their input in relation to the options presented in the menu. inbuilt function input() is used for prompt and input is verified using isdigit() method and if the input is a digit then the value is retuned.

Then an empty dictionary data structure was initiated using the default python dictionary representation “{}” and this was done in order to store contact names and phone numbers in the form of a dictionary for easy mapping.

And the “menu” function is called to display the user options on the screen.

If the phonebook is empty, user is prompted to enter up to 5 contacts by using the built-in function and if the entry is 5 or below then a for loop is used to iterate and call the add_contact() function in order to add the contacts. If not, user is informed.

A variable called “menu_choice” is declared to store the user’s selection and assigned an initial value of “0”.

And then a while loop is initiated (this is the main elements that runs and controls the phonebook application) to run until the user input is not 6 (i.e. the loops only breaks if the user input is 6) and the ‘menuchoice’ function that was created earlier was called by storing in a variable name “menu_choice”.

Conditions were created inside the while loop using conditional statements, for making decisions for an entry chosen by the user. Sub functions are called to perform tasks based

on the user entry and an error message is printed if the user entry is not within the expected value.

Here are the sub functions and their descriptions:

For user entry: 1 - add_contact() is called.

A variable is initiated. The inbuilt function input() is called to prompt for user input and this user input is stored in the newly created variable and then a decision block is created to check, if the entry already exists, If yes, then the user is notified along with currently held data. It is also checked if the entry is blank using inbuilt len() function and if yes, then the user is informed.
If the entry does not exist and if not blank, then a second variable is initiated and again function input() is called for user input and the response is stored in the created variable. The variable is also checked using isdigit() method to see if all the characters are digits. If yes, the phonebook is updated using the Python dictionary 'update method', which is one of the methods used to add new items to a Python dictionary. This was done by mapping the contact name to the phone number. Otherwise, the user is informed that the contact is not created.

For user entry: 2 - update_contact() is called.

A variable (name) is initiated. The inbuilt function input() is called to prompt for user input and this user input is stored in the created variable. Then a decision block is created to check, if the entry exists. If not, user is informed that the entry is not found.

If the entry exists, as per in add_contact() second variable initiated and verified for phone number. If verified, the phonebook is updated. Otherwise, the user is notified that the entered number is invalid, and the contact is not updated.

For user entry: 3 - search_contact() is called.

A variable (name) is initiated. The built-in function input() is used to prompt the user for input, which is then saved in the newly created variable. Then a decision block is created to

check if the entry exists. If it does, the contact name and phone number are displayed, otherwise, an error message is printed to inform the user.

For user entry: 4 - remove_contact() is called.

A variable is initiated. The built-in function input() is used to prompt the user for input, which is then saved in the newly created variable. Then a decision block is created to check if the entry exists. If it does, customer is prompted for confirmation and if confirmed, contact details removed using Python keyword “del” and a message is displayed to indicate the removal. And if not confirmed, user is informed that contact is not removed. otherwise, a “name not found message” is displayed.

For user entry: 5 - list_contact() is called.

A decision is set using the built-in len() that if the initial phonebook is empty, i.e. no contact has been stored before, it returns a message to indicate that the phonebook is empty. If not, the user is invited to select the order in which they want to display the contacts (asc or desc) using built-in function input (). The sorted() function is used to sort the dictionary on ascending or descending order. Sorted names and phone numbers are displayed in a vertical format.

Finally, for entry 6, the while loop initiated at the beginning breaks, hence stopping the application.

Also, the final else statement is used to display an error message, if the user enters a wrong entry (a digit != 6).

Testing and validation:

Testing to be conducted based on what was defined in Assignment1_Part1.docx (page7) of this assignment with few improvements and addition to the test cases to enhance the quality and relevance.

Unit testing: (test each functions)

1

“add_number” function testing.

| **Test case** | **Results** | **Status** |
|---|---|---|
| add five contacts | successfully added contacts (as per test data) | Pass |
| add contact where name is empty | Attention !!!...Invalid entry! no name supplied! | Pass |
| enter string or float for phone number | Attention !!!...Invalid phone number entered! Contact is not created | Pass |
| add an existing name (test1) | Attention !!!...Name already exist! and the number is: 123456789 | Pass |
| add a contact with no phone number | Attention !!!...Invalid phone number entered! Contact is not created | Pass |

Additional test data:
Name: Atest,Number:1122556644
Name: btest,Number:235689147
Name: Ctest,Number:987456321
For full test data refer to Assignment1: Part1

2
“update_number” function testing.

| Test case | Results | Status |
|---|---|---|
| update number "22445566" to name "Test1" | Contact updated, new number for Test1 is 22445566 | Pass |
| update a contact name that doesn't exist (Test25) | Attention !!!... Test25 was not found | Pass |
| enter string or float for phone number | Attention !!!...Invalid phone number entered! Contact is not updated | Pass |

For test data refer to Assignment1: Part1

3
“search_number” function testing.

| Test case | Results | Status |
|---|---|---|
| Search for an existing number (Test1) | The number is 22445566 | Pass |
| Search for a contact name that doesn't exist (Test25) | Attention !!!...Name: Test25 was not found! | Pass |

For test data refer to Assignment1: Part1

4
“del_number” function testing.

| Test case | Results | Status |
|---|---|---|
| try to delete an existing number (Test1) | are you sure that you want to remove contact: Y/N? | Pass |
| - decline by typing "N" or "n" | Contact Test1 not removed. (phonebook still contain the contact) | Pass |
| - confirm by typing "Y" or "y" | Contact: Test1 has been removed (contact no longer in the phonebook) | Pass |
| | | |

| | | |
|---|---|---|
| delete a conect name that doesn't exist (Test26) | Attention !!!...Name: Test26 was not found! | Pass |

For test data refer to Assignment1: Part1

5
“list_contact” function testing.

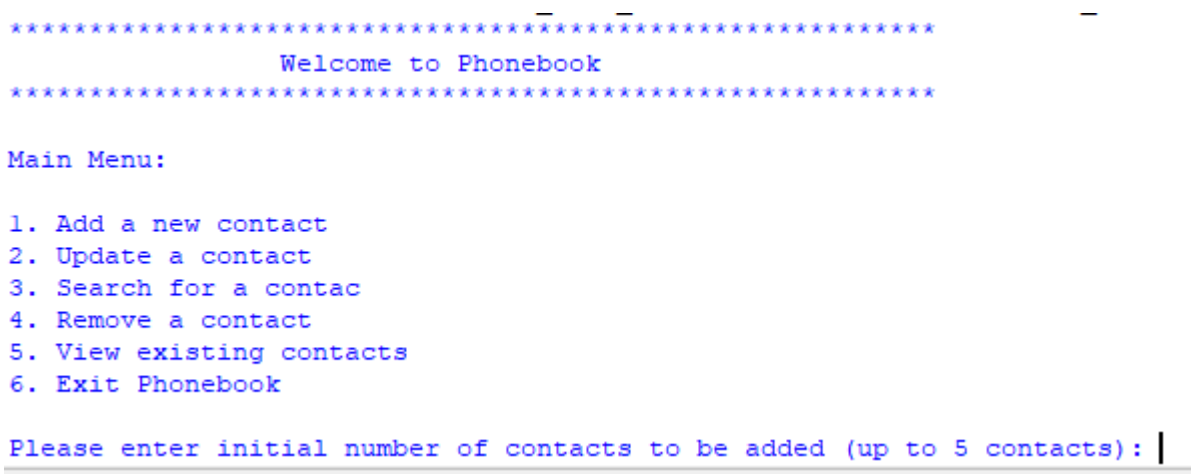
| **Test case** | **Results** | **Status** |
|---|---|---|
| View existing contacts | option presented to sort names in Asc or Desc Order | Pass |
| Option1 (Asc order) | Name: ATEST Number: 1122556644 | pass |
| | Name: BEST Number: 235689147 | |
| | Name: CTEST Number: 987456321 | |
| | Name: TEST2 Number: 234567890 | |
| | Name: TEST3 Number: 234567890 | |
| | | |
| Option1 (Desc order) | Name: TEST3 Number: 2345678901 | pass |
| | Name: TEST2 Number: 234567890 | |
| | Name: CTEST Number: 987456321 | |
| | Name: BEST Number: 235689147 | |
| | Name: ATEST Number: 1122556644 | |
| | | |
| If no entry in the phone book | Phonebook is empty!! | Pass |

System testing:

Checking the algorithm outlined for phonebook application.

Initiate the application.

If the phonebook is empty, display main menu and ask user to enter a up to 5 contacts to be added. Handle invalid entry by displaying a message.




If the phonebook is not empty, display main menu for a prompt asking user to enter a selection number… (1 - 6) and each options prompt:

```
************************************************************
                Welcome to Phonebook
************************************************************

Main Menu:

1. Add a new contact
2. Update a contact
3. Search for a contac
4. Remove a contact
5. View existing contacts
6. Exit Phonebook
------------------------------------------------------------

Enter an option(1,2,3,4,5 or 6) or press Enter for main menu:
```

Option1

```
------------------------------------------------------------

Enter an option(1,2,3,4,5 or 6) or press Enter for main menu: 1
Add a new Contact
Name:
```

Option2

```
------------------------------------------------------------

Enter an option(1,2,3,4,5 or 6) or press Enter for main menu: 2
Update a Contact
Name:
```

Option3

```
------------------------------------------------------------

Enter an option(1,2,3,4,5 or 6) or press Enter for main menu: 3
Enter the name of the contact you wish search:
```

Option4

```
------------------------------------------------------------

Enter an option(1,2,3,4,5 or 6) or press Enter for main menu: 4
Remove a Contact
Name:
```

Option5

```
------------------------------------------------------------

Enter an option(1,2,3,4,5 or 6) or press Enter for main menu: 5

How do you want the contacts ordered?
1. Names in ascending order
2. Names in descending order:

Type in a number (1 or 2):
```

Option6

```
------------------------------------------------------------

Enter an option(1,2,3,4,5 or 6) or press Enter for main menu: 6
Goodbye!

>>>
```

tested all the scenario tested for the unit testing and all worked as expected.

Reference:

Wimalendran, p (2021) Assignment1_Part1.University of Essex. Unpublished essay.