

Assignment 1 Part 2: **Testing Strategy**

In Assignment 1, the proposed strategy for testing the Guardian Directory code was unittest. However, as the code was being written, Manual Testing was determined to be more suitable. This is because the developer can pass through the checkpoints proposed in Assignment 1 while checking the functionality of its Graphical User Interface (GUI). According to Sharma (2014), Manual Testing is personally executed by humans without the help of automation tools in order to check and document how functions in the code run. Jacob and Mani (2020) also state that validating the operation of the Graphical User Interface (GUI) of programmes can be confirmed through Manual Testing.

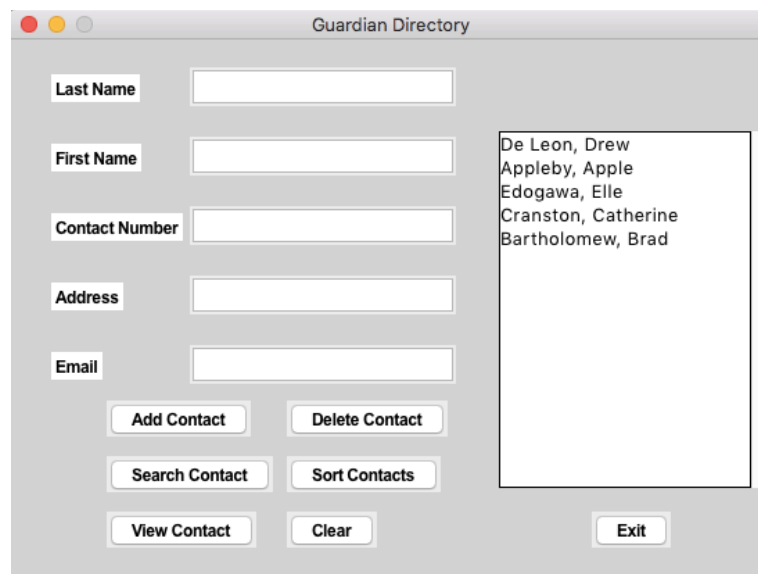


Figure 1: Checking the GUI aesthetic through Manual Testing

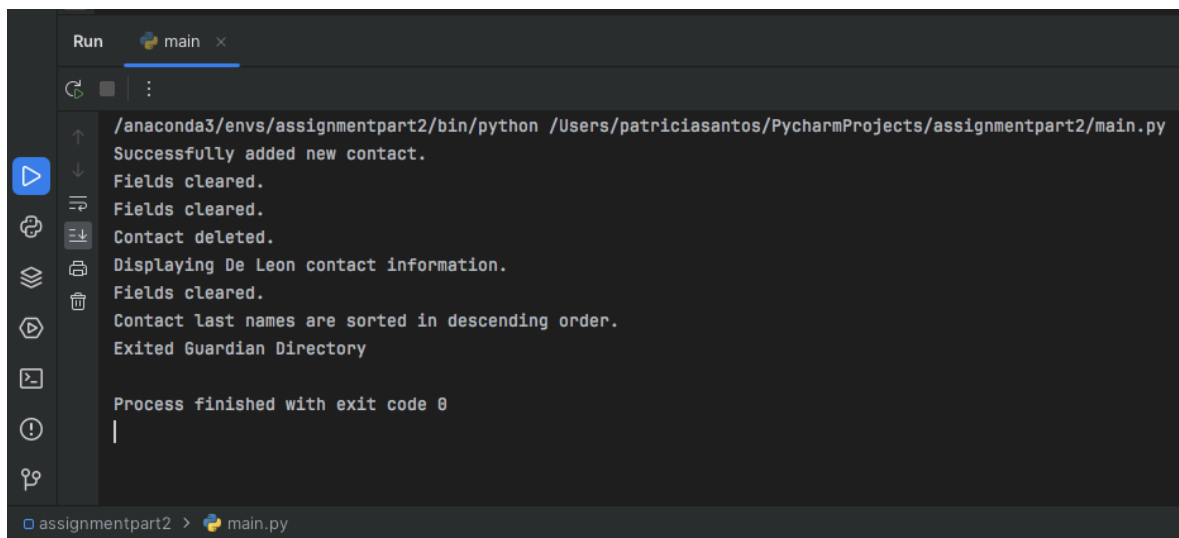
The Manual Test for the Guardian Directory was run on PyCharm, where the code was written, while the GUI was running. The checkpoints proposed in Assignment 1 have been revised to add more details, as well as to explain what makes the functions within the code either Pass or Fail.

| | OK (Pass) | F (Fail) |
|--------|--|--|
| Insert | User adds contact: <ul style="list-style-type: none"> - Last Name - First Name - Contact Number - Address - E-mail Result: Successfully added Contact | - No contact was added |
| Delete | If user types user Last Name and then clicks either View Contact or Search Contacts or; If user selects contact to be deleted from the list Result: Contact deleted | <ul style="list-style-type: none"> - No contact was selected - Contact is not in the directory - First letter of the Last Name is not capitalised - Misspelt Last Name - Last Name was not supplied |
| Search | User types contact Last Name in the field. User clicks: "Search" Result: Contact Information will be displayed | <ul style="list-style-type: none"> - No contact was selected - Contact is not in the directory - First letter of the Last Name is not capitalised - Misspelt last name - Last Name was not supplied |
| Sort | User clicks: "Sort" Result: Last names will be displayed in descending order | - No action was taken |

Figure 2: Updated test checkpoints followed during Manual Testing

In Figure 3, it is showing that as the code was being tested manually, the PyCharm tool window was simultaneously showing the result of the code as it was run. As the add,

delete, search, and sort contacts were tested successfully in the GUI, the programme was able to execute its intended functions. Therefore, we can conclude that the Guardian Directory is running correctly, as seen on the PyCharm tool window run results, and based on the correct execution of the button functions in its GUI.



The image shows a screenshot of the PyCharm IDE's Run window. The window title is 'Run' with a sub-tab 'main'. The output console displays the following text:
/anaconda3/envs/assignmentpart2/bin/python /Users/patriciasantos/PycharmProjects/assignmentpart2/main.py
Successfully added new contact.
Fields cleared.
Fields cleared.
Contact deleted.
Displaying De Leon contact information.
Fields cleared.
Contact last names are sorted in descending order.
Exited Guardian Directory
Process finished with exit code 0
The left sidebar of the Run window contains various icons for debugging and execution control. At the bottom of the window, the file path 'assignmentpart2 > main.py' is visible.

Figure 3: Running the code through Manual Testing

References:

- Jacob, P.M. & Mani, P. (2020) A Framework for Evaluating Performance on Software Testing Tools. *International Journal of Scientific Technology & Research*. 9(2): 2175-2180. Available from: <https://www.ijstr.org/paper-references.php?ref=IJSTR-0120-29613> [Accessed 16 October 2023].
- Sharma, R.M. (2014) Quantitative Analysis of Automation and Manual Testing. *International Journal of Engineering and Innovative Technology (IJEIT)*. 4(1): 252-257. Available from: https://www.ijeit.com/Vol%204/Issue%201/IJEIT1412201407_46.pdf [Accessed 16 October 2023].

Assignment 1 Part 2: README.md

Note: You can find the README.md on <https://github.com/patzsantos/GuardianDirectory/blob/main/README.md>.

There, you can see the .gifs moving that demonstrate how the functions work.

Guardian Directory

The Guardian Directory (GD) is a phonebook programme intended for the record-keeping of the information of parents and guardians of students in Early Learning Centres (ELC) such as daycares and nurseries.

The parental and guardian records kept in the GD are especially helpful in the event that correspondence with them is necessary. An example of this is, when there are any school announcements, or if teachers need to communicate with the parents or guardians, all they have to do is search for their contact information from the Guardian Directory. They can quickly reach out to them by phone, and if not, through e-mail. The address is there in case any mail, such as invitations to school events, are needed to be sent out by the ELCs.

Running the code

This code was written on **PyCharm** and runs on **Python 3.11**.

If you are having trouble running the code in Codio, please run it on PyCharm. The preview of the GUI, which was imported from **Tkinter**, can also be previewed in PyCharm.

Some changes to running the program has been modified and improved from the data structure and algorithm design proposal based on Assignment 1.

Functions of the Guardian Directory

The Guardian Directory has 7 functions:

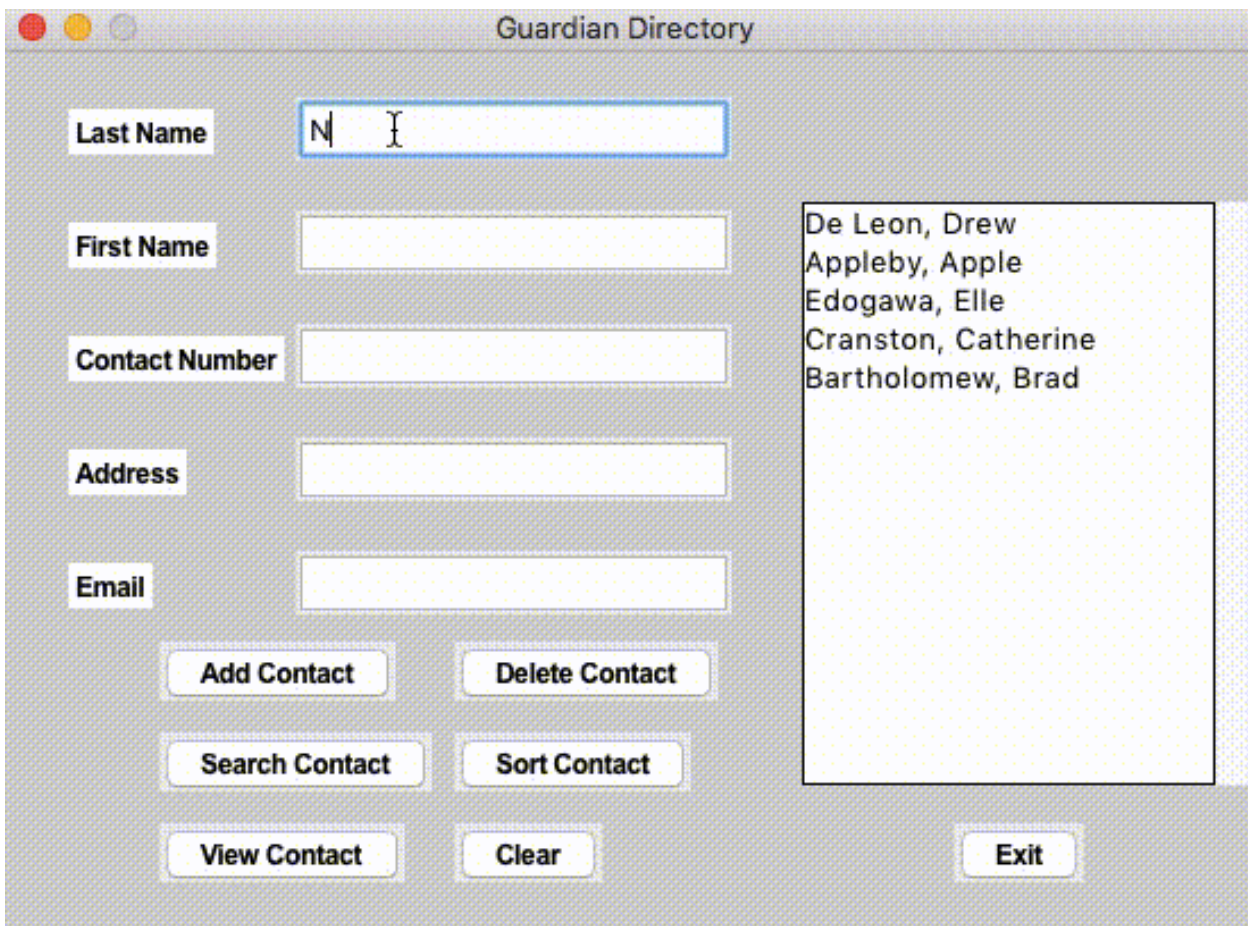
1. Add contact
2. Delete contact
3. Search contact
4. Sort contact
5. View contact

6. Clear fields
7. Exit

In the fields, the contact information details can be viewed. The details that will be asked upon listing a contact are:

1. Last name
2. First name
3. Contact Number
4. Address
5. E-mail

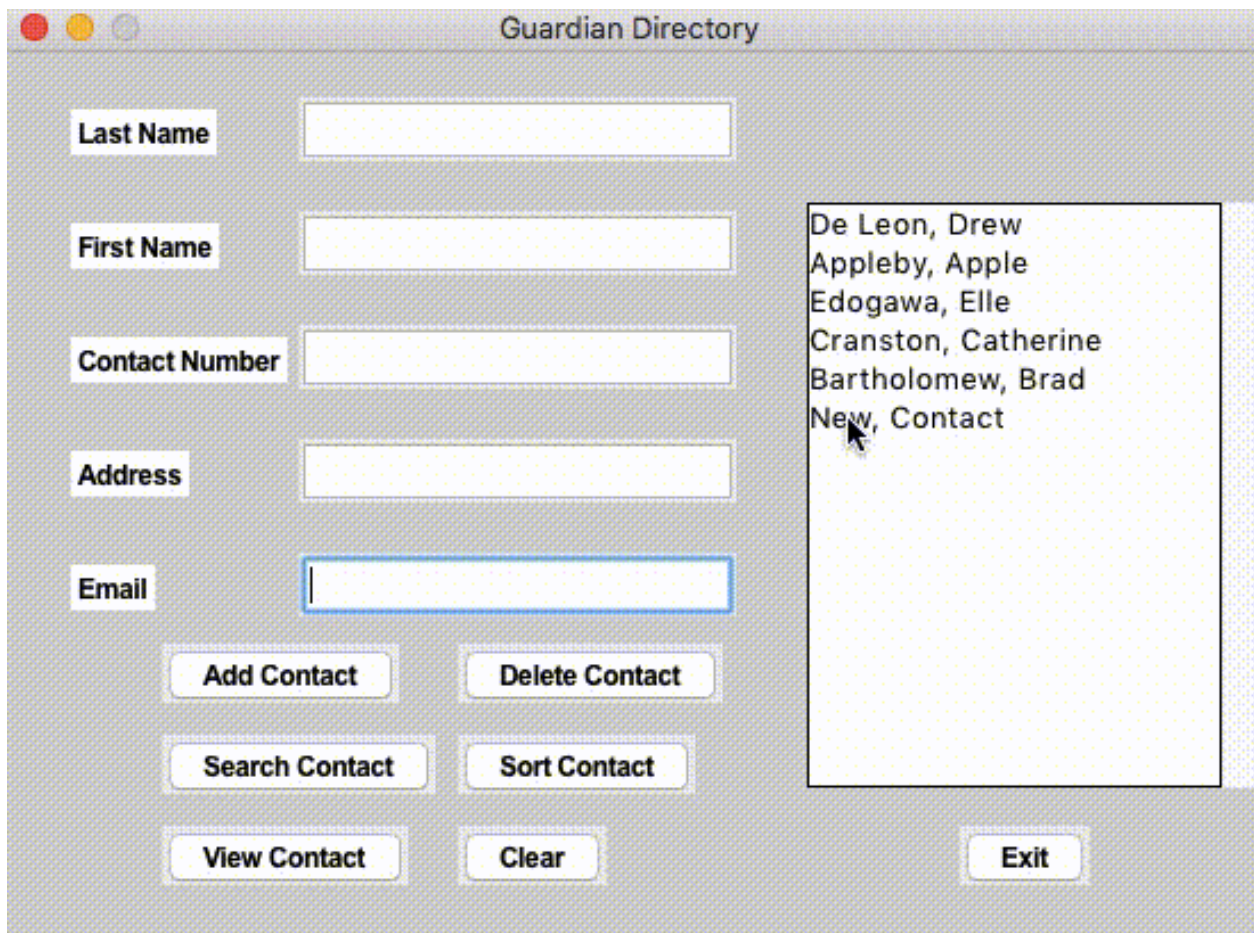
> Add Contact



The screenshot shows a window titled "Guardian Directory". On the left, there are five input fields with labels: "Last Name" (containing "N"), "First Name", "Contact Number", "Address", and "Email". Below these fields are five buttons: "Add Contact", "Delete Contact", "Search Contact", "Sort Contact", and "View Contact". At the bottom right is an "Exit" button. On the right side of the window, there is a list box containing the following names: "De Leon, Drew", "Appleby, Apple", "Edogawa, Elle", "Cranston, Catherine", and "Bartholomew, Brad".

If you want to add a parent or guardian to the directory, just fill in the fields with contact *Last Name*, *First Name*, *Contact Number*, *Address*, and *E-mail*. Once done, click **Add Contact**.

> Delete Contact



The screenshot shows a window titled "Guardian Directory". On the left, there are five input fields with labels: "Last Name", "First Name", "Contact Number", "Address", and "Email". Below these fields are seven buttons: "Add Contact", "Delete Contact", "Search Contact", "Sort Contact", "View Contact", "Clear", and "Exit". On the right side, there is a list box containing the following text: "De Leon, Drew", "Appleby, Apple", "Edogawa, Elle", "Cranston, Catherine", "Bartholomew, Brad", and "New, Contact". A mouse cursor is pointing at the "New, Contact" entry in the list box.

If you want to delete a contact, select the contact you want to delete from the list, and click **Delete Contact**.

There are other alternatives to deleting a contact.

First is by typing in the contact **Last Name** in the field, then clicking **Search Contact**, after which you can select **Delete Contact**.

The second alternative is typing in the contact **Last Name** in the field, then clicking **View Contact** before selecting **Delete Contact**.

These, however, are advisable only when there are many contacts stored in the GD, which makes finding them difficult.

Here, we can find the first difference from Assignment 1, where First Name was required to be written in the field. This decision to change to last name was made since it is safe to

assume that the ELC employees would find it easier to look for the family name of the student in the GD.

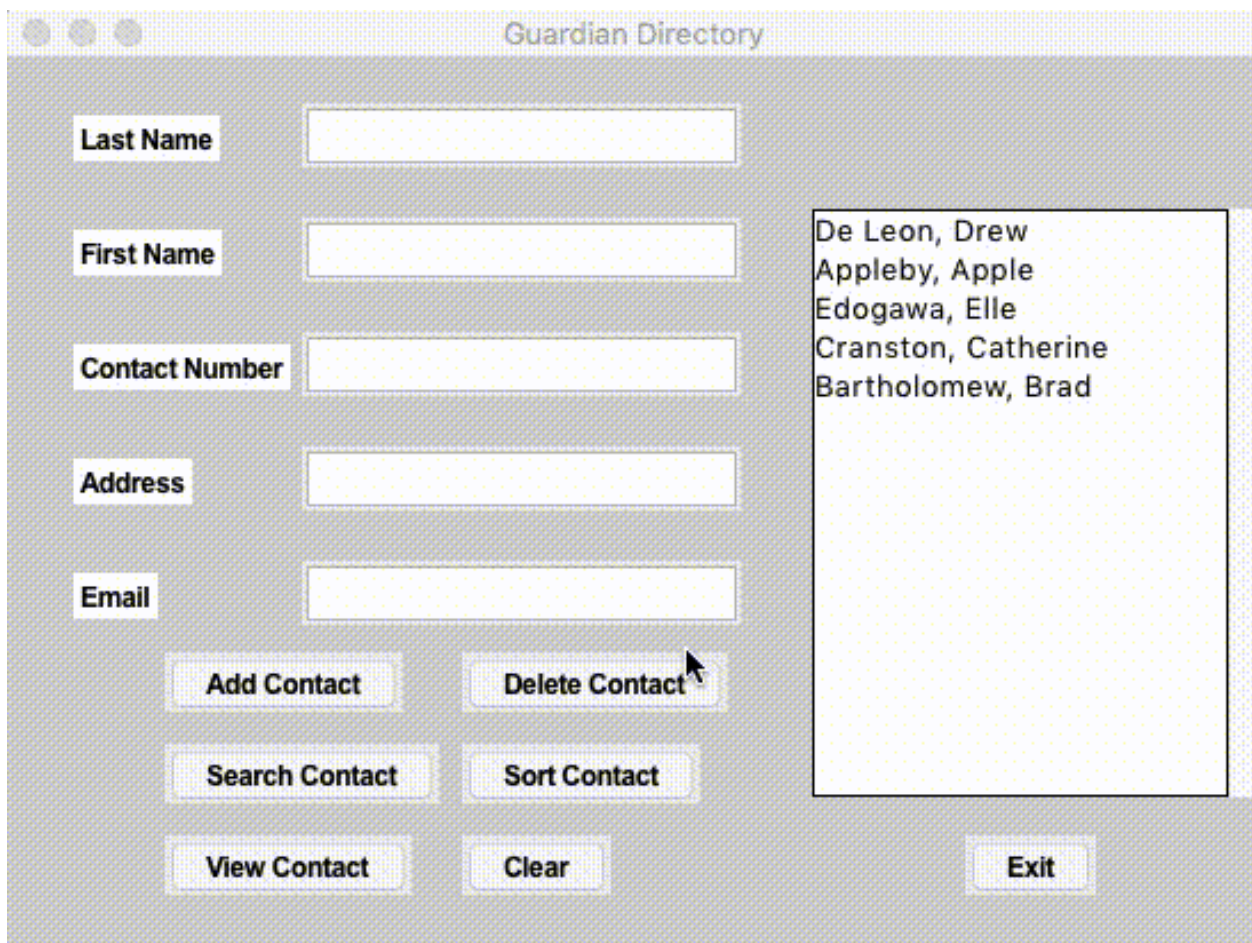
This change would continue to be applied in the next functions, which are **Search Contact** and **Sort Contacts**.

> Search Contact

The screenshot shows a window titled "Guardian Directory" with a light gray background. On the left, there are five input fields with labels: "Last Name", "First Name", "Contact Number", "Address", and "Email". The "Last Name" field is active, showing a cursor. Below these fields are six buttons: "Add Contact", "Delete Contact", "Search Contact", "Sort Contact", "View Contact", and "Clear". On the right side, there is a list box containing the following text: "De Leon, Drew", "Appleby, Apple", "Edogawa, Elle", "Cranston, Catherine", and "Bartholomew, Brad". At the bottom right, there is an "Exit" button.

If you want to search for contact information, type in the *Last Name* and click **Search Contact**. The rest of their details will then be displayed.

> Sort Contacts

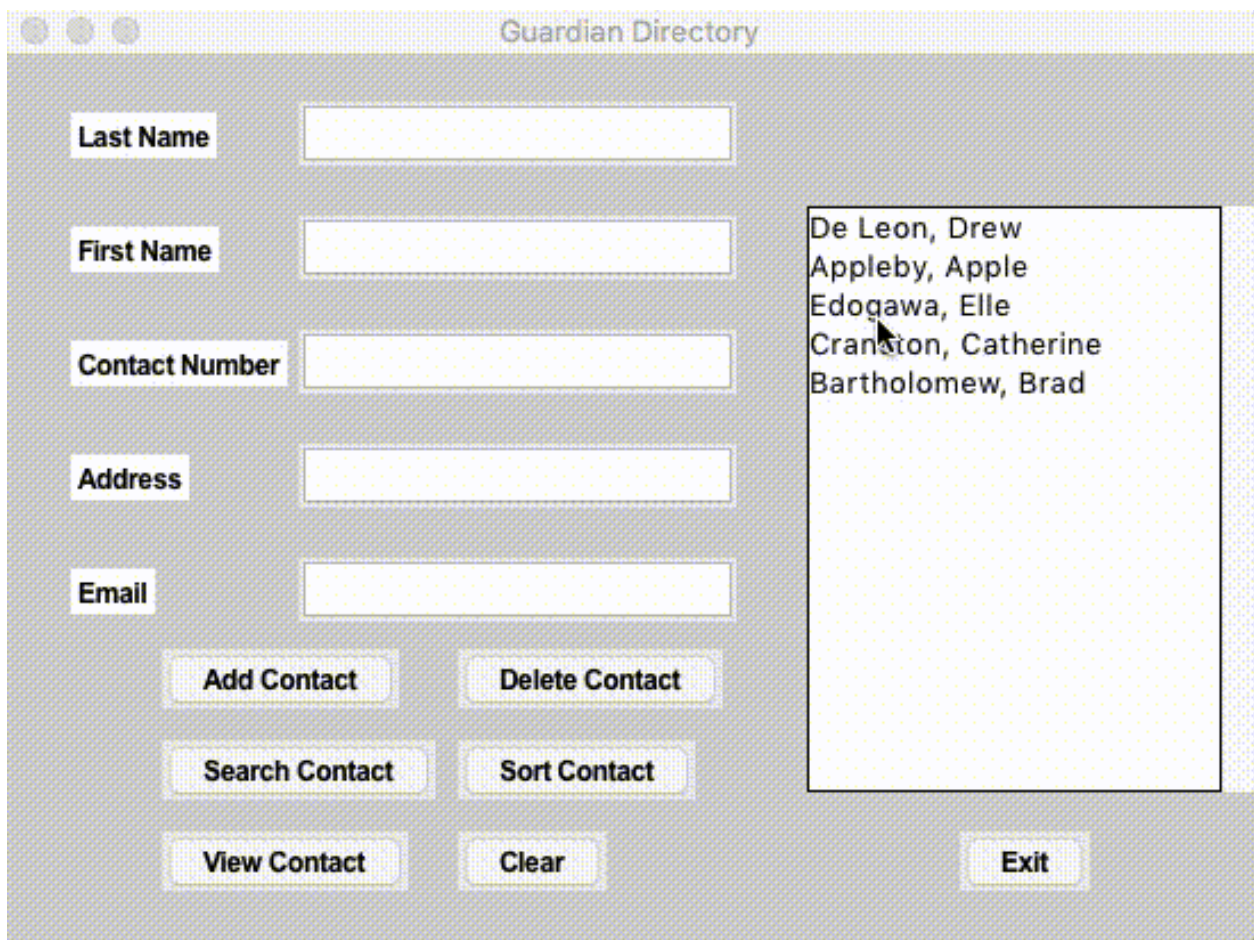


The screenshot shows a graphical user interface for a 'Guardian Directory' application. The window has a title bar with three standard OS window controls (minimize, maximize, close) on the left. The main area is divided into two sections. On the left, there are five input fields with labels: 'Last Name', 'First Name', 'Contact Number', 'Address', and 'Email'. Below these fields are seven buttons: 'Add Contact', 'Delete Contact', 'Search Contact', 'Sort Contact', 'View Contact', 'Clear', and 'Exit'. On the right, there is a large rectangular text area displaying a list of names: 'De Leon, Drew', 'Appleby, Apple', 'Edogawa, Elle', 'Cranston, Catherine', and 'Bartholomew, Brad'. A mouse cursor is pointing at the 'Delete Contact' button.

When you click **Sort Contacts**, the directory will list the *Last Names* in descending order.

This is the second difference from Assignment 1, where the initial proposal was to just arrange the first names alphabetically with the sort function.

> View Contact and Clear Fields



The screenshot shows a graphical user interface for a program titled "Guardian Directory". On the left side, there are five input fields with labels: "Last Name", "First Name", "Contact Number", "Address", and "Email". Below these fields are seven buttons arranged in three rows: "Add Contact" and "Delete Contact" in the first row, "Search Contact" and "Sort Contact" in the second row, and "View Contact" and "Clear" in the third row. On the right side, there is a list box containing the following text: "De Leon, Drew", "Appleby, Apple", "Edogawa, Elle", "Cranston, Catherine", and "Bartholomew, Brad". At the bottom right of the window is an "Exit" button.

If you want to view a contact and their details, click **View Contact**.

If you want to clear the fields, click **Clear**.

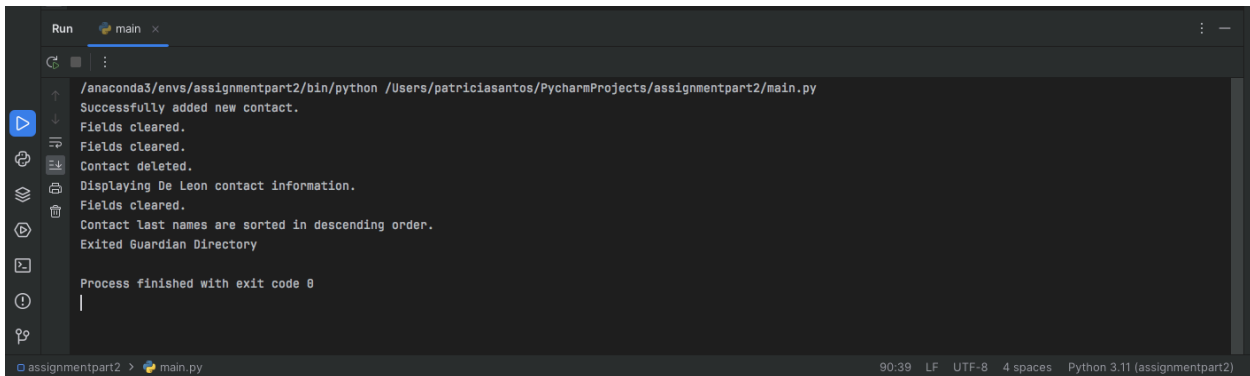
These two additional functions were proposed and added by [michael305y](#), who helped me, through discuss.python.org, how to make the **Search Contact** and **Sort Contacts** functions work.

> Exit the programme

Click the **Exit** button to close the programme.

Testing the Code

Manual Testing was used to test the code. This changed from the supposed unittest that was proposed in Assignment 1. The code run successfully. The GUI displays and operates the correct buttons when the programme is opened, and the programme can be closed when the exit button is selected. The programme is also able to function successfully, as it can add, delete, search, sort, view, and clear contacts.



```
Run main
/anaconda3/envs/assignmentpart2/bin/python /Users/patriciasantos/PycharmProjects/assignmentpart2/main.py
Successfully added new contact.
Fields cleared.
Fields cleared.
Contact deleted.
Displaying De Leon contact information.
Fields cleared.
Contact last names are sorted in descending order.
Exited Guardian Directory

Process finished with exit code 0
```

How to Better Improve the Code

As mentioned in Assignment 1, completing the U function, which stands for Update, in the CRUD method, is one way to further improve the functionality of the Guardian Directory.

Another thing to look into is the security of the data privacy of the guardians stored in the GD.

What software should be used in order to strengthen the cyber security of the GD?

How long should the contacts be stored in the directory?

These are questions to ask in the future, when further developing the programme.

Purpose

This code was built as a requirement in the Launching Into Computer Science Module of University of Essex Online, and serves as the Assignment Part 2 project of the course.

Credits

As I mentioned earlier, a big shoutout to [michael305y](#), who helped a great deal in not just kindly pointing out my mistakes, but also in teaching me how to make my code actually work the way I want it to.

The GUI of the Guardian Directory was inspired by the tutorials of [Sam CodeHub](#) and [Geekedu](#).

The gifs of how the function work, as seen above, were converted on [cloudconvert](#).

Introduction

Dictionaries in Python hold information that can be accessed using keys (Mastrodomenico, 2022). Python dictionaries are able to retrieve stored data quickly (Overland, 2017). Therefore, they are ideal in programmes that are specifically designed for record keeping such as phonebooks. With these in mind, a Python dictionary will be used to create a contact book called “Guardian Directory (GD)”. GD is intended to be a phonebook programme that can be used by early learning centers (ELC) such as nurseries, daycares, and preschools.

The GD can be used for the record management of the contact information of family members and guardians of students in ELCs. They can be used when ELC teachers and staff need to look for their contact information if there is a need to correspond with them regarding school announcements or for matters related to the student they are responsible for. These types of databases are helpful in early childhood learning centers.

Data Structure

The data structure that will be utilised for the GD is the associative array, also known as dictionaries. They are regarded as associative arrays due to their ability to plot associated values with their respective keys (Sturtz, 2018). According to Mastromatteo (2020), this is due to hash functions. He further explains that since values are set and hash yields identical results, the computing process is speedy.

We learn from Chris (2022) that CRUD, which is short for Create, Read, Update, and Delete, is a requirement in executing applications. He adds that they should contain a

user interface, server, and database. The Guardian Directory partly applies the CRUD operations. We mentioned partly, since this programme currently does not have the option to update the contact information, only insert (create), search (read), and delete. In lieu of this, we have the sorting option, which would display the contact names alphabetically. Data that will be held in the database are the first name, last name, address, contact number, and e-mail of the guardians.

Algorithm Design

In order to better demonstrate how to use the GD, we will use Flode. Flode is designed to demonstrate flowcharts (Codio, 2023). Please note that Flode shows the start and end of each process. Start will always be the first step, while end will always be the last. Only the steps between the beginning and the end of the program will be explained in the step by step process in the flowcharts below.

We start with inserting the contacts. The programme will prompt the user to add a contact. Once selected, the user can insert the following information: first name, last name, address, contact number, and e-mail address of the new contact. The GD will display the message “Successfully added new contact” when the save option is selected.

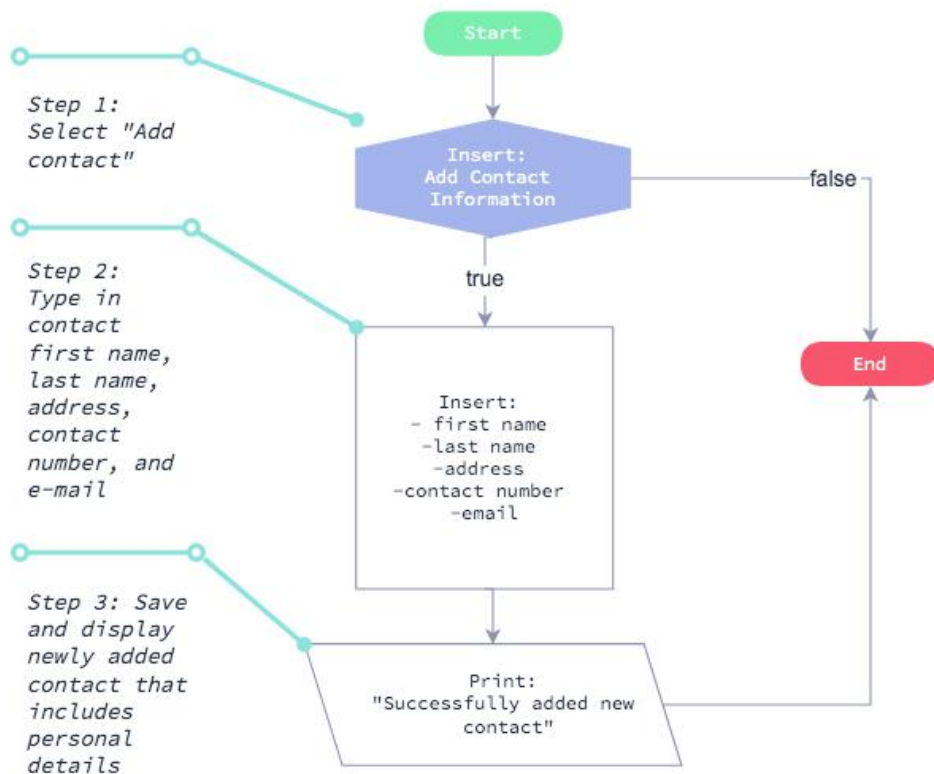


Figure 1: Adding a new contact

The next command is deleting a contact information in its entirety. The user can type in the name of the contact. When the contact is selected, there is an option to remove it together with all the details that are stored with it. Once this action is confirmed, and a message that says "Contact deleted" is displayed, they will be removed from the database.

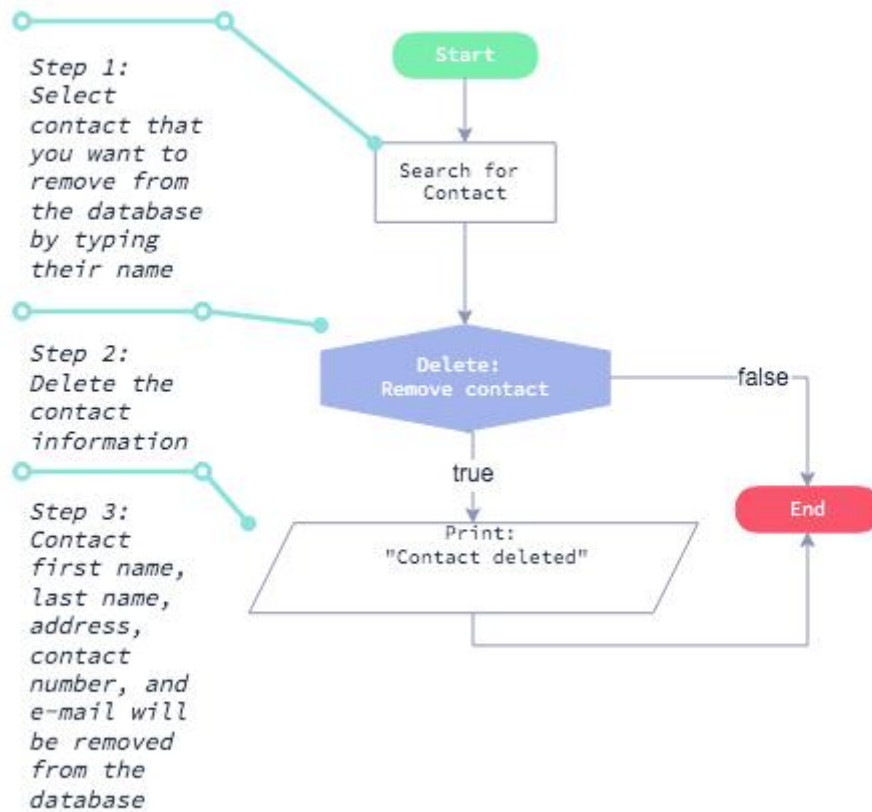


Figure 2: Deleting a contact

In order to search for a contact, the user should type in their name in the search bar. The GD will then display the contact information.

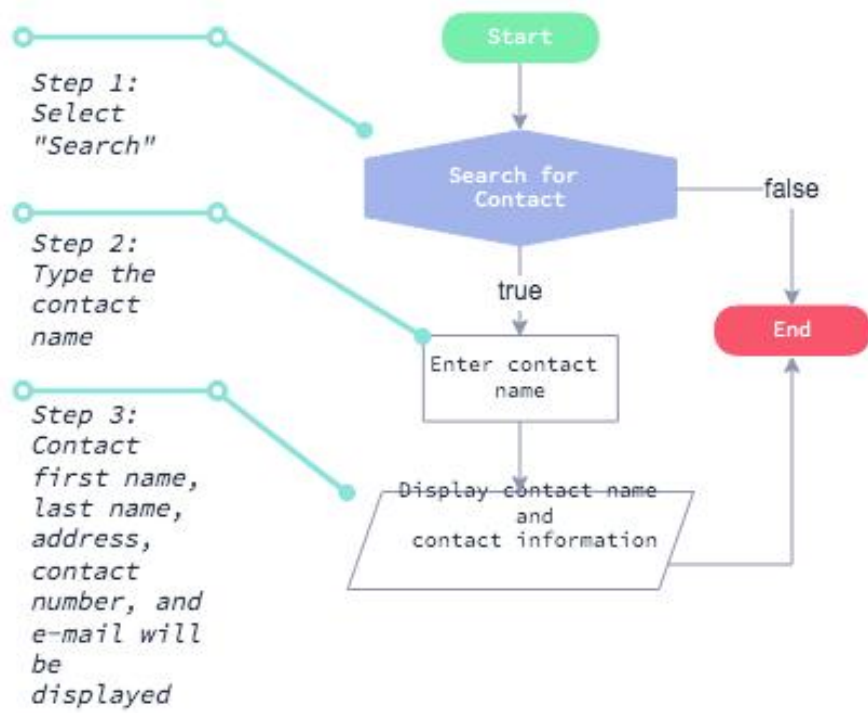


Figure 3: Searching for a contact

The last command that the GD can execute is sorting. The programme will display the "Sort Database" option, which would list the first names of the contacts in alphabetical order. This would enable the user to look through the contacts easily.

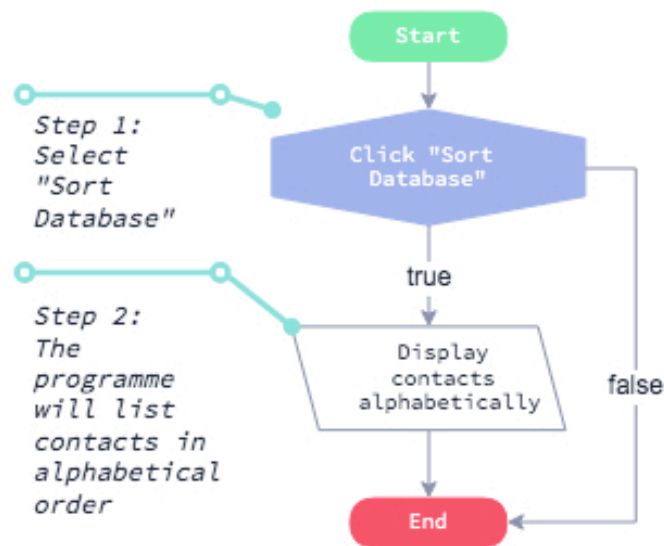


Figure 4: Sorting contacts

Testing

Testing would be crucial in determining whether the commands in the programme will run smoothly. In order to check whether the plan for the Guidance Directory is viable and executable, we will run a Unit Test using unittest. The unittest is an automated Python built-in which is designed to run tests, where an OK result will come out if it works, while Fail (F) if it does not (Pajankar, 2017). Chng (2022) states that Unit Testing examines each software unit to check whether they are working or not. We can therefore use this testing method to run each command in the GD, as shown in the table below.

| | Test will pass if: | Else: |
|--------|---|---|
| Insert | Programme saves contact: 1) First name 2) Last name 3) Address 4) Contact num 5) E-mail Result: OK | Test will Fail (F) if conditions are not met |
| Delete | Programme removes contact: 1) First name 2) Last name 3) Address 4) Contact num 5) E-mail Result: OK | |
| Search | Programme displays contact : 1) First name 2) Last name 3) Address 4) Contact num 5) E-mail Result: OK | |
| Sort | Program arranges contacts in database in alphabetical order Result: OK | |

Figure 5: Test checkpoints using Unit Testing

Conclusion

Python dictionaries, also known as associative arrays, are useful in creating programmes that would execute like a database, such as this Guardian Directory. Since dictionaries are executed through hash functions, adding, deleting, sorting, and searching for keys become straightforward once connected to their related value. In order to make sure that the programme would run smoothly, a Unit Test can be run to check whether the GD can execute their intended function. The CRUD method can guide developers when de-

signing python dictionaries. Since this programme is in its initial stages, further complexities in its execution can be developed in the future, such as completing the Update (U) method in CRUD wherein users can update contact details. Other testing methods, such as Quality Testing, can also be run in order to ensure the efficiency of the programme.

References:

Chng, Z. M. (2022) A Gentle Introduction to Unit Testing in Python. Available from: <https://machinelearningmastery.com/a-gentle-introduction-to-unit-testing-in-python/> [Accessed 27 September 2023].

Chris, K. (2022) CRUD Operations- What is CRUD? Available from: <https://www.freecodecamp.org/news/crud-operations-explained/> [Accessed 27 September 2023].

Codio (2023) Flode. Available from: <https://docs.codio.com/instructors/setupcourses/resources/resourcetools/flode.html#> [Accessed 27 September 2023].

Mastrodomenico, R. (2022) *The Python Book*. New Jersey: Wilden. Available from: <https://learning.oreilly.com/library/view/the-python-book/9781119573319/> [Accessed 26 September 2023].

Mastromatteo, D. (2020) Python Hashtables: Understanding Dictionaries. Available from: <https://thepythoncorner.com/posts/2020-08-21-hash-tables-understanding-dictionaries/> [Accessed 1 October 2023].

Overland, B. (2018) *Python Without Fear*. Boston: Addison-Wesley. Available from: <https://learning.oreilly.com/library/view/python-without-fear/9780134688251/> [Accessed 26 September 2023].

Pajankar, A. (2017) *Python Unit Test Automation: Practical Techniques for Python Developers and Testers*. New York: Apress. Available from: <https://learning.oreilly.com/library/view/python-unit-test/9781484226766/> [Accessed 27 September 2023].

Sturtz, J. (2018) Dictionaries in Python. Available from: <https://realpython.com/python-dicts/> [Accessed 27 September 2023].

Bibliography:

Alwayswin, J. (2017) CRUD System Flowchart. Retrieved from: <https://codeschart.blogspot.com/2017/02/crud-system-flowchart.html> [Accessed 26 September 2023].

Monk, S. (2014) *Raspberry Pi Cookbook*. California: O'Reilly. Available from: <https://learning.oreilly.com/library/view/raspberry-pi-cookbook/9781449365288/> [Accessed 26 September 2023].

Oliveira, B. (2018) *pytest Quick Start Guide*. Birmingham: Packt Publishing. Available from <https://learning.oreilly.com/library/view/pytest-quick-start/9781789347562/>.

Pajankar, A. (2022) *Python Unit Test Automation: Automate, Organize, and Execute Unit Tests in Python*. New York: Apress. Available from: <https://learning.oreilly.com/library/view/python-unit-test/9781484278543/> [Accessed 26 September 2023].

