**Introduction**

Although digital mediums such as the Kindle have made the consumption of literature much more convenient, many people still prefer reading printed books. As the shelves of book-enthusiasts overflow, keeping a digital list is a great way to have a clear overview of one’s collection.

**A brief outline of the application**

The application described in this report can be used to keep a record of books in a book collection.

To use this program, the user must:

1. Turn on their computer and double-click the executable file on the desktop
2. Select the “Add Book” option from the menu to add a new book
3. Select the “Search Collection” option to search for a particular book
4. Select the “Show Collection” option to see the entire collection
5. Select the “A-Z” or “Z-A” option to sort the collection
6. Select the “Delete Book” option to delete a book

**Design of data structure and algorithms**

Insertion

The term “insertion” describes the process of adding data to a list, dictionary, or database.

Since Python can be an object-oriented programming language (Lott & Phillips, 2021), it has preconfigured methods to manipulate a list (Brookshear & Brylow, 2019), which includes both insertion and deletion.

def AddBook():

Ask user for book title and assign it to the variable BookTitle

Rewrite contents of BookTitle in all uppercase and assign to variable AddBook

if AddBook in Collection list:

print “{BookTitle} is already in your collection”

else:

Append AddBook to the Collection list

AscendingSort()

When the AddBook() function is called, the program asks for the title of the book and assigns the input to the variable BookTitle. The data in BookTitle is rewritten in all uppercase letters and assigned to the variable AddBook. Using the .upper() method ensures that the format of each user input is identical, making it easy to search the list. The AddBook variable is added to the end of the Collection list, at which point the AscendingSort() function is called, which automatically sorts the Collection list in ascending order.

Deletion

Contrary to insertion, the term “deletion” describes removing data from a list, dictionary, or database.

def DeleteEntry():

Ask the user for title of book to be removed and assign it to the variable RemoveTitle

Rewrite RemoveTitle in all capitals and assign to RemoveBook

if RemoveBook in Collection:

Ask the user if they are sure that they want to delete the book

if user says yes:

Delete RemoveBook from Collection

print “You have successfully deleted {RemoveBook} from Collection”

elif user says no:

MainMenu()

else:

print “Error” to screen

MainMenu()

else:

print “{RemoveBook} is not in the collection”

When the DeleteEntry() function is called, the program asks for the title of the unwanted book. This input is assigned to the variable RemoveTitle, which is rewritten in all capitals and assigned to the variable RemoveBook. The program then checks if RemoveBook exists in the Collection list. If it does not exist, the program informs the user accordingly.

If the book does exist, the program inquires whether the user is sure about deleting the book. If the user agrees, the program deletes RemoveBook from the Collection list and informs the user that the process was successful. If the user says no, the program returns to the main menu. In case of unexpected input, the program displays an error message.

Sorting

The term “sorting” describes the process of arranging data in a specific order. The sorting in this application will be achieved using a combination between merge sort and insertion sort, known as TimSort (Rathore, 2021).

*Ascending Order:*

def AscendingSort():

Collection.sort()

When the AscendingSort() function is called, the program sorts the Collection list in ascending order, using the built-in Python method sort().

*Descending Order:*

def DescendingSort():

Collection.sort(reverse = True)

When the DescendingSort() function is called, the program sorts the Collection list in descending order. While there might be more efficient ways of sorting lists, these two methods will be sufficient for this application.

Both functions will be nested within another function, called SortRequested().

def SortRequested():

Ask the user if Collection should be sorted in ascending or descending order

if ascending order:

AscendingSort()

print(Collection)

elif descending order:

DescendingSort()

print(Collection)

else:

print “Error - Please select valid option”

When the SortRequested() function is called, the program asks if the list is to be sorted in ascending or descending order. The program will sort the list depending on the input, and print it to the screen. Defining three functions may seem unnecessary, as all instructions could theoretically be included within the SortRequested() function. However, when the AscendingSort() function is called after a new book is added to the list, the list should not be printed to the screen.

Searching

The term “searching” describes the process of looking through a collection of data to check whether a particular block of data exists within that collection. There are various sorting algorithms (GeeksforGeeks.org, 2022), such as sequential/linear search algorithms, as used below:

def CollectionSearch():

Ask user for requested book and assign input to variable RequestedBook

if RequestedBook in Collection:

print “{RequestedBook} is in your collection!”

else:

print “{RequestedBook} is not in your collection!”

When the CollectionSearch() function is called, the program asks for the title of the book and searches through the collection. If the program finds RequestedBook in the list, the user will be informed that the book exists within the collection. If the search is unsuccessful, the program informs the user that the book does not exist in the collection.

**Test plan**

|  |  |
| --- | --- |
| **Checkpoint** | **Expected Outcome** |
| Execute program | Program runs and displays main menu |
| Select “Add Book” and add “Deep Learning using Python” to the collection | Book is successfully added to the collection |
| Select “Show Collection” | List of all books is displayed correctly |
| Select “Search Collection” and search for “Deep Learning using Python” | Program finds the book in the list and informs the user correctly |
| Select “Add Book” and add “Power and Prediction: The Disruptive Economics of Artificial Intelligence” | Book is successfully added to the collection |
| Select “Delete Book” and select “Deep Learning using Python” | Book is successfully removed from collection |

**Conclusion**

The successful implementation of these functions and algorithms would result in a useful tool that would aid collectors and book enthusiasts to keep records of their collections. Due to its simple nature, this program would be easy to use and would not require extensive tutorials and explanations.

**Bibliography**

Lott, F. S. & Phillips, D. (2021) *Python Object-Oriented Programming: Build robust and maintainable object-oriented Python applications and libraries*. 4th Edition. Packed Publishing. Available from: <https://books.google.com.qa/books/about/Python_Object_Oriented_Programming.html?id=J5Y2EAAAQBAJ&printsec=frontcover&source=kp_read_button&hl=en&redir_esc=y#v=onepage&q&f=false>  [Accessed 10 November 2022].

Brookshear, G. & Brylow, D. (2019) *Computer Science: An Overview, Global Edition*. 13th Edition. Pearson International Content. Available from: <https://essexonline.vitalsource.com/reader/books/9781292263441/pageid/279> [Accessed 10 November 2022].

Rathore, A. S. (2021) Sorting Algorithms - With Python. Available from: <https://towardsdatascience.com/sorting-algorithms-with-python-4ec7081d78a1#:~:text=Python's%20default%20sort%20uses%20Tim,merge%20sort%20and%20insertion%20sort>. [Accessed 11 November 2022].

GeeksforGeeks.org (2022) Searching Algorithms. Available from: <https://www.geeksforgeeks.org/searching-algorithms/> [Accessed 11 November 2022].