

**Module Code & Module Title**

**CS4051NI Fundamentals of Computing**

**Assessment Weightage & Type**

**60% Individual Coursework**

**Year and Semester**

**2021 Autumn**

**Student Name: Aadarsha Muni Shakya**

**Group: N1**

**London Met ID: 20049438**

**College ID:NP01NT4S210023**

**Assignment Due Date:**

**Assignment Submission Date: 10th September, 2021**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded*

Table of content

[**1.** **Introduction** 5](#_Toc82151263)

[**2.** **Discussion and Analysis** 6](#_Toc82151264)

[**2.1** **Algorithm** 6](#_Toc82151265)

[**2.2** **Flowchart** 9](#_Toc82151266)

[**2.3** **Pseudocode** 11](#_Toc82151267)

[**2.4** **Data Structures** 25](#_Toc82151268)

[**3** **Program** 29](#_Toc82151269)

[**4** **Testing** 36](#_Toc82151270)

[**4.1** **Test 1** 36](#_Toc82151271)

[**4.2** **Test 2** 37](#_Toc82151272)

[**4.3** **Test 3** 39](#_Toc82151273)

[**4.4** **Test 4** 41](#_Toc82151274)

[**4.5** **Test 5** 43](#_Toc82151275)

[**5** **Conclusion** 46](#_Toc82151276)

[**6** **Appendix** 47](#_Toc82151277)

[Appendix 1: main 47](#_Toc82151278)

[Appendix 2: functions 48](#_Toc82151279)

[**7** **References** 65](#_Toc82151280)

List of Figures

[Figure 1: Flowchart 10](#_Toc82121249)

[Figure 2: Converting text file to 1D list 26](#_Toc82121250)

[Figure 3: Display data of list 26](#_Toc82121251)

[Figure 4: Appending value of list 27](#_Toc82121252)

[Figure 5: int data type 27](#_Toc82121253)

[Figure 6: Float data type 27](#_Toc82121254)

[Figure 7: str data type 28](#_Toc82121255)

[Figure 8: Welcome message and stock 29](#_Toc82121256)

[Figure 9: Borrow a book 30](#_Toc82121257)

[Figure 10: Borrow multiple books 31](#_Toc82121258)

[Figure 11: Bill of borrow 32](#_Toc82121259)

[Figure 12: Return with no fine 33](#_Toc82121260)

[Figure 13: Return bill with no fine 33](#_Toc82121261)

[Figure 14: Return with no fine 34](#_Toc82121262)

[Figure 15: Return bill with no fine 35](#_Toc82121263)

[Figure 16:Exit the program 35](#_Toc82121264)

[Figure 17: Implementation of try, exception 36](#_Toc82121265)

[Figure 18: Assigning negative value as input 37](#_Toc82121266)

[Figure 19:Assigning non-existing value as input 38](#_Toc82121267)

[Figure 20: Borrow process part 1 39](#_Toc82121268)

[Figure 21: Borrow process part 2 39](#_Toc82121269)

[Figure 22: Bill generated after borrow 40](#_Toc82121270)

[Figure 23: Return process part 1 40](#_Toc82121271)

[Figure 24: Return process part 2 41](#_Toc82121272)

[Figure 25: Bill generated after return 42](#_Toc82121273)

[Figure 26: Stock before borrow and return process 43](#_Toc82121274)

[Figure 27: Process of borrowing and returning part 1 43](#_Toc82121275)

[Figure 28: Process of borrowing and returning part 2 44](#_Toc82121276)

[Figure 29: Stock after borrow and return process 44](#_Toc82121277)

List of Tables

[Table 1: Test 1 36](#_Toc82121278)

[Table 2: Test 2 37](#_Toc82121279)

[Table 3: Test 3 38](#_Toc82121280)

[Table 4: Test 4 40](#_Toc82121281)

[Table 5: Test 5 42](#_Toc82121282)

# **Introduction**

In this project, integrated development environment (IDLE) for python is used to code the library management system project. This project is about creating a library management system which allows the use to borrow multiple books and return a single book. Talking about python, Python is a high-level, general-purpose and a very popular programming language (GeeksforFeeks, 2021). Python has features like allowing programming in Object-Oriented, programs are generally smaller than other programming languages, huge collection of standard libraries which will allow the users to do machine learning, GUI applications, Web frameworks and many more.

Throughout the project the main goal was to create a responsive library management system which will increase the quantity when borrowed and decrease when returned. On top of that, a bill is also generated; when books or book is borrowed the bill will have information like name of the book, name of the customer, date and time when borrowed and total cost of book or books. And when a book is returned the bill will have information like name of the book, name of the customer, date and time when returned and if the book is borrowed for more then 10 days fine is also added.

Similarly, the objectives of the project are to learn basic programming, validation, iteration, data structures, file handling and exception handling. From creating program which add two integers to creating a library management system is covered in this module. In-order to code an advanced program, the basics codes should be learned. Similarly, throughout this module basic coding was taught which finally helped in the overall project. Basic programming like printing strings, mathematical operations and many more. For validation, propre use of if-else like nested if-else can be done. Similarly for iteration, loop like for and while can be used. Also, the value of a 2D list cand be extracted using nested loop. Moreover, data structures can help to store data in an organized way which will allow the values to be accessed and modified effectively.

For file handling, the objective of extracting data from text file, writing data in text file were met. the bill generating part of the project uses file handling. Similarly, exception handling is used to avoid the program from crashing. For illustration, if user assigns string value in a variable which takes integer values the program will crash. to avoid this exception handling is done.

# **Discussion and Analysis**

## **Algorithm**

Step 1: Display welcome message

Step 2: Read data.txt file

Step 3: Extract data from file and store it in a list

Step 4: Display list in a table format

Step 5: Initialize total\_cost to 0, book\_name to empty string and flag to true

Step 6: Display 1 for Borrow, 2 for return and 3 to exit

Step 7: Loop while flag equals to true

Step 8: Input value 1, 2 or 3

Step 9: If value equals to 1

Step 10: Input name of the customer

Step 11: Loop while flag is equals to True

Step 12: Input bookID of the book

Step 13: If bookID equals to 1

Step 14: If quantity not equals to 0

Step 15: subtract 1 from quantity of book with ID 1

Step 16: Add cost of the book to total\_cost

Step 17: Add book name with id 1 to book\_name

Step 18: Else, display out of stock message

Step 19: Display y to borrow another book

Step 20: If y, Jump to step 7

Step 21: Else, Create a new text file with unique name

Step 22: Write total\_cose, book\_name, name, date and time

Step 23: break

Step 24: If bookID equals to 2

Step 25: Check if quantity equals to 0

Step 26: If no, subtract 1 from quantity of book with ID 2

Step 27: Add cost of the book to total\_cost

Step 28: Add book name with id 2 to book\_name

Step 29: If yes, display out of stock message

Step 30: Display y to borrow another book

Step 31: If y, Jump to step 7

Step 32: Else, Create a new text file with unique name

Step 33: Write total\_cose, book\_name, name, date and time

Step 34: break

Step 35: If bookID equals to 3

Step 36: Check if quantity equals to 0

Step 37: If no, subtract 1 from quantity of book with ID 3

Step 38: Add cost of the book to total\_cost

Step 39: Add book name with id 3 to book\_name

Step 40: If yes, display out of stock message

Step 41: Display y to borrow another book

Step 42: If y, Jump to step 7

Step 43: Else, Create a new text file with unique name

Step 44: Write total\_cose, book\_name, name, dateand time

Step 45: break

Step 46: If bookID equals to 4

Step 47: Check if quantity equals to 0

Step 48: If no, subtract 1 from quantity of book with ID 4

Step 49: Add cost of the book to total\_cost

Step 50: Add book name with id 4 to book\_name

Step 51: If yes, display out of stock message

Step 52: Display y to borrow another book

Step 53: If y, Jump to step 7

Step 54: Else, Create a new text file with unique name

Step 55: Write total\_cose, book\_name, name, dateand time

Step 56: break

Step 57: If bookID equals to 5

Step 58: Check if quantity equals to 0

Step 59: If no, subtract 1 from quantity of book with ID 5

Step 60: Add book name with id 5 to book\_name

Step 61: Add cost of the book to total\_cost

Step 62: If yes, display out of stock message

Step 63: Display y to borrow another book

Step 64: If y, Jump to step 7

Step 65: Else, Create a new text file with unique name

Step 66: Write total\_cose, book\_name, name, dateand time

Step 67: break

Step 68: If value equals to 2

Step 69: Input the name of the customer

Step 70: Input the book ID

Step 71: Input number of days the book was borrowed

Step 72: If bookID equals to 1

Step 73: Add 1 to quantity of book with ID 1

Step 74: If days greater than 10

Step 75: Fine of $2 per day is added

Step 76: Else fine of $0 is added

Step 77: Create a new text fine with unique name

Step 78: Write customer name, Name of book with bookID 1, fine and date and time

Step 79: If bookID equals to 2

Step 80: Add 1 to quantity of book with ID 2

Step 81: If days greater than 10

Step 82: Fine of $2 per day is added

Step 83: Else fine of $0 is added

Step 84: Create a new text fine with unique name

Step 85: Write customer name, Name of book with bookID 2, fine and date and time

Step 86: If bookID equals to 3

Step 87: Add 1 to quantity of book with ID 3

Step 88: If days greater than 10

Step 89: Fine of $2 per day is added

Step 90: Else fine of $0 is added

Step 91: Create a new text fine with unique name

Step 92: Write customer name, Name of book with bookID 3, fine and date and time

Step 93: If bookID equals to 4

Step 94: Add 1 to quantity of book with ID 4

Step 95: If days greater than 10

Step 96: Fine of $2 per day is added

Step 97: Else fine of $0 is added

Step 98: Create a new text fine with unique name

Step 99: Write customer name, Name of book with bookID 4, fine and date and time

Step 100: If bookID equals to 5

Step 101: Add 1 to quantity of book with ID 5

Step 102: If days greater than 10

Step 103: Fine of $2 per day is added

Step 104: Else fine of $0 is added

Step 105: Create a new text fine with unique name

Step 106: Write customer name, Name of book with bookID 5, fine and date and a time

Step 107: If value equals to 3

Step 108: Exit the program

## **Flowchart**

Flowchart is a graphical representation of an algorithm (Aggarwal, 2020). This will help the users to visually imagine the program and conduct necessary changes. Checking on algorithm for editing the program is a hectic work of flowchart is used which is less hectic.

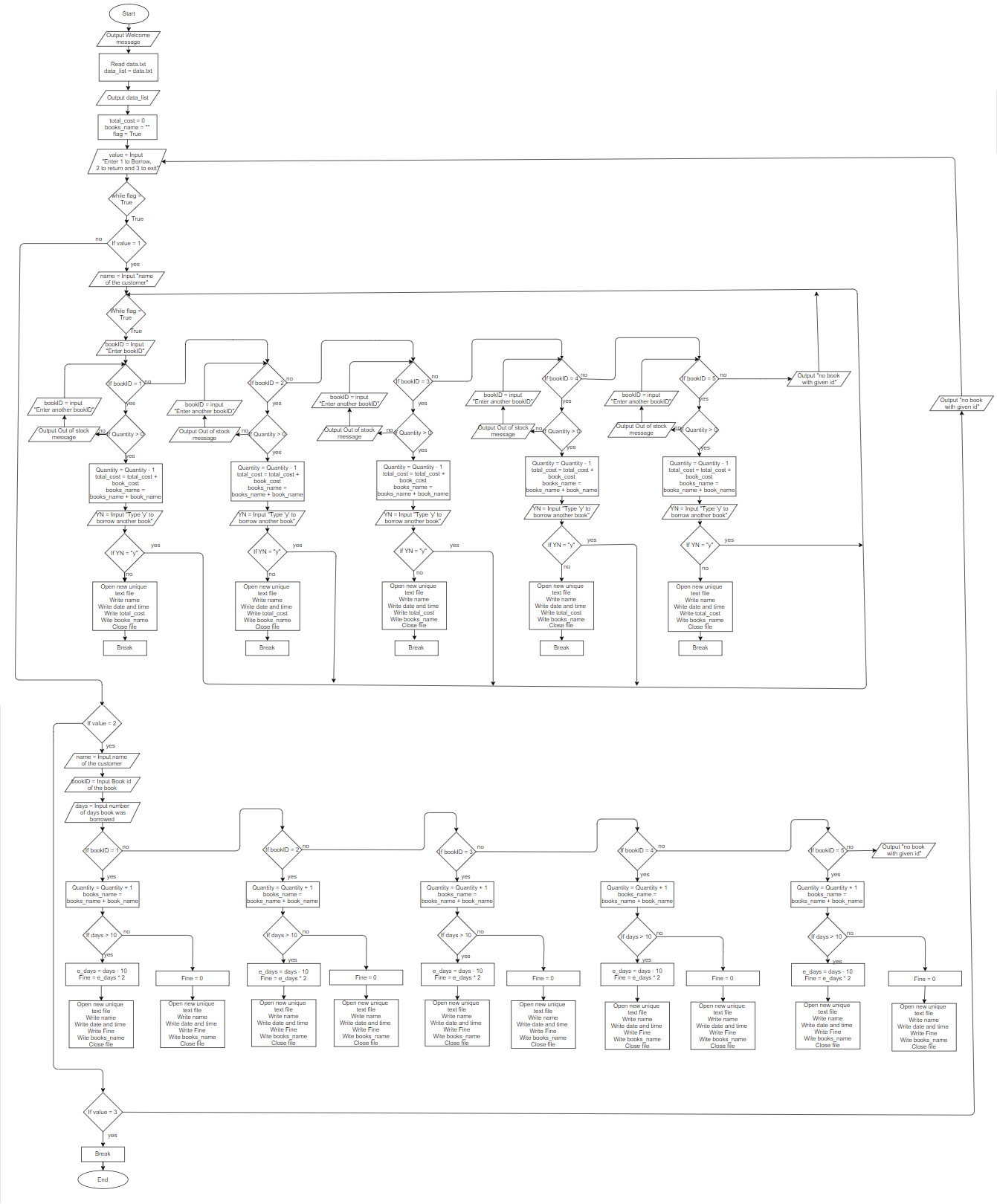


Figure 1: Flowchart

## **Pseudocode**

**MODULE**: function

**FUNCTION** welcome ()

**OUTPUT** “Hello and Welcome to library management system”

**END FUNCTION**

**FUNCTION** valid\_value ()

**OUTPUT** “The entered value is invalid please try again”

**END FUNCTION**

**FUNCTION** ty\_three ()

**OUTPUT** “Thankyou for using my library management system”

**END FUNCTION**

**FUNCTION** now\_borrow ()

**OUTPUT** “You will now borrow a book”

**END FUNCTION**

**FUNCTION** now\_return ()

**OUTPUT** “You will now return a book”

**END FUNCTION**

**FUNCTION** error\_book ()

**OUTPUT** “Please provide a valid Book ID !!!” & “Try again with a different book ID”

**END FUNCTION**

**FUNCTION** n\_avilable\_book ()

**OUTPUT** “Book out of stock !!!”

**END FUNCTION**

**FUNCTION** avilable\_display ()

**OUTPUT** “Book is available”

**END FUNCTION**

**FUNCTION** ty\_alt ()

**OUTPUT** “Thankyou for borrowing book/books” & “Be sure to return it within 10 days or pay the fine”

**END FUNCTION**

**FUNCTION** fine ()

**OUTPUT** “Thankyou for returning the book” & ” You have returned the book late so you will have to pay the fine”

**END FUNCTION**

**FUNCTION** thankyou ()

**OUTPUT** “Thankyou for returning the book” & “If you want to return another book enter 2 down below”

**END FUNCTION**

**FUNCTION** invalid\_dt\_bookid ()

**OUTPUT** “ID should be a integer between 1-5”

**END FUNCTION**

**FUNCTION** invalid\_dt ()

**OUTPUT** “Enter a integer value”

**END FUNCTION**

**FUNCTION** one\_d\_list () RETURNS ARRAY

**DECLARE** dd\_list : ARRAY

**DECLARE** data\_list : ARRAY

**OPENFILE** “data” **FOR READ**

dd\_list ← lines in data

**FOR** i ← 0 TO length(dd\_list)

**FOR** j ← 0 TO length(dd\_list[i])

data\_list ← dd\_list[i][j]

**END FOR**

**END FOR**

**CLOSEFILE** “data”

**RETURN** data\_list

**END FUNCTION**

**FUNCTION** display\_book(data\_list)

**OUTPUT** data\_list[0] & data\_list[1] & data\_list[2] & data\_list[3] & data\_list[4]

**OUTPUT** data\_list[5] & data\_list[6] & data\_list[7] & data\_list[8] & data\_list[9]

**OUTPUT** data\_list[10] & data\_list[11] & data\_list[12] & data\_list[13] & data\_list[14]

**OUTPUT** data\_list[15] & data\_list[16] & data\_list[17] & data\_list[18] & data\_list[19]

**OUTPUT** data\_list[20] & data\_list[21] & data\_list[22] & data\_list[23] & data\_list[24]

**END FUNCTION**

**FUNCTION** continue\_borrow (data\_list, display\_book, total\_cost, name, booksName) RETURNS INTEGER, STRING

add ← TRUE

Book\_cost, bookName ← borrow\_book(data\_list)

total\_cost ← total\_cost + real(book\_cost)

booksName ← booksName & bookName

**WHILE** add = TRUE

YN ← **INPUT** “If tou want to borrow a another book type 'y', else type any onthe word or letter :”

**IF** YN = “y” **THEN**

Book\_cost, bookName ← borrow\_book(data\_list)

total\_cost ← total\_cost + real(book\_cost)

booksName ← booksName & bookName

**ELSE**

ty\_alt()

**BREAK**

**END IF**

**END WHILE**

**RETURN** total\_cost, booksName

**END FUNCTION**

**FUNCTION** borrow\_book (data\_list) RETURNS INTEGER, STRING

flag ← TRUE

BookID ← **INPUT** “Enter the book Id of the book you want to borrow:”

**WHILE** flag = TRUE

**IF** BookID = 1 **THEN**

qnt ← integer(data\_list[3])

**IF** qnt > 0 **THEN**

**CALL** available\_display()

**CALL** remove\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

price ← **CALL** cost(data\_list,bookID)

bookName ← **CALL** book\_name (data\_list, bookID)

**BREAK**

**ELSE**

n\_available\_book()

BookID ← **INPUT** “Enter the book Id of a another book you want to borrow:”

**END IF**

**ELSE IF** BookID = 2 **THEN**

qnt ← integer(data\_list[8])

**IF** qnt > 0 **THEN**

**CALL** available\_display()

**CALL** remove\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

price ← **CALL** cost(data\_list,bookID)

bookName ← **CALL** book\_name (data\_list, bookID)

**BREAK**

**ELSE**

n\_available\_book()

BookID ← INPUT “Enter the book Id of a another book you want to borrow:”

**END IF**

**ELSE IF** BookID = 3 **THEN**

qnt ← integer(data\_list[13])

**IF** qnt > 0 **THEN**

**CALL** available\_display()

**CALL** remove\_book (bookID, qnt, data\_list)

**CALL** display\_book(data\_list)

price ← **CALL** cost(data\_list,bookID)

bookName ← **CALL** book\_name (data\_list , bookID)

**BREAK**

**ELSE**

n\_available\_book()

BookID ← **INPUT** “Enter the book Id of a another book you want to borrow:”

**END IF**

**ELSE IF** BookID = 4 **THEN**

qnt ← integer(data\_list[18])

**IF** qnt > 0 **THEN**

**CALL** available\_display()

**CALL** remove\_book ( bookID, qnt, data\_list)

**CALL** display\_book(data\_list)

price ← **CALL** cost(data\_list,bookID)

bookName ← **CALL** book\_name (data\_list, bookID)

**BREAK**

**ELSE**

n\_available\_book()

BookID ← **INPUT** “Enter the book Id of a another book you want to borrow:”

**END IF**

**ELSE IF** BookID = 5 **THEN**

qnt ← integer(data\_list[23])

**IF** qnt > 0 **THEN**

**CALL** available\_display()

**CALL** remove\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

price ← **CALL** cost(data\_list,bookID)

bookName ← **CALL** book\_name(data\_list, bookID)

**BREAK**

**ELSE**

n\_available\_book()

BookID ← **INPUT** “Enter the book Id of a another book you want to borrow:”

**END IF**

**ELSE**

**CALL** error\_book()

**END IF**

**END WHILE**

**RETURN** price, bookName

**END FUNCTION**

**FUNCTION** remove\_book(bookID, quantity,data\_list)

qnt1 ←data\_list[3]

qnt2 ←data\_list[8]

qnt3 ←data\_list[13]

qnt4 ←data\_list[18]

qnt5 ←data\_list[23]

**OPENFILE** “data” **FOR WRITE**

**IF** BookID = 1 **THEN**

qnt1 ← integer(data\_list[3]) - 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & string(qnt1) & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 2 **THEN**

qnt2 ← integer(data\_list[8]) - 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1 & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & string(qnt2) & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 3 **THEN**

qnt3 ← integer(data\_list[13]) - 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1 & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & string(qnt3) & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 4 **THEN**

qnt4 ← integer(data\_list[18]) - 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1)& ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & string(qnt4) & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 5 **THEN**

qnt5 ← integer(data\_list[23]) - 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1 & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & string(qnt5) & ",$14.00"

**END IF**

**CLOSEFILE** “data”

fnl\_qnt ← quantity – 1

**IF** BookID = 1 **THEN**

data\_list[3] ← string(fnl\_qnt)

**ELSE IF** BookID = 2 **THEN**

data\_list[8] ← string(fnl\_qnt)

**ELSE IF** BookID = 3 **THEN**

data\_list[13] ← string(fnl\_qnt)

**ELSE IF** BookID = 4 **THEN**

data\_list[18] ← string(fnl\_qnt)

**ELSE IF** BookID = 5 **THEN**

data\_list[23] ← string(fnl\_qnt)

**END IF**

**END FUNCTION**

**FUNCTION** cost(data\_list, bookID) RETURNS STRING

**IF** bookID = 1 **THEN**

price ← data\_list[4]

**ELSE IF** bookID = 2 **THEN**

price ← data\_list[9]

**ELSE IF** bookID = 3 **THEN**

price ← data\_list[14]

**ELSE IF** bookID = 4 **THEN**

price ← data\_list[19]

**ELSE IF** bookID = 5 **THEN**

price ← data\_list[24]

**END IF**

**RETURN** price

**END FUNCTION**

**FUNCTION** book\_name(data\_list, bookID) RETURNS STRING

**IF** bookID = 1 **THEN**

bookName ← data\_list[1]

**ELSE IF** bookID = 2 **THEN**

bookName ← data\_list[6]

**ELSE IF** bookID = 3 **THEN**

bookName ← data\_list[11]

**ELSE IF** bookID = 4 **THEN**

bookName ← data\_list[16]

**ELSE IF** bookID = 5 **THEN**

bookName ← data\_list[21]

**END IF**

**RETURN** bookName

**END FUNCTION**

**FUNCTION** b\_bill(total\_cost,name,booksName)

dnt ← Current DateTime

min ← str(Current Minute)

sec ← str(Current Second)

msec ← str(Current Microsecond)

fileName ← name & min & sec & msec

**OPENFILE** fileName **FOR WRITE**

**WRITEFILE** “Name of the Customer: ” & name

**WRITEFILE** “Price of the book/books borrowed: $” & string(total\_cost)

**WRITEFILE** “Date and Time of borrow:” & string(dnt)

**WRITEFILE** “Name of the book/books borrowed: ” & booksName

**CLOSEFILE** filename

**END FUNCTION**

**FUNCTION** return\_book(data\_list, display\_book, name)

flag ← TRUE

BookID ← **INPUT** “Enter the book Id of the book you want to borrow:”

days← **INPUT** “Enter the number of days you borrowed the book: ”

**CALL** fine\_yn(days,data\_list,name,bookID)

**WHILE** flag = TRUE

**IF** BookID = 1 **THEN**

qnt ← integer(data\_list[3])

**CALL** add\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

**BREAK**

**ELSE IF** BookID = 2 **THEN**

qnt ← integer(data\_list[8])

**CALL** add\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

**BREAK**

**ELSE IF** BookID = 3 **THEN**

qnt ← integer(data\_list[13])

**CALL** add\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

**BREAK**

**ELSE IF** BookID = 4 **THEN**

qnt ← integer(data\_list[18])

**CALL** add\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

**BREAK**

**ELSE IF** BookID = 5 **THEN**

qnt ← integer(data\_list[23])

**CALL** add\_book(bookID,qnt,data\_list)

**CALL** display\_book(data\_list)

**BREAK**

**ELSE**

**CALL** error\_book()

**BREAK**

**END IF**

**END WHILE**

**END FUNCTION**

**FUNCTION** add\_book(bookID, quantity, data\_list)

qnt1 ←data\_list[3]

qnt2 ←data\_list[8]

qnt3 ←data\_list[13]

qnt4 ←data\_list[18]

qnt5 ←data\_list[23]

**OPENFILE** “data” **FOR WRITE**

**IF** BookID = 1 **THEN**

qnt1 ← integer(data\_list[3]) + 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & string(qnt1) & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 2 **THEN**

qnt2 ← integer(data\_list[8]) + 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1 & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & string(qnt2) & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 3 **THEN**

qnt3 ← integer(data\_list[13]) + 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1 & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & string(qnt3) & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 4 **THEN**

qnt4 ← integer(data\_list[18]) + 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1)& ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & string(qnt4) & ",$16.00\n5,Invinsible Man,Ralph Ellison," & qnt5 & ",$14.00"

**ELSE IF** BookID = 5 **THEN**

qnt5 ← integer(data\_list[23]) + 1

**WRITEFILE** "1,Nineteen Eighty-Four,George Orwell," & qnt1 & ",$9.99\n2,To Kill a Mockingbird,Harper Lee," & qnt2 & ",$17.99\n3,The Catcher in the Rye,J.D.Salinger," & qnt3 & ",$16.99\n4,Beloved,Toni Morrison," & qnt4 & ",$16.00\n5,Invinsible Man,Ralph Ellison," & string(qnt5) & ",$14.00"

**END IF**

**CLOSEFILE** “data”

fnl\_qnt ← quantity + 1

**IF** BookID = 1 **THEN**

data\_list[3] ← string(fnl\_qnt)

**ELSE IF** BookID = 2 **THEN**

data\_list[8] ← string(fnl\_qnt)

**ELSE IF** BookID = 3 **THEN**

data\_list[13] ← string(fnl\_qnt)

**ELSE IF** BookID = 4 **THEN**

data\_list[18] ← string(fnl\_qnt)

**ELSE IF** BookID = 5 **THEN**

data\_list[23] ← string(fnl\_qnt)

**END IF**

**END FUNCTION**

**FUNCTION** fine\_yn(days, data\_list, name, bookID)

**IF** days > 10 **THEN**

**CALL** r\_f\_bill(name,data\_list,bookID,days)

**CALL** fine()

**ELSE**

**CALL** r\_nf\_bill(name,data\_list,bookID)

**CALL** thankyou()

**END IF**

**END FUNCTION**

**FUNCTION** r\_f\_bill(name, data\_list, bookID, days)

bookName ← **CALL** book\_name(data\_list, bookID)

fine\_ame ← (days - 10) \* 2

dnt ← Current DateTime

min ← str(Current Minute)

sec ← str(Current Second)

msec ← str(Current Microsecond)

fileName ← name & min & sec & msec

**OPENFILE** fileName FOR WRITE

**WRITEFILE** “Name of the Customer: ” & name

**WRITEFILE** “fine: $” & string(fine\_ame)

**WRITEFILE** “Date and Time of borrow:” & string(dnt)

**WRITEFILE** “Name of the book borrowed: ” & bookName

**CLOSEFILE** filename

**END FUNCTION**

**FUNCTION** r\_nf\_bill(name, data\_list, bookID)

bookName ← **CALL** book\_name(data\_list, bookID)

fine\_ame ← 0

dnt ← Current DateTime

min ← str(Current Minute)

sec ← str(Current Second)

msec ← str(Current Microsecond)

fileName ← name & min & sec & msec

**OPENFILE** fileName **FOR WRITE**

**WRITEFILE** “Name of the Customer: ” & name

**WRITEFILE** “fine: $” & string(fine\_ame)

**WRITEFILE** “Date and Time of borrow:” & string(dnt)

**WRITEFILE** “Name of the book borrowed: ” & bookName

**CLOSEFILE** filename

**ENDFUNCTION**

**ENDMODULE**

**MODULE** main

**IMPORT** function

**CALL** welcome()

D\_list ← **CALL** functions.one\_d\_list()

**CALL** functions.display\_book(D\_list)

**DECLARE** total\_cost : INTEGER

**DECLARE** books\_name : STRING

**DECLARE** flag : BOOLEAN

total\_cost ← 0

books\_name ← ""

flag ← TRUE

**WHILE** flag = TRUE

**OUTPUT** “Enter '1' to borrow a book, Enter '2' to return a book, Enter '3' to exit”

value ← **INPUT** “Please enter a value: ”

**IF** value = 1 **THEN**

**CALL** functions.now\_borrow()

name ← **INPUT** “Enter the name of the person:”

t\_cost, Book\_name = **CALL** continue\_borrow ( D\_list, functions.display\_book, total\_cost, name, books\_name)

**CALL** functions.b\_bill(t\_cost,name,Book\_name)

**ELSE IF** value = 2 **THEN**

**CALL** functions.now\_return()

r\_name ← **INPUT** “Enter the name of the person:”

**CALL** functions.return\_book()

**ELSE IF** value = 3 **THEN**

flag ← FALSE

**CALL** functions.ty\_three()

**BREAK**

**ELSE**

**CALL** functions.valid\_value()

**END IF**

**END WHILE**

**END MODULE**

## **Data Structures**

Data Structure is a collection of data type which is stored in an organized way which will allow the values to be accessed and modified effectively (McDonnell, 2019). In this project array list is used to store data, int is used to convert string number to integer and float is used to store integer values which are not whole numbers. Similarly, str is used to convert other data type to string. Other data structures like dictionary can also be used but array allows the appending of values which makes the work easy.

In array data structure, each data is assigned at an index of an array and another data is assigned at the next index and so on (Busbee, 2018). In *Figure 2* given down below the data from text file is extracted and converted into a 1D list. At first, while extracting the data is formed in a 2D list and finally the 2D list is converted to 1D list.

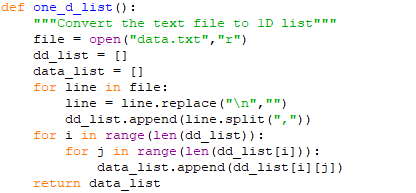


Figure 2: Converting text file to 1D list

Using array list the data can be displayed in an effective way. For example, in *Figure 3* down below the 1D list is being used to display the data in a tabular format.

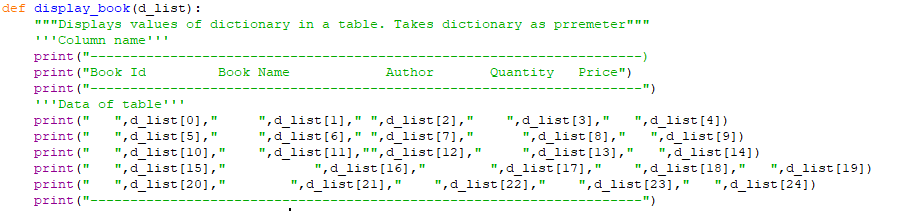


Figure 3: Display data of list

Also, appending data is very easy. For example, in *Figure* *4* down below the variable qnt1 is easily changed. This will allow the programmer to append data whenever they want.

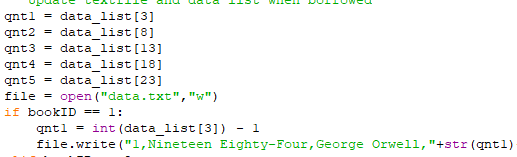


Figure 4: Appending value of list

For int data type, in this project it is used to convert string numbers to integer value. Down below in *Figure 5* the element of data\_list is converted to integer for further arithmetic operations.

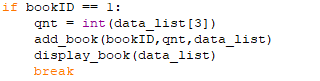


Figure 5: int data type

Similarly, float data type, in the project it is used to calculate total cost (For visual representation look for *Figure 6*). The reason behind using this data type is the cost of the books are not whole numbers. So, in order to add floating-point numbers float data type should be used.

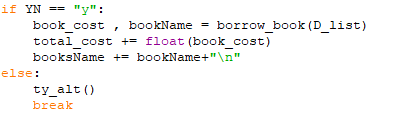


Figure 6: Float data type

For str data type, any other data type can be converted to string. In this project this data type is used to convert date type, float type, integer type to string in order to write in the text file (for visual representation look for *Figure 7*). This is very important because other data type cannot be written in the text file.

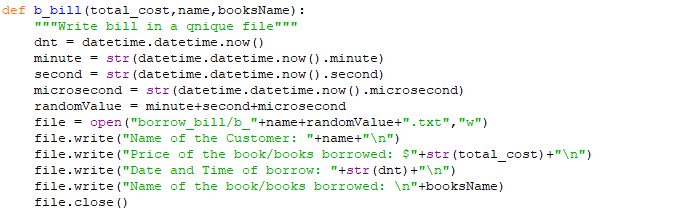


Figure 7: str data type

In other data structure like dictionary, a key is used to uniquely identify the values assigned to the key, the key can be in any order so this is called unordered collection of key-value pairs (Yildirim, 2021). Data dictionary is also known as associative arrays. While using this data structure the values cannot be changed which will cause trouble in this project because quantity must be reduced when borrowed.

# **Program**

When the project is runed a welcome message is displayed. Along with the message the details of the books like book ID, book name, author, quantity and price are displayed in a table format. Also, a message is displayed which tells the users to enter ‘1’ to borrow, enter ‘2’ to return and enter ’3’ to exit the program. (Look for *Figure 8*)

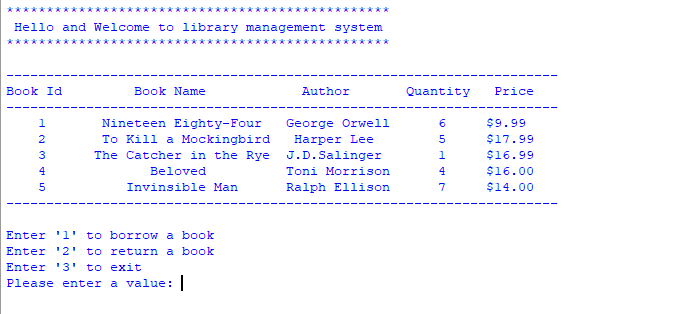


Figure 8: Welcome message and stock

For borrow, when ‘1’ is entered by the users a note is displayed which says ‘You will now borrow a book’. After this the users are supposed to enter their name. Then the users enter the book ID of the book. After assigning book ID the program checks whether the book is available or not if no error message is displayed. Else, an available message and the updated table of information about the book is displayed. If the users want to borrow another book they can type ‘y’ else type anything they want. (Look for *Figure 9*)

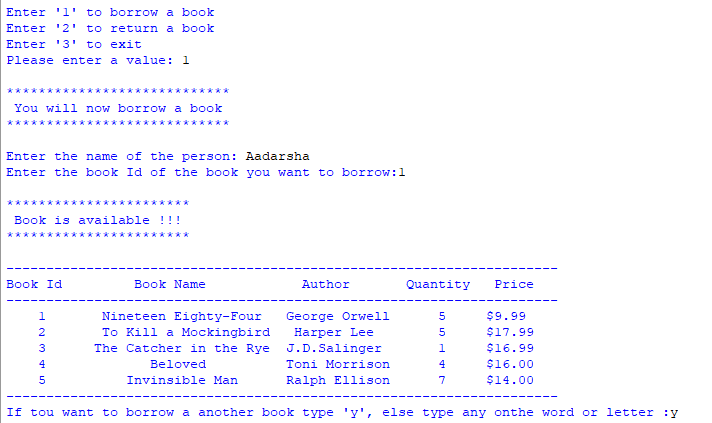


Figure 9: Borrow a book

If the user borrows another book, the users must assign the book ID of the other book they want to borrow. Again, the book available message and updated table is displayed. When any other value except ‘y’ is assigned the borrow process comes to the end and the bill is generated. (Look for *Figure 10*)

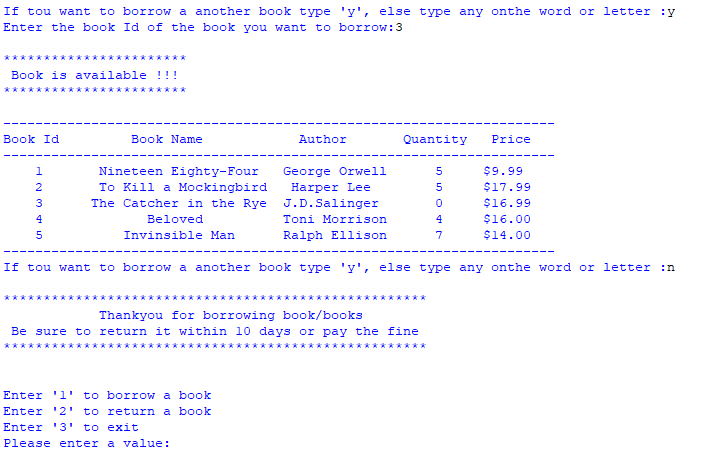


Figure 10: Borrow multiple books

When a borrow process is complete a bill is generated with a unique name using the costumers name and current minute, second and microsecond. In the bill, borrower name, total cost of books, date and time and book names are written. (Look for *Figure 11*)

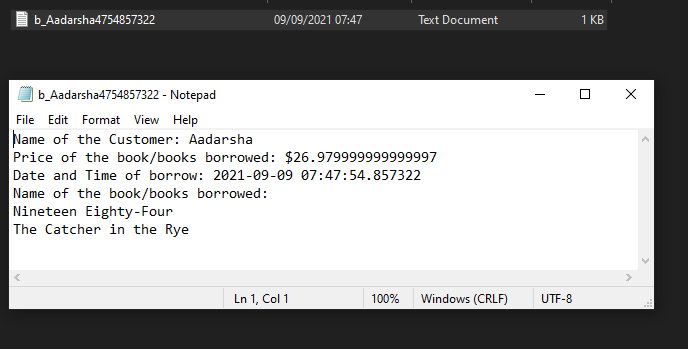


Figure 11: Bill of borrow

For return, when ‘2’ is entered by the users a note is displayed which says ‘You will now return a book’. After this the users are supposed to enter their name. Then the users enter the book ID of the book and the number of days the book was borrowed. After this the program will check if the given book ID is present in the stock or not. If book is not available an error message is displayed. Else the number of days the book was borrowed will be compared to 10. If days is less than 10 a thank you message is displayed and the bill is generated. (Look for *Figure 12*)

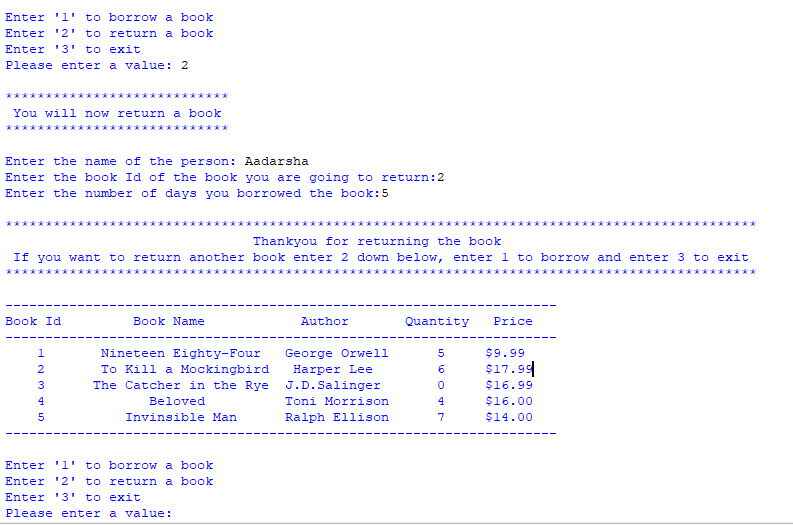


Figure 12: Return with no fine

When the bill is generated after returning the book in time details like name of the customer, fine of $0, date and time and the name of the book is writen. (Look for *Figure 13*)

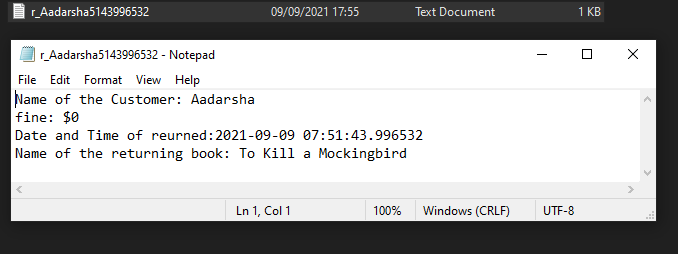


Figure 13: Return bill with no fine

When the book is returned late, thank you message and an additional message saying ‘you have returned the book late so you will have to pay the fine’ is displayed and the bill is generated. (Look for *Figure 14*)

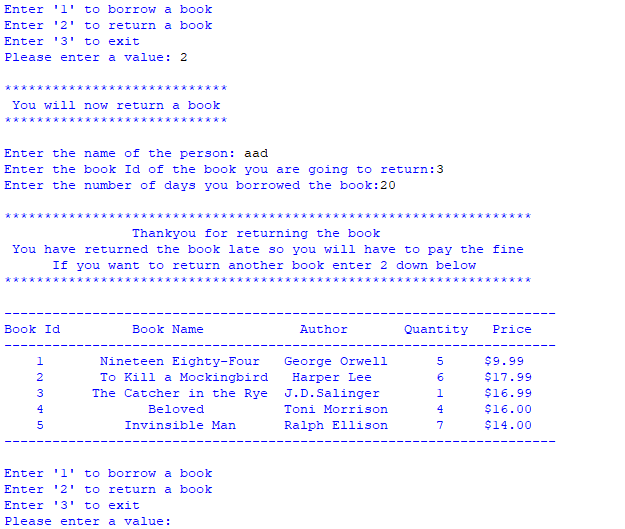


Figure 14: Return with no fine

When the bill is generated after returning the book late details like name of the customer, fine of $2 per day, date and time and the name of the book is write. (Look for *Figure 15*)

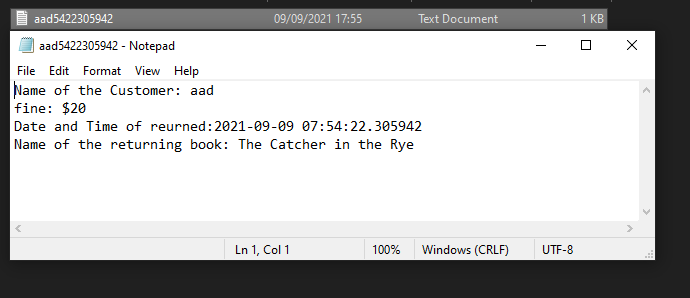


Figure 15: Return bill with no fine

To exit the program the value assigned should be ‘3’. If yes, a thank you message is displayed. (Look for *Figure 16*)

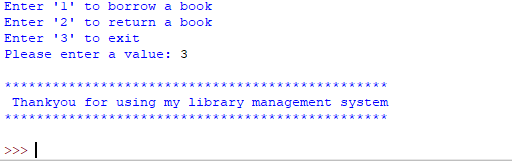


Figure 16:Exit the program

# **Testing**

## **Test 1**

Table 1: Test 1

|  |  |
| --- | --- |
| Objectives | Show implementation of try, exception |
| Action | Provide invalid input while assigning value  Assign “xyz” |
| Expected Results | Error message saying “Enter a integer value” |
| Actual Result | “Enter a integer value” is displayed |
| Conclusion | Test successful. |

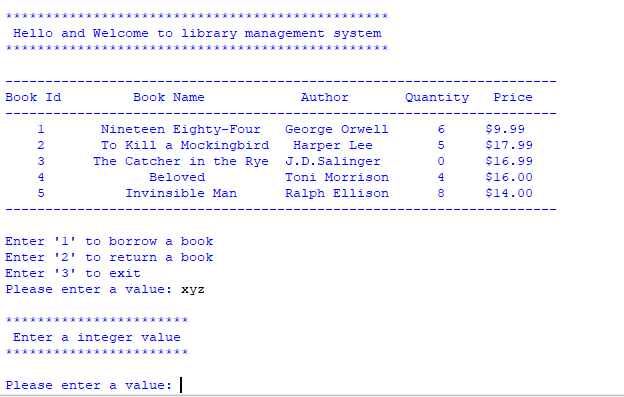


Figure 17: Implementation of try, exception

## **Test 2**

Table 2: Test 2

|  |  |
| --- | --- |
| Objectives | Selection borrow and return option |
| Action | Provide negative value as input  Provide non-existing value as input  Assign -100  Assign 6 |
| Expected Results | Error Message saying “The enterer value is invalid please try again” Should be displayed |
| Actual Result | “The enterer value is invalid please try again” is displayed |
| Conclusion | Test successful. |

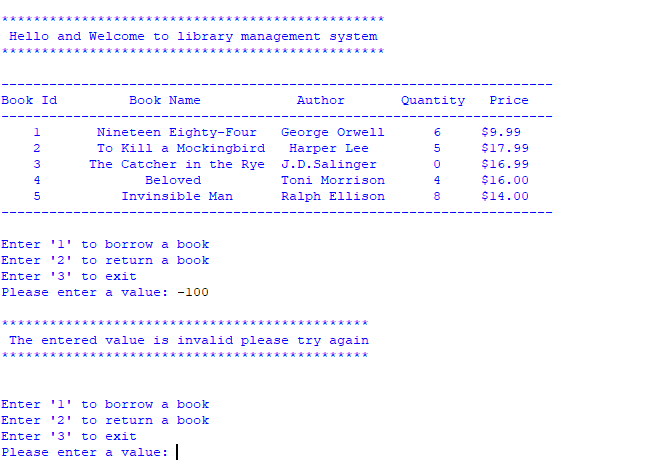
****

Figure 18: Assigning negative value as input

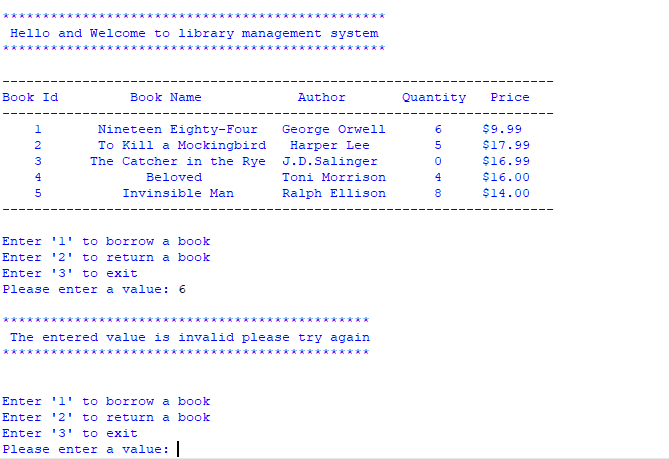
****

Figure 19:Assigning non-existing value as input

## **Test 3**

Table 3: Test 3

|  |  |
| --- | --- |
| Objectives | File generation of borrow |
| Action | Do a complete borrow process  Borrow a book with book ID 1  Show quantity decrease in shell  Show new generated txt file |
| Expected Results | The book should be borrowed successfully |
| Actual Result | The book was borrowed successfully |
| Conclusion | Test successful. |

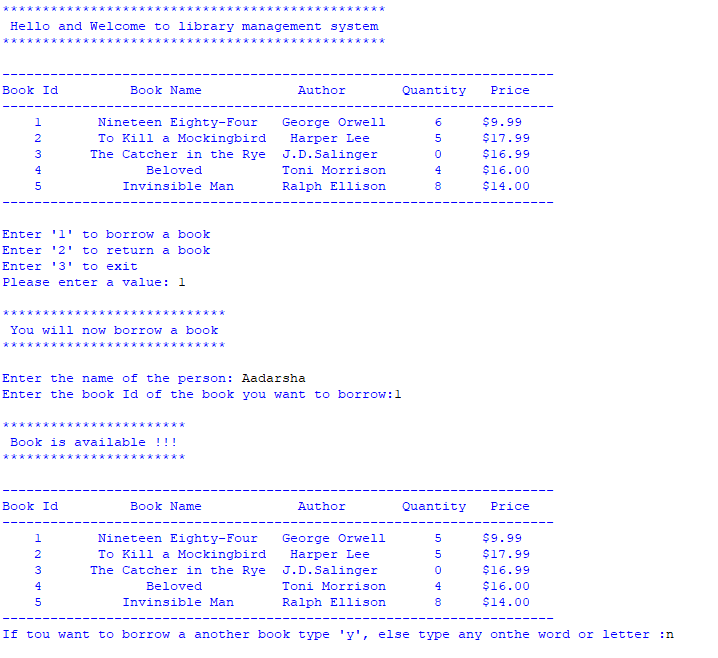


Figure 20: Borrow process part 1

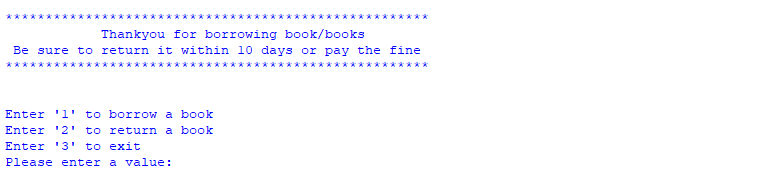


Figure 21: Borrow process part 2

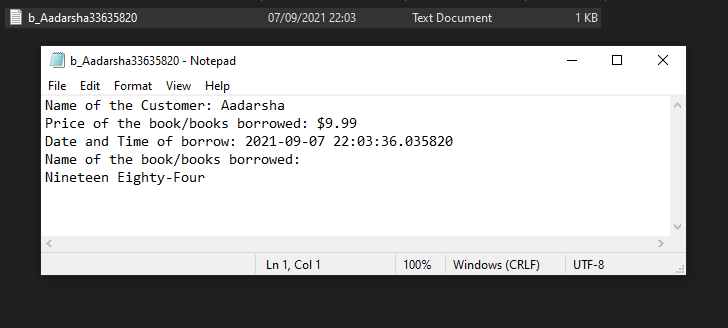


Figure 22: Bill generated after borrow

## **Test 4**

Table 4: Test 4

|  |  |
| --- | --- |
| Objectives | File generation of return |
| Action | Do a complete return process  Return a book with book ID 3  Show quantity increase in shell  Show new generated txt file |
| Expected Results | The book will be returned successfully |
| Actual Result | The book was returned successfully |
| Conclusion | Test successful. |

Figure 23: Return process part 1

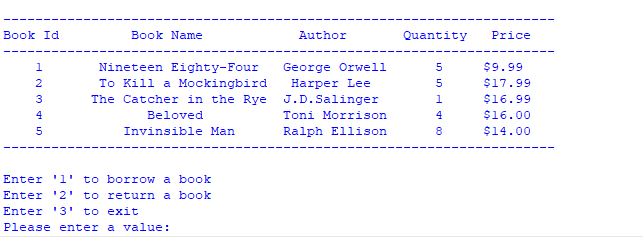
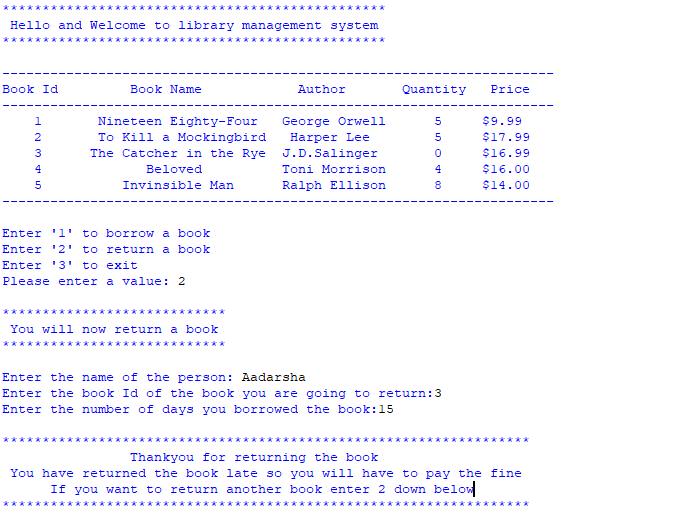


Figure 24: Return process part 2

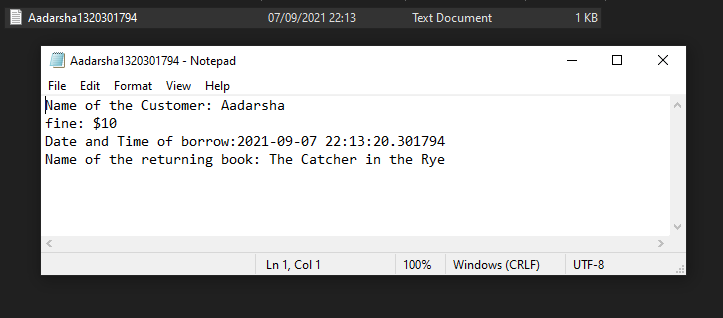


Figure 25: Bill generated after return

## **Test 5**

Table 5: Test 5

|  |  |
| --- | --- |
| Objectives | Show the update in stock |
| Action | Show stock before borrowing and returning a book  Borrow book with book ID 5  Return book with book ID 1  Show stock after borrowing and returning a book |
| Expected Results | Quantity must decrease when borrowed and must increase when returned |
| Actual Result | Quantity is decrease when borrowed and increased when returned |
| Conclusion | Test successful. |

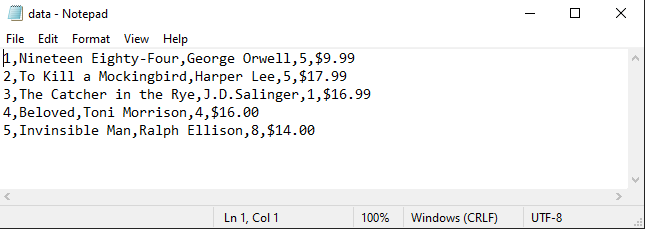


Figure 26: Stock before borrow and return process

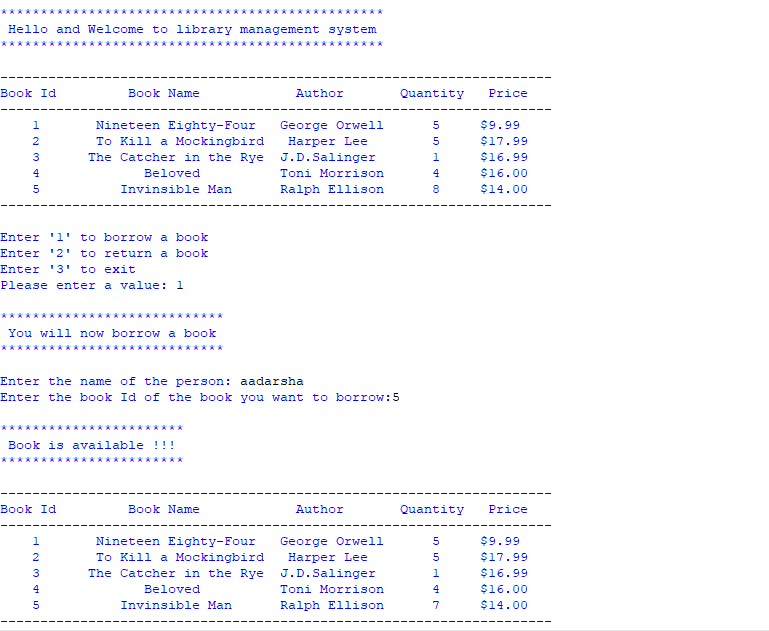


Figure 27: Process of borrowing and returning part 1

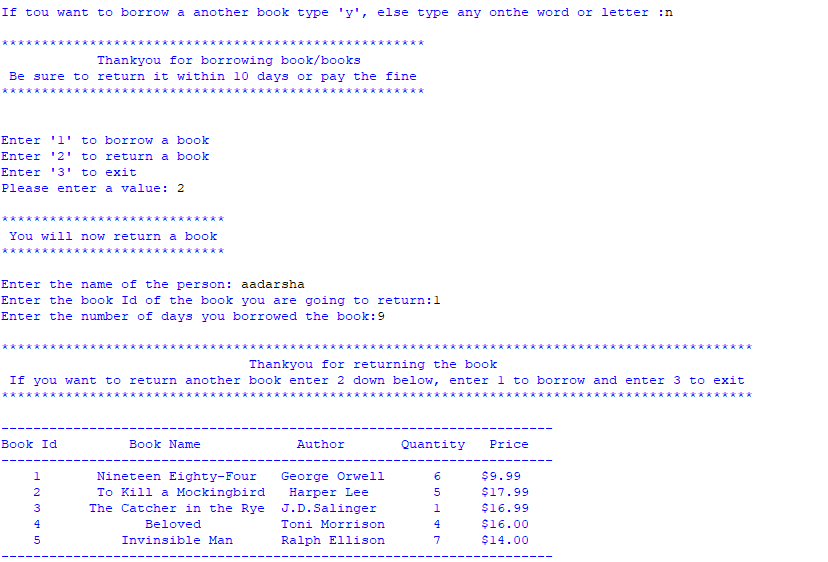


Figure 28: Process of borrowing and returning part 2

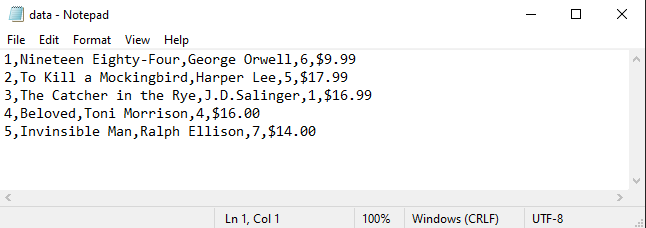


Figure 29: Stock after borrow and return process

# **Conclusion**

For conclusion, throughout this module a new programming language python was taught. Other things like writing algorithm, flowchart and pseudocode were taught. In python basic programming, validation, iteration, data structures, file handling and exception handling were covered. These small programming concepts were used to create an advanced and responsive system. Without those concepts building an advanced system is impossible. For illustration, if a programmer is building a program to add two numbers. And if the user assigns a string value the whole program will crash. To avoid the program from crashing the programmer should know the use of exception handling. Which will lead to a proper program.

Moreover, file handling was very important for this project. This is because all work like reading and extracting data from a text file, writing in a new text file with unique name were covered by this topic. If a programmer lacks these skills the extracting and editing of data and bill generation will not work properly. For example, while extracting and editing the quantity the format of other values should be same. If not, the whole text file containing data will be deleted. To avoid this problem programmers should use proper file handling.

Previously, algorithm, flowchart and pseudocode of small basic program was made. But for this project algorithm, flowchart and pseudocode of an advanced system is made. Therefore, this was all for the library management system project.

# **Appendix**

## Appendix 1: main

import functions

functions.welcome() #Calling welcome function

D\_list = functions.one\_d\_list() #Calling one\_d\_list function and assigning the returned value in D\_list

functions.display\_book(D\_list) #Calling display\_book function

total\_cost = 0 # Initializing total\_cost

books\_name = "" # Initializing books\_name

flag = True # Initializing flag

while flag == True:

print("\nEnter '1' to borrow a book\nEnter '2' to return a book\nEnter '3' to exit") #Print statement to make the users aware about the work done by the program

valid = True

'''Try except'''

while valid == True:

try:

value = int(input("Please enter a value: ")) #Taking value from the users

break

except:

functions.invalid\_dt() #Calling invalid\_dt function

if value == 1:

'''For borrow'''

functions.now\_borrow() #Calling now\_borrow function

name = input("Enter the name of the person: ") #Taking name as input from users

t\_cost, Book\_name = functions.continue\_borrow(D\_list,functions.display\_book,total\_cost,name,books\_name) #Calling continue\_borrow function amd assigning the returned value in t\_cost, Book\_name

functions.b\_bill(t\_cost,name,Book\_name) #Calling b\_bill function

elif value == 2:

'''For return'''

functions.now\_return() #Calling now\_return function

r\_name = input("Enter the name of the person: ") #Taking r\_name as input from users

functions.return\_book(D\_list,functions.display\_book,r\_name) #Calling return\_book function

elif value == 3:

'''For exit'''

flag == False

functions.ty\_three() #Calling ty\_three function

break

else:

'''if book ID is not 1,2 or 3'''

functions.valid\_value() #Calling valid\_value function

## Appendix 2: functions

import datetime

def welcome():

"""Function to Dislpay welcome message"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Hello and Welcome to library management system\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def valid\_value():

"""Function to Dislpay enter valid value message"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n The entered value is invalid please try again\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def ty\_three():

"""Function to Dislpay thank you message"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Thankyou for using my library management system\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def now\_borrow():

"""Function to Dislpay now you will borrow message"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n You will now borrow a book\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def now\_return():

""" Function toDislpay now you will return message"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n You will now return a book\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def error\_book():

"""Function to Display invalid book id"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Please provide a valid Book ID !!!\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Try again with a different book ID\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def n\_avilable\_book():

"""Function to Display out of stock"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Book out of stock !!!\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def avilable\_display():

"""Function to Dislpay Book is available """

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Book is available !!!\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def ty\_alt():

"""Function to Display thank you message after borrowing a book"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Thankyou for borrowing book/books\n Be sure to return it within 10 days or pay the fine\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def fine():

"""Function to Display thank you message after returning a book late"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Thankyou for returning the book\n You have returned the book late so you will have to pay the fine\n If you want to return another book enter 2 down below\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def thankyou():

"""Function to Display thank you message after returning a book in time"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Thankyou for returning the book\n If you want to return another book enter 2 down below, enter 1 to borrow and enter 3 to exit\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def invalid\_dt\_bookid():

"""Function which Displayes ID should be a integer between 1-5 when execption occer while assigning book ID"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n ID should be a integer between 1-5\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def invalid\_dt():

"""Function which Displayes when Enter a integer value when execption occer while assigning value"""

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Enter a integer value\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") #print statement

def one\_d\_list():

"""Convert the text file to 1D list"""

file = open("data.txt","r")

'''Initializing lists'''

dd\_list = []

data\_list = []

for line in file:

line = line.replace("\n","") #Replacing '\n' with empty siring

dd\_list.append(line.split(",")) #Converting to 2D list

'''converting 2D list to 1D list'''

for i in range(len(dd\_list)):

for j in range(len(dd\_list[i])):

data\_list.append(dd\_list[i][j])

return data\_list

def display\_book(d\_list):

"""Displays values of dictionary in a table. Takes dictionary as prremeter"""

'''Column name'''

print("---------------------------------------------------------------------") #Border line

print("Book Id Book Name Author Quantity Price") #Column name

print("---------------------------------------------------------------------")#Border line

'''Data of table'''

print(" ",d\_list[0]," ",d\_list[1]," ",d\_list[2]," ",d\_list[3]," ",d\_list[4]) #details of with book ID 1

print(" ",d\_list[5]," ",d\_list[6]," ",d\_list[7]," ",d\_list[8]," ",d\_list[9]) #details of with book ID 2

print(" ",d\_list[10]," ",d\_list[11],"",d\_list[12]," ",d\_list[13]," ",d\_list[14]) #details of with book ID 3

print(" ",d\_list[15]," ",d\_list[16]," ",d\_list[17]," ",d\_list[18]," ",d\_list[19]) #details of with book ID 5

print(" ",d\_list[20]," ",d\_list[21]," ",d\_list[22]," ",d\_list[23]," ",d\_list[24])#details of with book ID 5

print("---------------------------------------------------------------------")#Border line

def continue\_borrow(D\_list,display\_book,total\_cost,name,booksName):

"""To calculate total cost and all the books borrowed"""

add = True #Initializing add

book\_cost , bookName = borrow\_book(D\_list) #Calling borrow\_book function and assigning the returned value to book\_cost and bookName

total\_cost += float(book\_cost) #Adding book\_cost of borrowed book and assigning to total\_cost

booksName += bookName+"\n" #Joining bookName of borrowed book and assigning it to booksName

while add == True:

YN = input("If tou want to borrow a another book type 'y', else type any onthe word or letter :")#Taking YN as input from the users

if YN == "y":

book\_cost , bookName = borrow\_book(D\_list) #Calling borrow\_book function and assigning the returned value to book\_cost and bookName

total\_cost += float(book\_cost) #Adding book\_cost of borrowed book to total\_cost

booksName += bookName+"\n" #Joining bookName of borrowed book to booksName

else:

ty\_alt() #alling ty\_alt function

break

return total\_cost, booksName

def borrow\_book(data\_list):

"""Function which decreases quantity, displays books, returns price and bookName"""

'''Initializing price, bookName, flag, vld, vld1'''

price = 0

bookName = ""

flag = True

vld = True

vld1 = True

'''Exception handling'''

while vld == True:

try:

bookID = int(input("Enter the book Id of the book you want to borrow:")) #Taking book ID as input from the users

break

except:

invalid\_dt\_bookid()#Calling invalid\_dt function

while flag == True:

if bookID == 1:

qnt = int(data\_list[3]) #Assigning the 3 index data\_list to qnt

if qnt > 0:

avilable\_display() #Calling avalable\_display function

remove\_book(bookID,qnt,data\_list) #Calling remove\_book function

display\_book(data\_list) #Calling display\_book functin

price = cost(data\_list,bookID) #Calling cost function

bookName = book\_name(data\_list,bookID) #Calling book\_name function

break

else:

n\_avilable\_book() #Calling n\_avilable\_book function

while vld1 == True:

try:

bookID = int(input("Enter the book Id of a another book you want to borrow:")) #Taking book ID as input from the users

break

except:

invalid\_dt\_bookid()#Calling invalid\_dt function

elif bookID == 2:

qnt = int(data\_list[8]) #Assigning the 8 index data\_list to qnt

if qnt > 0:

avilable\_display()#Calling avalable\_display function

remove\_book(bookID,qnt,data\_list) #Calling remove\_book function

display\_book(data\_list) #Calling display\_book functin

price = cost(data\_list,bookID) #Calling cost function

bookName = book\_name(data\_list,bookID)#Calling book\_name function

break

else:

n\_avilable\_book() #Calling n\_avilable\_book function

while vld1 == True:

try:

bookID = int(input("Enter the book Id of a another book you want to borrow:"))#Taking book ID as input from the users

break

except:

invalid\_dt\_bookid() #Calling invalid\_dt function

elif bookID == 3:

qnt = int(data\_list[13])#Assigning the 13 index data\_list to qnt

if qnt > 0:

avilable\_display()#Calling avalable\_display function

remove\_book(bookID,qnt,data\_list) #Calling remove\_book function

display\_book(data\_list)#Calling display\_book functin

price = cost(data\_list,bookID)#Calling cost function

bookName = book\_name(data\_list,bookID)#Calling book\_name function

break

else:

n\_avilable\_book() #Calling n\_avilable\_book function

while vld1 == True:

try:

bookID = int(input("Enter the book Id of a another book you want to borrow:")) #Taking book ID as input from the users

break

except:

invalid\_dt\_bookid() #Calling invalid\_dt function

elif bookID == 4:

qnt = int(data\_list[18])#Assigning the 18 index data\_list to qnt

if qnt > 0:

avilable\_display()#Calling avalable\_display function

remove\_book(bookID,qnt,data\_list)#Calling remove\_book function

display\_book(data\_list)#Calling display\_book functin

price = cost(data\_list,bookID) #Calling cost function

bookName = book\_name(data\_list,bookID) #Calling book\_name function

break

else:

n\_avilable\_book() #Calling n\_avilable\_book function

while vld1 == True:

try:

bookID = int(input("Enter the book Id of a another book you want to borrow:")) #Taking book ID as input from the users

break

except:

invalid\_dt\_bookid() #Calling invalid\_dt function

elif bookID == 5:

qnt = int(data\_list[23])#Assigning the 23 index data\_list to qnt

if qnt > 0:

avilable\_display() #Calling avalable\_display function

remove\_book(bookID,qnt,data\_list)#Calling remove\_book function

display\_book(data\_list) #Calling display\_book functin

price = cost(data\_list,bookID) #Calling cost function

bookName = book\_name(data\_list,bookID)#Calling book\_name function

break

else:

n\_avilable\_book() #Calling n\_avilable\_book function

while vld1 == True:

try:

bookID = int(input("Enter the book Id of a another book you want to borrow:"))#Taking book ID as input from the users

break

except:

invalid\_dt\_bookid()#Calling invalid\_dt function

else:

error\_book()

break

return price , bookName

def remove\_book(bookID,quantity,data\_list):

"""Update textfile and data list when borrowed"""

'''Initializing qnt1,qnt2,qnt3,qnt4,qnt4'''

qnt1 = data\_list[3]

qnt2 = data\_list[8]

qnt3 = data\_list[13]

qnt4 = data\_list[18]

qnt5 = data\_list[23]

file = open("data.txt","w")

if bookID == 1:

qnt1 = int(data\_list[3]) - 1 #Updating variable qnt1

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+str(qnt1)+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 2:

qnt2 = int(data\_list[8]) - 1 #Updating variable qnt2

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+str(qnt2)+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 3:

qnt3 = int(data\_list[13]) - 1 #Updating variable qnt3

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+str(qnt3)+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 4:

qnt4 = int(data\_list[18]) - 1 #Updating variable qnt4

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+str(qnt4)+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 5:

qnt5 = int(data\_list[23]) - 1 #Updating variable qnt5

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+str(qnt5)+",$14.00")

file.close()

fnl\_qnt = quantity-1 #Calculating quantity afrer borrow

if bookID == 1:

data\_list[3] = str(fnl\_qnt)#Assigning fnl\_qnt to the 3 index data\_list

elif bookID == 2:

data\_list[8] = str(fnl\_qnt)#Assigning fnl\_qnt to the 3 index data\_list

elif bookID == 3:

data\_list[13] = str(fnl\_qnt)#Assigning fnl\_qnt to the 3 index data\_list

elif bookID == 4:

data\_list[18] = str(fnl\_qnt)#Assigning fnl\_qnt to the 3 index data\_list

elif bookID == 5:

data\_list[23] = str(fnl\_qnt)#Assigning fnl\_qnt to the 3 index data\_list

def cost(data\_list,bookID):

"""Retuens the price of the book borrowed"""

if bookID == 1:

price = data\_list[4] #Assigning the 4 index data\_list to price

price = price.replace("$","") #Replacing $ with empty string

elif bookID == 2:

price = data\_list[9] #Assigning the 9 index data\_list to price

price = price.replace("$","") #Replacing $ with empty string

elif bookID == 3:

price = data\_list[14] #Assigning the 14 index data\_list to price

price = price.replace("$","") #Replacing $ with empty string

elif bookID == 4:

price = data\_list[19]#Assigning the 19 index data\_list to price

price = price.replace("$","") #Replacing $ with empty string

elif bookID == 5:

price = data\_list[24]#Assigning the 24 index data\_list to price

price = price.replace("$","") #Replacing $ with empty string

return price

def book\_name(data\_list,bookID):

"""Returns book name which is borrowed"""

bookName = "" #Initializing bookName

if bookID == 1:

bookName = data\_list[1]#Assigning the 1 index data\_list to bookName

elif bookID == 2:

bookName = data\_list[6]#Assigning the 6 index data\_list to bookName

elif bookID == 3:

bookName = data\_list[11]#Assigning the 11 index data\_list to bookName

elif bookID == 4:

bookName = data\_list[16]#Assigning the 16 index data\_list to bookName

elif bookID == 5:

bookName = data\_list[21]#Assigning the 21 index data\_list to bookName

return bookName

def b\_bill(total\_cost,name,booksName):

"""Write bill in a qnique file"""

dnt = datetime.datetime.now()#Returns current date and time

'''Creatimg unique name for file generation'''

minute = str(datetime.datetime.now().minute)

second = str(datetime.datetime.now().second)

microsecond = str(datetime.datetime.now().microsecond)

randomValue = minute+second+microsecond

'''File handling'''

file = open("borrow\_bill/b\_"+name+randomValue+".txt","w")

file.write("Name of the Customer: "+name+"\n")

file.write("Price of the book/books borrowed: $"+str(total\_cost)+"\n")

file.write("Date and Time of borrow: "+str(dnt)+"\n")

file.write("Name of the book/books borrowed: \n"+booksName)

file.close()

def return\_book(data\_list, display\_book,name):

"""Function to take book ID. If valid ID is entered call other functions else call error function"""

'''initializing boolean variables'''

flag = True

vld = True

vld1 = True

'''Exception handling'''

while vld == True:

try:

bookID = int(input("Enter the book Id of the book you are going to return:")) #Taking bookID as input from users

break

except:

invalid\_dt\_bookid()#Calling invalid\_dt\_bookid

while vld == True:

try:

days = int(input("Enter the number of days you borrowed the book:")) #Taking days as input from users

break

except:

invalid\_dt() #Calling invalid\_dt function

fine\_yn(days,data\_list,name,bookID)#Calling fine\_yn function

while flag == True:

if bookID == 1:

qnt = int(data\_list[3]) #Assigning the 3 index data\_list to qnt

add\_book(bookID,qnt,data\_list) #Calling add\_book function

display\_book(data\_list) #Calling display\_book functin

break

elif bookID == 2:

qnt = int(data\_list[8]) #Assigning the 8 index data\_list to qnt

add\_book(bookID,qnt,data\_list)#Calling add\_book function

display\_book(data\_list)#Calling display\_book functin

break

elif bookID == 3:

qnt = int(data\_list[13])#Assigning the 13 index data\_list to qnt

add\_book(bookID,qnt,data\_list)#Calling add\_book function

display\_book(data\_list)#Calling display\_book functin

break

elif bookID == 4:

qnt = int(data\_list[18])#Assigning the 18 index data\_list to qnt

add\_book(bookID,qnt,data\_list)#Calling add\_book function

display\_book(data\_list)#Calling display\_book functin

break

elif bookID == 5:

qnt = int(data\_list[23]) #Assigning the 23 index data\_list to qnt

add\_book(bookID,qnt,data\_list)#Calling add\_book function

display\_book(data\_list)#Calling display\_book functin

break

else:

error\_book()#Calling error\_book functin

break

def add\_book(bookID,quantity,data\_list):

"""Function to update textfile and data list when borrowed"""

'''Initializing qnt1,qnt2,qnt3,qnt4,qnt4'''

qnt1 = data\_list[3]

qnt2 = data\_list[8]

qnt3 = data\_list[13]

qnt4 = data\_list[18]

qnt5 = data\_list[23]

'''file handling'''

file = open("data.txt","w")

if bookID == 1:

qnt1 = int(data\_list[3]) + 1 #Updating variable qnt1

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+str(qnt1)+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 2:

qnt2 = int(data\_list[8]) + 1 #Updating variable qnt2

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+str(qnt2)+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 3:

qnt3 = int(data\_list[13]) + 1 #Updating variable qnt3

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+str(qnt3)+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 4:

qnt4 = int(data\_list[18]) + 1 #Updating variable qnt4

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+str(qnt4)+",$16.00\n5,Invinsible Man,Ralph Ellison,"+qnt5+",$14.00")

elif bookID == 5:

qnt5 = int(data\_list[23]) + 1 #Updating variable qnt5

'''Writing updated variable in txt file'''

file.write("1,Nineteen Eighty-Four,George Orwell,"+qnt1+",$9.99\n2,To Kill a Mockingbird,Harper Lee,"+qnt2+",$17.99\n3,The Catcher in the Rye,J.D.Salinger,"+qnt3+",$16.99\n4,Beloved,Toni Morrison,"+qnt4+",$16.00\n5,Invinsible Man,Ralph Ellison,"+str(qnt5)+",$14.00")

file.close()

fnl\_qnt = quantity + 1 #Calculating quantity afrer return

if bookID == 1:

data\_list[3] = str(fnl\_qnt)#Assigning fnl\_qnt to the 3 index data\_list

elif bookID == 2:

data\_list[8] = str(fnl\_qnt)#Assigning fnl\_qnt to the 8 index data\_list

elif bookID == 3:

data\_list[13] = str(fnl\_qnt)#Assigning fnl\_qnt to the 13 index data\_list

elif bookID == 4:

data\_list[18] = str(fnl\_qnt)#Assigning fnl\_qnt to the 18 index data\_list

elif bookID == 5:

data\_list[23] = str(fnl\_qnt)#Assigning fnl\_qnt to the 23 index data\_list

def fine\_yn(days,data\_list,name,bookID):

"""Finction to determine fine needs to be payed or not"""

if days > 10:

r\_f\_bill(name,data\_list,bookID,days)#Calling r\_f\_bill function

fine()#Calling fine function

else:

r\_nf\_bill(name,data\_list,bookID) #Calling r\_nf\_bill function

thankyou() #Calling thankyou function

def r\_f\_bill(name,data\_list,bookID,days):

"""Function to write in the txt file if returned late"""

bookName = book\_name(data\_list,bookID) #Calling book\_name function

fine\_amt = (days-10) \* 2 #calculating fine amount

dnt = datetime.datetime.now() #Current returns date and time

'''Creatimg unique name for file generation'''

minute = str(datetime.datetime.now().minute)

second = str(datetime.datetime.now().second)

microsecond = str(datetime.datetime.now().microsecond)

randomValue = minute+second+microsecond

'''File handling'''

file = open("retuen\_bill/"+name+randomValue+".txt","w")

file.write("Name of the Customer: "+name+"\n")

file.write("fine: $"+str(fine\_amt)+"\n")

file.write("Date and Time of return:"+str(dnt)+"\n")

file.write("Name of the returning book: "+bookName)

file.close()

def r\_nf\_bill(name,data\_list,bookID):

"""Function to write in the txt file if returned in time"""

bookName = book\_name(data\_list,bookID)#Calling book\_name function

fine\_amt = 0 # fine is 0 because book is returned in time

dnt = datetime.datetime.now() #Current returns date and time

'''Creatimg unique name for file generation'''

minute = str(datetime.datetime.now().minute)

second = str(datetime.datetime.now().second)

microsecond = str(datetime.datetime.now().microsecond)

randomValue = minute+second+microsecond

'''File handling'''

file = open("retuen\_bill/r\_"+name+randomValue+".txt","w")

file.write("Name of the Customer: "+name+"\n")

file.write("fine: $"+str(fine\_amt)+"\n")

file.write("Date and Time of return:"+str(dnt)+"\n")

file.write("Name of the returning book: "+bookName)

file.close()

# **References**

Aggarwal, N., 2020. *GeeksforGeeks.* [Online]   
Available at: https://www.geeksforgeeks.org/an-introduction-to-flowcharts/  
[Accessed 8 September 2021].

Busbee, K., 2018. *Arrays and Lists.* [Online]   
Available at: https://press.rebus.community/programmingfundamentals/chapter/arrays-and-lists/  
[Accessed 9 September 2021].

GeeksforFeeks, 2021. *Python Programming Language.* [Online]   
Available at: https://www.geeksforgeeks.org/python-programming-language/  
[Accessed 9 September 2021].

McDonnell, M., 2019. *Data Types and Data Structures.* [Online]   
Available at: https://www.integralist.co.uk/posts/data-types-and-data-structures/#data-structures  
[Accessed 9 September 2021].

Yildirim, S., 2021. *4 Must-Know Features of Python Dictionaries.* [Online]   
Available at: https://towardsdatascience.com/4-must-know-features-of-python-dictionaries-d62af8c22fd2  
[Accessed 9 September 2021].