Python training project (2022)

Library Management System

Team members:

1. Ru’a Ayman Mohamed

2. Reham Ibrahim Abdeldayem

3. Sohaila Ibrahim Ramadan

4. Zeyad El-sayed Abdelazim (Team Leader)

5.Ziad Muhammad Mahmoud

6.Zeyad Adel Abdallah.

Project overview:

Description and Functionality:

Libraries range in size from a few shelves to large buildings with a huge collection of knowledge not only books ,but maps ,periodicals, and other data storage sources, so we did a system to make libraries easier to work with libraries are divided into two basic types ,and they are private and public libraries. In this project we're dealing with public libraries, It is a system where college or school libraries can lend books to students through ID student, and the book is returned after 7 days, and helps in saving the book's data and ID student.

Tools:

- **Programs:-**
  - Visual Studio Code.
  - PyCharm
  - DB Browser For SQLite
- **Programming language and libraries :**
  - Python (Programming language).
  - Tkinter.
  - PIL.
  - sqlite3.

GUI :

We have 4 frames :

1. Login page.
2. Home Page.
3. Book Data Frame
4. Student Frame.

Login Page :

Login Frame is the first interface in the system ,login screen is the frame for admin login there are(two labels ,two entries , one button). admin should enter his username and password to login if information is true. by pressing on login ,second frame will appear

- **Code:-**

```
def Login():
    admin_db()
    if USERNAME.get() == "" or PASSWORD.get() == "":
        messagebox.showinfo("Error","Please complete the required field!")
    else:
        cur.execute("SELECT * FROM `Admins` WHERE `username` = ? AND `password` = ?", (USERNAME.get(), PASSWORD.get()))
        if cur.fetchone() is not None:
            app.destroy()
            a=main_lb()
            a.mainloop()

        else:
            messagebox.showinfo("Error","Invalid username or password.")
            USERNAME.set("")
            PASSWORD.set("")
```

```
USERNAME = StringVar()
PASSWORD = StringVar()

lbl_title = Label(cnv,text="ADMIN  LOGIN" , font=('Papyrus', 30,'bold'), bg='black' , fg= 'white' , width=15)
lbl_title.place(x =490 , y = 90)

lbl1 = Label(cnv,text=' Username ' , font=('Papyrus', 15,'bold') , bg='black', fg='white' , bd =4)
lbl1.place(x = 500 ,y= 230)
ent_username = Entry(cnv,textvariable=USERNAME , bg='black', fg='white' , font = (14) , bd=4)
ent_username.place(x=650 ,y = 230)
lbl2 = Label(cnv,text=' Password ' , font=('Papyrus', 15,'bold') , bg='black', fg='white' , bd = 3)
lbl2.place(x= 500 ,y = 330 )
ent_password = Entry(cnv,textvariable=PASSWORD , bg='black', fg='white' , font = (14) , show='*' , bd=4)
ent_password.place(x=650 ,y = 330)

btn_login = Button(cnv,text=" LOGIN ", font=('Papyrus', 15,'bold'),bg='black',fg='white' , width=20,command=Login)
btn_login.place(x=550 , y = 450)
btn_login.bind('<Return>', Login)
```

2. Home Page :-

In this frame there are(two labels(book data and student data)) ,and admin should choose which data he wants. by clicking on book data,3rd frame will appear, And By clicking on student data ,4th frame will appear.

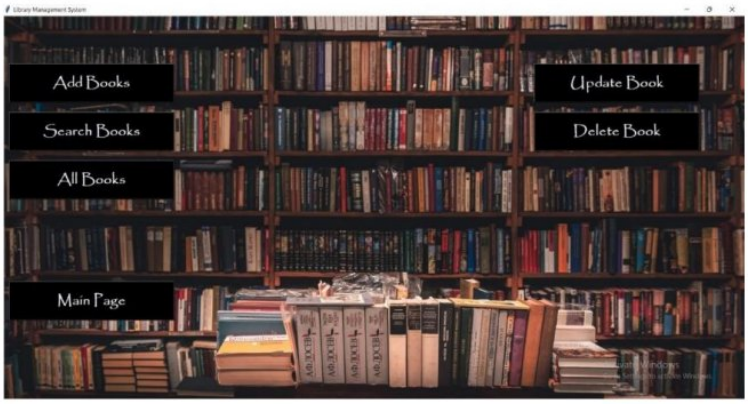
- **Code:-**

```
l1 = Button(self.ph1, text="BOOK DATA", fg="white", bg="black",
            font="Papyrus 22 bold", borderwidth=0, command=self.book)
l1.place(x=200, y=500)
l1 = Button(self.ph1, text="STUDENT DATA", fg="white", bg="black",
            font="Papyrus 22 bold", borderwidth=0, command=self.student)
l1.place(x=900, y=500)
self.wind1.mainloop()
```

Book Data Frame :-

In this frame there are (6 labels ,6 entries and 6 buttons):-

1. Add book for adding new books.
2. Search for a book in the library.
3. Update data of books in the library.
4. Delete books .
5. Showing all books in the library .

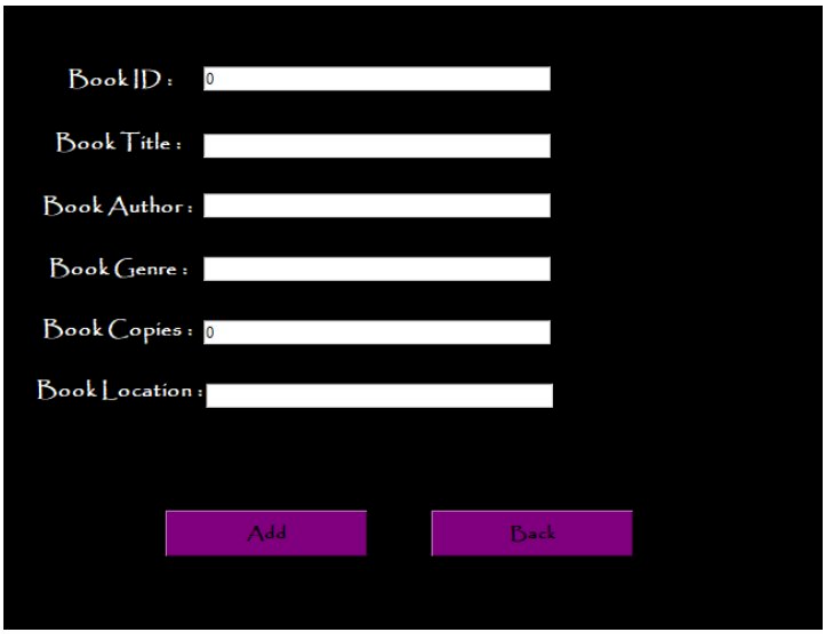


- **Code:-**

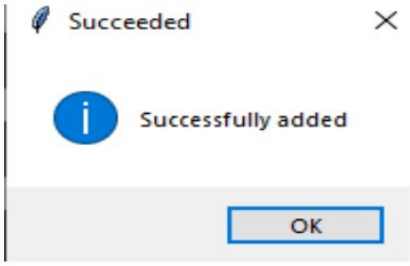
```
def book(self):
    self.ph1.destroy()
    self.ph1 = self.photos(img2)
    btn1 = Button(self.ph1, text="Add Books", bg="black", fg="white",
                  font="Papyrus 22 bold", width=15, padx=10, command=self.addbook)
    btn1.place(x=12, y=100)
    btn2 = Button(self.ph1, text="Search Books", bg="black", fg="white",
                  font="Papyrus 22 bold", width=15, padx=10, command=self.search)
    btn2.place(x=12, y=200)
    btn3 = Button(self.ph1, text="All Books", bg="black", fg="white",
                  font="Papyrus 22 bold", width=15, padx=10, command=self.all_book)
    btn3.place(x=12, y=300)

    btn4 = Button(self.ph1, text="Main Page", bg="black", fg="white",
                  font="Papyrus 22 bold", width=15, padx=10, command=self.mainmenu)
    btn4.place(x=12, y=550)

    btn5 = Button(self.ph1, text="Delete Book", bg="black", fg="white",
                  font="Papyrus 22 bold", width=15, padx=10,command=self.deletebook)
    btn5.place(x=1100, y=200)
    btn6 = Button(self.ph1, text="Update Book", bg="black", fg="white",
                  font="Papyrus 22 bold", width=15, padx=10,command=self.updatebook )
    btn6.place(x=1100, y=100)
```

1. Add Books:-

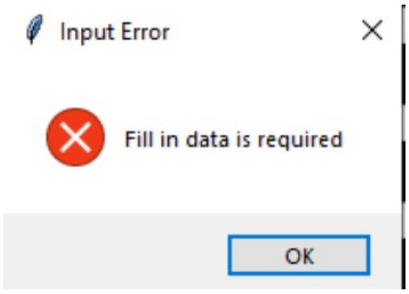
- To insert new Book , When you click here, it appears like this:



- Then insert data and click Add , A message confirming the addition appears, such as:



- If you do not fill in all the data, an error message like this will appear, and in the case of entering the.. zero, also:



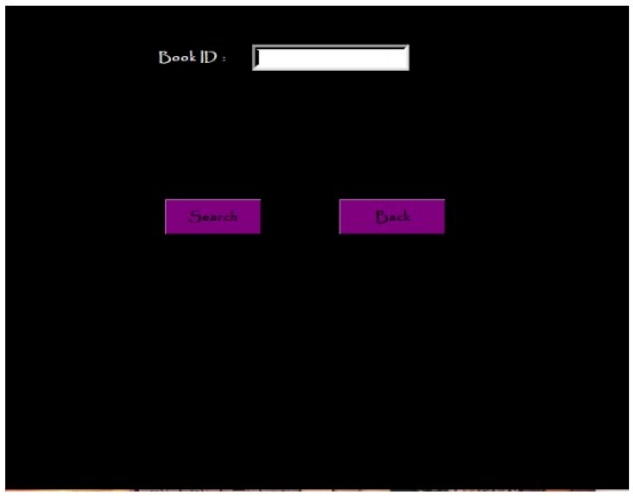
- **Code:**

```
def addbook(self):
    self.a_id = IntVar()  # book id
    # self.a_id.get()
    self.a_title = StringVar()
    self.a_author = StringVar()
    self.a_genre = StringVar()
    self.a_copies = IntVar()
    self.a_loc = StringVar()

    # -------frame for books info------------
    self.f1 = Frame(self.ph1, height=500, width=650, bg='black')
    self.f1.place(x=400, y=100)
    # book id-------
    lb_id = Label(self.f1, text="Book ID : ",
                  font="Papyrus 12 bold", fg="white", bg="black")
    lb_id.place(x=50, y=45)
    e1 = Entry(self.f1, textvariable=self.a_id, width=45, bg="white")
    e1.place(x=160, y=50)
    # book_title-------
    lb_title = Label(self.f1, text="Book Title :",
                  font="Papyrus 12 bold", fg="white", bg="black")
    lb_title.place(x=40, y=95)
    e2 = Entry(self.f1, textvariable=self.a_title, width=45, bg="white")
    e2.place(x=160, y=103)
    # book author--------
    lb_au = Label(self.f1, text="Book Author :",
                  font="Papyrus 12 bold", fg="white", bg="black")
    lb_au.place(x=30, y=145)
    e3 = Entry(self.f1, textvariable=self.a_author, width=45, bg="white")
    e3.place(x=160, y=150)
```

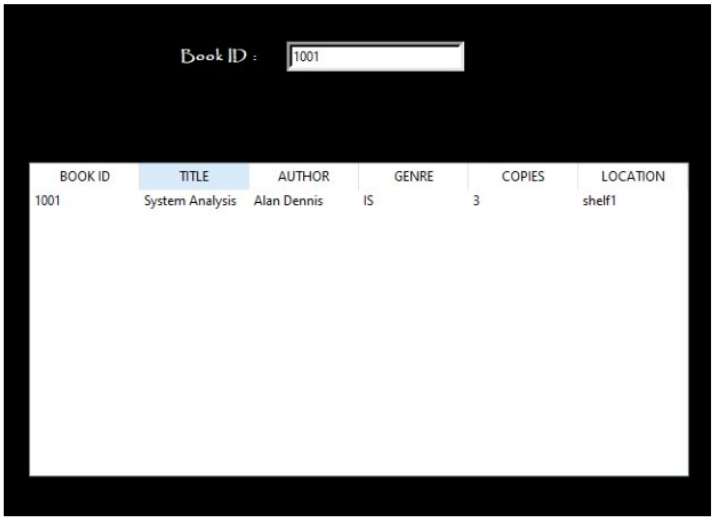
```
    # book genre--------
    lb_ge = Label(self.f1, text="Book Genre :",
                  font="Papyrus 12 bold", fg="white", bg="black")
    lb_ge.place(x=35, y=194)
    e4 = Entry(self.f1, textvariable=self.a_genre, width=45, bg="white")
    e4.place(x=160, y=200)
    # book copies--------
    lb_copy = Label(self.f1, text="Book Copies : ",
                  font="Papyrus 12 bold", fg="white", bg="black")
    lb_copy.place(x=30, y=242)
    e5 = Entry(self.f1, textvariable=self.a_copies, width=45, bg="white")
    e5.place(x=160, y=250)
    # book location--------
    lb_loc = Label(self.f1, text="Book Location :",
                  font="Papyrus 12 bold", fg="white", bg="black")
    lb_loc.place(x=25, y=290)
    e6 = Entry(self.f1, textvariable=self.a_loc, width=45, bg="white")
    e6.place(x=162, y=300)

    btn_add2 = Button(self.f1, text="Add", fg="black", bg="purple",
                  width=15, font="Papyrus 10 bold", command=self.AddBook_db)
    btn_add2.place(x=130, y=400)
    btn_back = Button(self.f1, text="Back", fg="black", bg="purple",
                  width=15, font="Papyrus 10 bold", command=self.rm)
    btn_back.place(x=340, y=400)
```

2. Search :

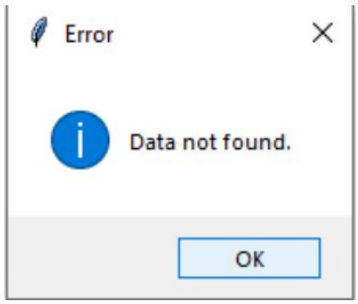
Search for a book in the library, Search By Book ID:



- When you insert right id :



- When you insert incorrect id:



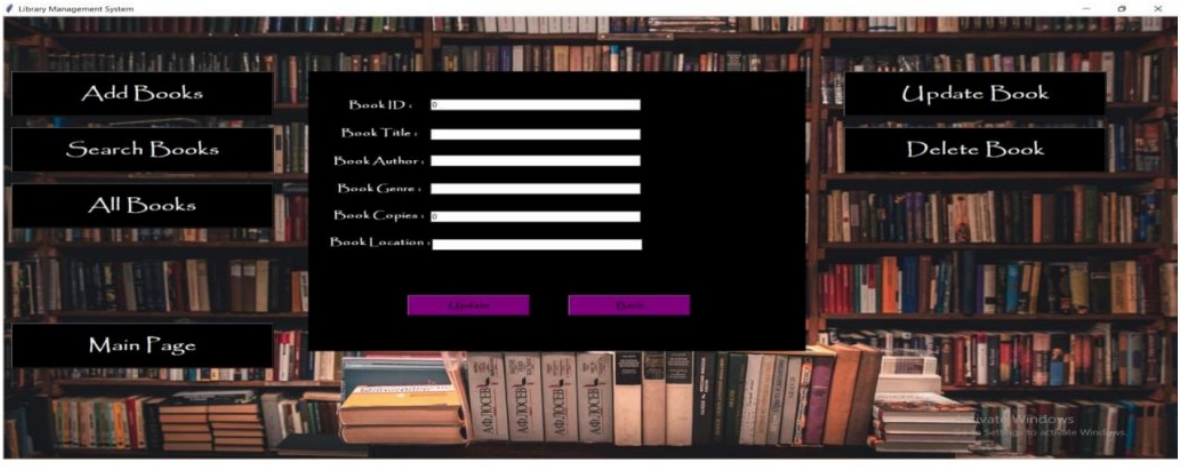
- Code :

```
def search(self):

    self.a_id = IntVar()
    self.f1 = Frame(self.ph1, height=500, width=650, bg='black')
    self.f1.place(x=400, y=100)
    l1 = Label(self.f1, text='Book ID : ', font=(
        'Papyrus 10 bold'), bd=2, fg='white', bg='black')
    l1.place(x=160, y=40)
    e1 = Entry(self.f1, width=25, bd=5, bg='white',
               fg='black', textvariable=self.a_id)
    e1.place(x=260, y=40)
    b1 = Button(self.f1, text='Search', bg='purple', fg="black", font='Papyrus 10 bold',width=9, bd=2, command=self.search1)
    b1.place(x=170, y=200)
    b2 = Button(self.f1, text='Back', bg='purple', fg="black",
                font='Papyrus 10 bold', width=10, bd=2, command=self.rm)
    b2.place(x=350, y=200)
```

1. Update:

In our system we can update a book in the library by entering book id and other data .

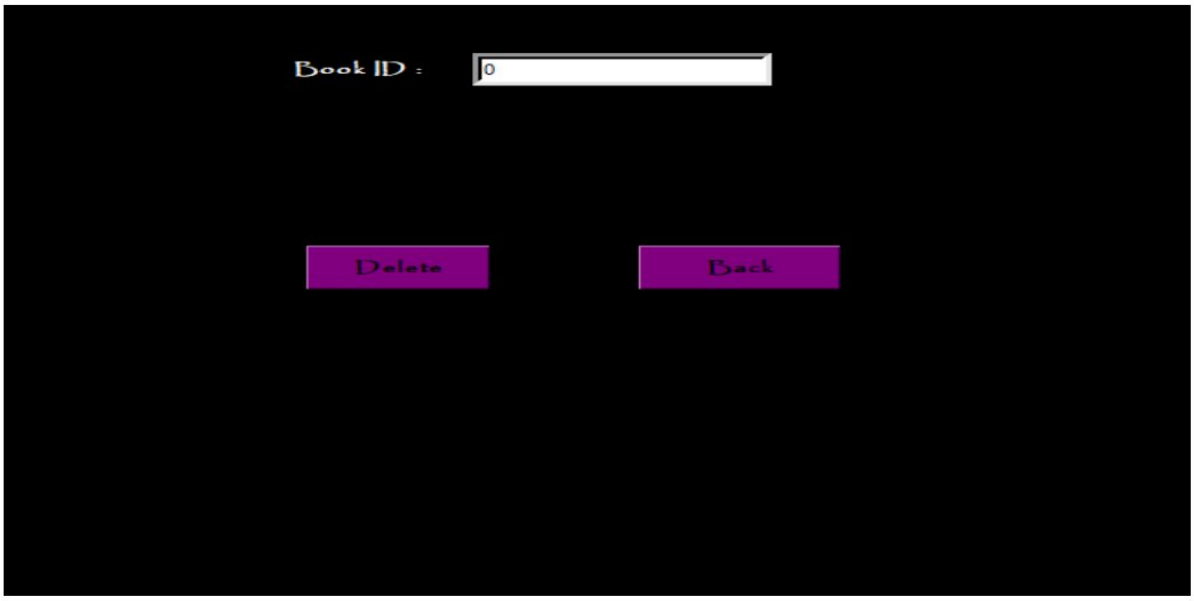
For example ,we can update number of copies for a book



Enter new data and click Update , An update confirmation message appears like:

2. Delete:

In our system we can delete a book from the library by entering book id and clicking delete, Deleted By Book Id .



An Delete confirmation message appears like

- **Code :**

```
def deletebook(self):

    self.a_id = IntVar()
    self.f1 = Frame(self.ph1, height=500, width=650, bg='black')
    self.f1.place(x=400, y=100)
    l1 = Label(self.f1, text='Book ID : ', font=(
        'Papyrus 10 bold'), bd=2, fg='white', bg='black')
    l1.place(x=160, y=40)
    e1 = Entry(self.f1, width=25, bd=5, bg='white',
               fg='black', textvariable=self.a_id)
    e1.place(x=260, y=40)
    b1 = Button(self.f1, text='Delete', bg='purple', fg="black", font='Papyrus 10 bold',width=9, bd=2, command=self.DeleteBook_db)
    b1.place(x=170, y=200)
    b2 = Button(self.f1, text='Back', bg='purple', fg="black",
                font='Papyrus 10 bold', width=10, bd=2, command=self.rm)
    b2.place(x=350, y=200)
```

Show All :

This table shows all data of books ,Book id , title , author , genre , copies and location . We used for this Treeview to show data of books in a table.

- **code:**

```
def create_tree(self, plc, lists):
    self.tree = ttk.Treeview(
        plc, height=13, column=(lists), show='headings')
    n = 0
    while n is not len(lists):
        self.tree.heading("#" + str(n + 1), text=lists[n])
        self.tree.column("" + lists[n], width=100)
        n = n + 1
    return self.tree

def all_book(self):
    self.f1 = Frame(self.ph1, bg="black", height=500, width=650)
    self.f1.place(x=400, y=100)
    self.lists = ("BOOK ID", "TITLE", "AUTHOR",
                  "GENRE", "COPIES", "LOCATION")
    self.trees = self.create_tree(self.f1, self.lists)
    self.trees.place(x=25, y=80)

    btn_back = Button(self.f1, text="Back", fg="black", bg="purple",
                      font='Papyrus 10 bold', width=10, command=self.rm)
    btn_back.place(x=250, y=400)

    c = self.db.execute("select* from book_info")
    g = c.fetchall()
    if len(g) !=0:
        for row in g:
            self.trees.insert('',END,values=row)
    self.db.commit()
```

Student Frame:

In this frame there are 7 labels , 2 entries and 3 buttons).

1. Issue book
2. Return book
3. Student Activity

- **Code:**

```
def student(self):
    self.ph1.destroy()
    self.ph1 = self.photos(img2)
    b1 = Button(self.ph1, text="Issue book", font='Papyrus 22 bold',
                fg='white', bg='Black', width=15, padx=10, command=self.issue)
    b1.place(x=12, y=100)
    b2 = Button(self.ph1, text="Return book", font='Papyrus 22 bold',
                fg='white', bg='Black', width=15, padx=10, command=self.returnn)
    b2.place(x=12, y=200)
    b3 = Button(self.ph1, text="Student Activity", font='Papyrus 22 bold',
                fg='white', bg='Black', width=15, padx=10, command=self.activetiy)
    b3.place(x=12, y=300)
    b4 = Button(self.ph1, text="Main Page", font='Papyrus 22 bold',
                fg='white', bg='Black', width=15, padx=10, command=self.mainmenu)
    b4.place(x=12, y=600)
```

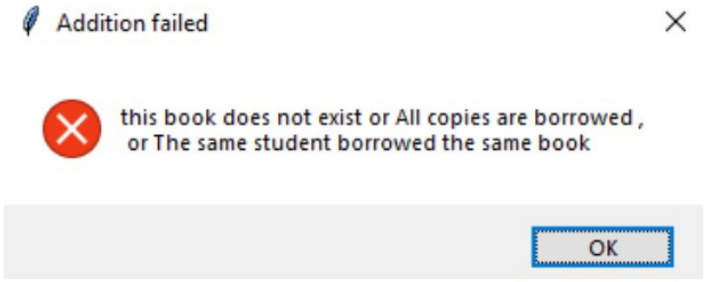
1. Issue book

In our system we can issue a book in the library by entering book id and student id .Every student can return a book within 7 days.

- **Code :**

```
def issue(self):
    self.a_id = IntVar()
    self.a_student = IntVar()
    self.f1 = Frame(self.ph1, height=550, width=500, bg='black')
    self.f1.place(x=500, y=100)
    lb1 = Label(self.f1, text="Book ID : ",font="papyrus 15 bold", fg="white", bg="black")
    lb1.place(x=50, y=100)
    e1 = Entry(self.f1, textvariable=self.a_id, bg="white", width=25, bd=4)
    e1.place(x=180, y=100)
    lb2 = Label(self.f1, text="Student ID : ",font="papyrus 15 bold", fg="white", bg="black")
    lb2.place(x=50, y=150)
    e2 = Entry(self.f1, textvariable=self.a_student,bg="white", width=25, bd=4)
    e2.place(x=180, y=150)
    btn1 = Button(self.f1, text="Issue", bg="purple",fg="black", width=10, bd=3, font='Papyrus 10 bold', command=self.IssueBook_db)
    btn1.place(x=50, y=250)
    btn2 = Button(self.f1, text="Back", bg="purple", fg="black", width=10, font='Papyrus 10 bold', bd=3, command=self.rm)
    btn2.place(x=200, y=250)
```

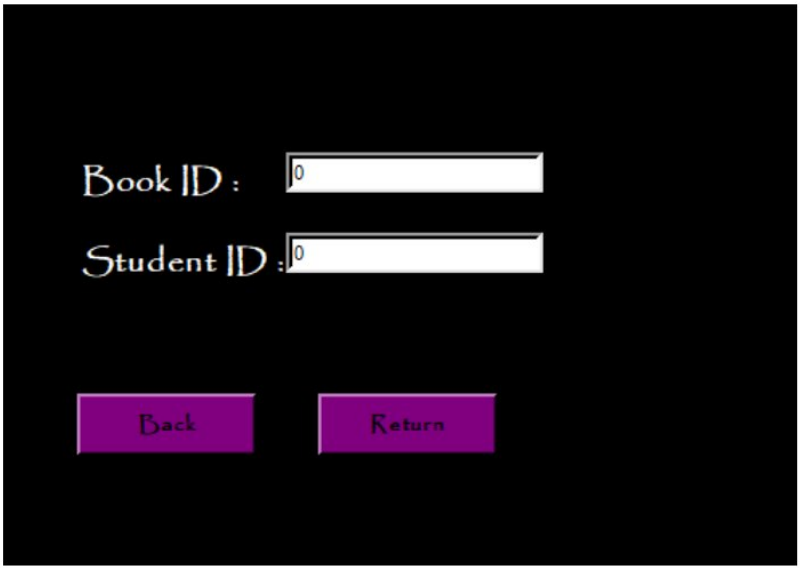
- If the student or book has been added, or the number of books has run out, this message will appear:



- Else

2. Return book :

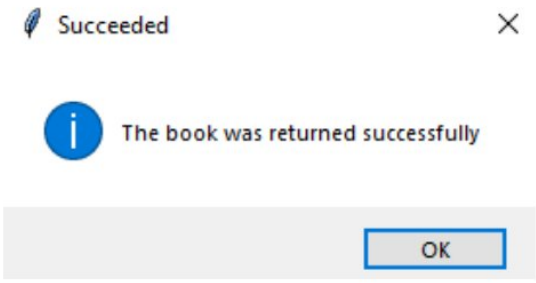
In our system we can delete a book from the library by entering book id and clicking delete.



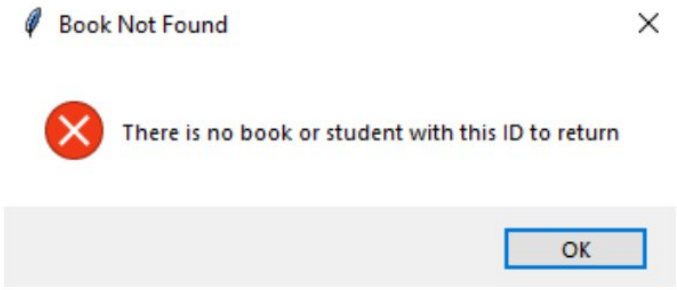
- **Code :**

```
def returnn(self):
    self.a_id = IntVar()
    self.a_student = IntVar()

    self.f1 = Frame(self.ph1, height=550, width=500, bg='black')
    self.f1.place(x=500, y=100)

    l1 = Label(self.f1, text='Book ID : ',font='papyrus 15 bold', fg='white', bg='black')
    l1.place(x=50, y=100)
    e1 = Entry(self.f1, width=25, bd=4, bg='white', textvariable=self.a_id)
    e1.place(x=180, y=100)
    l2 = Label(self.f1, text='Student ID : ',font='papyrus 15 bold', fg='white', bg='black')
    l2.place(x=50, y=150)
    e2 = Entry(self.f1, width=25, bd=4, bg='white',textvariable=self.a_student)
    e2.place(x=180, y=150)
    b1 = Button(self.f1, text='Back', font='Papyrus 10 bold',bg='purple', fg='black', width=10, bd=3, command=self.rm)
    b1.place(x=50, y=250)
    b2 = Button(self.f1, text='Return', font='Papyrus 10 bold', bg='purple', fg='black', width=10, bd=3, command=self.ReturnBook_db)
    b2.place(x=200, y=250)
    self.f1.grid_propagate(0)
```

- When you enter the correct information, it will be retrieved and this message will appear

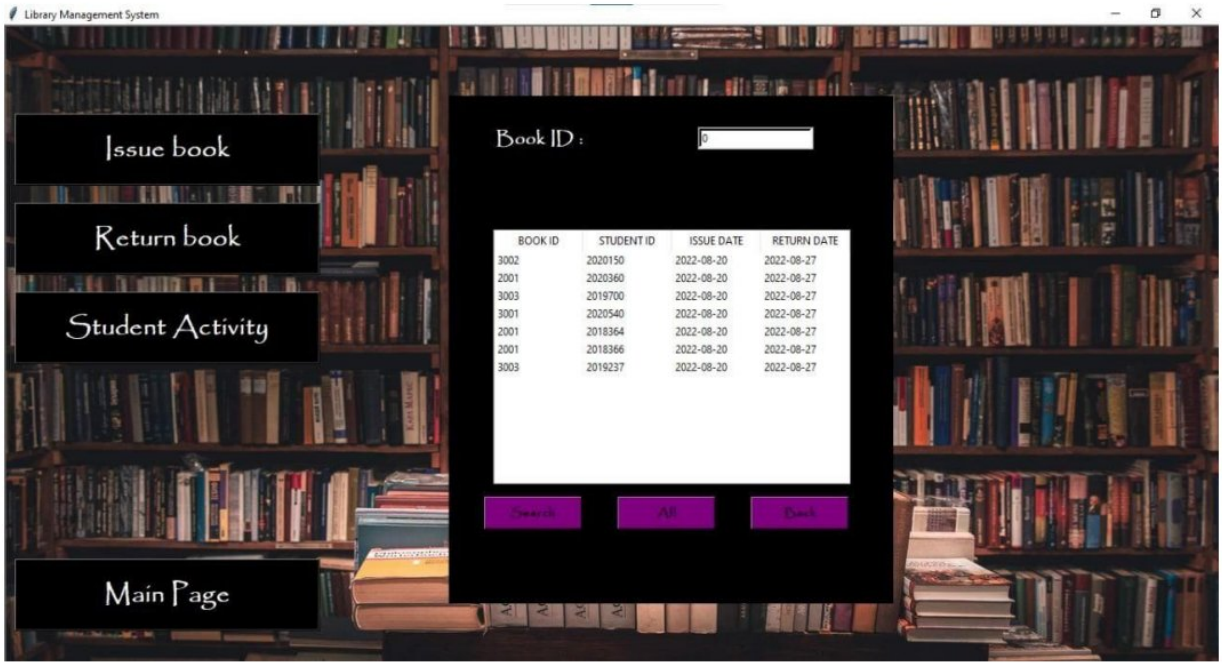


- When you enter the incorrect information, it will be retrieved and this message will appear



Student Activity:

This table shows all student activity .Book id , Student id , Issue date and return date . We used treeview to show data in a table .



- **Code:**

```
def activetiy(self):
    self.a_id = IntVar()
    self.a_student = IntVar()
    self.f1 = Frame(self.ph1, width=500, height=550, bg='black')
    self.f1.place(x=500, y=80)

    self.lists = ("BOOK ID", "STUDENT ID", "ISSUE DATE", "RETURN DATE")
    self.trees = self.create_tree(self.f1, self.lists)
    self.trees.place(x=50, y=150)

    lb1 = Label(self.f1, text="Book ID :", fg="white",
                bg="black", font="Papyrus 15 bold")
    lb1.place(x=50, y=30)
    en1 = Entry(self.f1, textvariable=self.a_id,
                width=20, bd=4, bg="white")
    en1.place(x=280, y=35)
    btn1 = Button(self.f1, text="Search", bg="purple",
                  fg="black", width=10, font='Papyrus 10 bold',command=self.searchact)
    btn1.place(x=40, y=450)
    btn2 = Button(self.f1, text="All", bg="purple",
                  fg="black", width=10, font='Papyrus 10 bold',command=self.searchall)
    btn2.place(x=190, y=450)
    btn3 = Button(self.f1, text="Back", bg="purple", fg="black",
                  width=10, font='Papyrus 10 bold', command=self.rm)
    btn3.place(x=340, y=450)
```

- **Code:**

,and can you search by Book Id :

```
def searchact(self):
    self.list2 = ("BOOK ID", "STUDENT ID", "ISSUE DATE", "RETURN DATE")
    self.trees = self.create_tree(self.f1, self.list2)
    self.trees.place(x=50, y=150)
    try:
        c = self.db.execute("select * from book_issue where book_id=? ",
                            (self.a_id.get(),))
        d = c.fetchall()
        if len(d) != 0:
            for row in d:
                self.trees.insert("", END, values=row)
        else:
            messagebox.showinfo("Error", "Data not found.")
        self.db.commit()
    except Exception as e:
        messagebox.showinfo(e)

def searchall(self):
    self.list2 = ("BOOK ID", "STUDENT ID", "ISSUE DATE", "RETURN DATE")
    self.trees = self.create_tree(self.f1, self.list2)
    self.trees.place(x=50, y=150)
    try:
        c = self.db.execute("select * from Book_issue")
        d = c.fetchall()
        for row in d:
            self.trees.insert("", END, values=row)
        self.db.commit()

    except Exception as e:
        messagebox.showinfo(e)
```

We have three tables in database :-

1. Book_info
2. Book_issue
3. Admins

And We have three Modules to create and Operations on data base (create, insert ,update, delete ,select):

1. cGeneral .
2. cBook_info.
3. cBook_issue.

DataBase :

1. Book_info:

- In this table there are 6 columns:-
  1. book_id (PK) : to Store id for books.
  2. book_title: to Store name title for books.
  3. book_author: to Store name author for books.
  4. book_genre: to Store Genre for books.
  5. book_copies: to Store number of copies.
  6. Book_location: To store the book location

- **Schema:**

Book_info	
book_id	INTEGER
book_titel	TEXT
book_author	TEXT
book_genre	TEXT
book_copies	INTEGER
book_location	TEXT

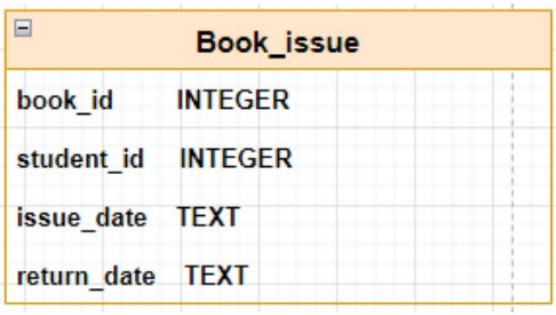
- **Code :**

```
CREATE TABLE IF NOT EXISTS `Book_info`(
`book_id` INTEGER  PRIMARY KEY ,
`book_title` TEXT NOT NULL,
`book_author` TEXT NOT NULL,
`book_genre` TEXT NOT NULL,
`book_copies` INTEGER NOT NULL,
`book_location` TEXT NOT NULL
```

2. Book_issue:

- In this table there are 4 columns :-
  1. book_id: to Store id for books.
  2. student_id : to Store id for students.
  3. issue_date : to store the borrowing date
  4. return_date: to store the return date

- **Schema** :-



- **Code:-**

```
Create Table IF NOT EXISTS `Book_issue` (
        `book_id` INTEGER  NOT NULL,
        `student_id` INTEGER NOT NULL,
        `issue_date` date not null,
        `return_date` date not null,
        FOREIGN KEY(book_id) REFERENCES Book_info(book_id)
```

3. Admins :

- In this table there are 4 columns :-
    1. admin_id: to Store id for admin and it is auto increment .
    2. username : to Store username for admins.
    3. password : to store password for admins.

- **Schema:**

Admins	
admin_id	INTEGER
username	TEXT
password	TEXT

- **Code:**

```
CREATE TABLE IF NOT EXISTS `Admins` (admin_id INTEGER NOT NULL
PRIMARY KEY  AUTOINCREMENT, "
        "username TEXT ,"
        " password TEXT)
```

Modules:

1. cGeneral :

we create this module to include functions to use it in general on any tables in database, and include this Functions:

cGeneral
__init__(self,nameOfDataBase)
DeletePK_isString(self, table, PK, PkValue)
DeletePK_isInt(self, table, PK, PkValue)
SelectOnePK_isString(self, table, PK, PkValue)
SelectOnePK_isInt(self, table, PK, PkValue)
SelectAll(self,table)
getNumOf_book(self,id)
__del__(self)

- _init(Self,nameOfDataBase) :

Constractor used to connect to database and take name of database as input

- **Code:**

```
try:
  self.db = sqlite3.connect(f"{nameOfDataBase}")
   self.cr = self.db.cursor()
  except Exception as e:
   messagebox.showerror(
   "Connction Filed", f"Connection Failed with
database.\n\t\t{e}")
```

- **DeletePK_isString(self, table, PK, PkValue)**
- **DeletePK_isInt(self, table, PK, PkValue):**

two function it is same functionality , Delete function used to delete any 'Row ' in any table in database and take input (name of table , primary key ,value of primary key)

- isString : Use this when PK is String.
- isn't : Use this when PK is Integer

- **Code** :

```
                try:
            mydata = self.SelectOnePK_isInt(table, PK, PkValue)
            if mydata != None:
             self.cr.execute(
             f"Delete FROM `{table}` WHERE {PK}={PkValue}")
              self.db.commit()

            return True  # return >> to check in gui deleted done or not
            else:
            return False  # return >> to check in gui deleted done or not
            except Exception as e:
            messagebox.showerror(
            "DataBase Error", f"Error in database .\n\t\t{e}")
            print(
            f"{'='*20} \n > Error in class cGeneral from Function
'DeletePK_isInt' \n{'='*20}")
```

- **SelectOnePK_isString(self, table, PK, PkValue)**
- **SelectOnePK_isInt(self, table, PK, PkValue)**

two function it is same functionality , select function used to select any row in any table in database and take input (name of table , primary key ,value of primary key)

- isString : Use this when PK is String.
- isn't : Use this when PK is Integer

- **Code:**

```
try:
        self.cr.execute(f"SELECT * FROM `{table}` WHERE {PK}={PkValue} ")
        return self.cr.fetchone()
    except Exception as e:
        messagebox.showerror(
            "DataBase Error", f"Error in database .\n\t\t{e} ")
        print(
            f"{'='*20} \n > Error in class cGeneral from Function 'SelectOnePK_isInt' \n{'='*20}")
```

- **SelectAll(self,table)**

used to select All rows in any table in database and take input (name of table)

- **Code:**

```
try:
    self.cr.execute(f"SELECT * FROM `{table}`")
    return self.cr.fetchall()
except Exception as e:
    messagebox.showerror(
        "DataBase Error", f"Error in database .\n\t\t{e}
")
    print(  f"{'='*20} \n > Error in class cGeneral
from Function 'SelectAll' \n{'='*20}")
```

➢ **getNumOf_book(self,id):**

this function used to get number of books from table `Book_info` in database.

- **Code:**

```
try:
    self.cr.execute(
        f"SELECT `book_copies` FROM `Book_info`
WHERE `book_id`={id}")
    return self.cr.fetchone()
except Exception as e:
    messagebox.showerror(
        "DataBase Error", f"Error in database .\n\t\t{e} ")
    print(
        f"{'='*20} \n > Error in class cGeneral from Function
'GetnumOf_book' \n{'='*20}")
```

- **__del__(self) :**

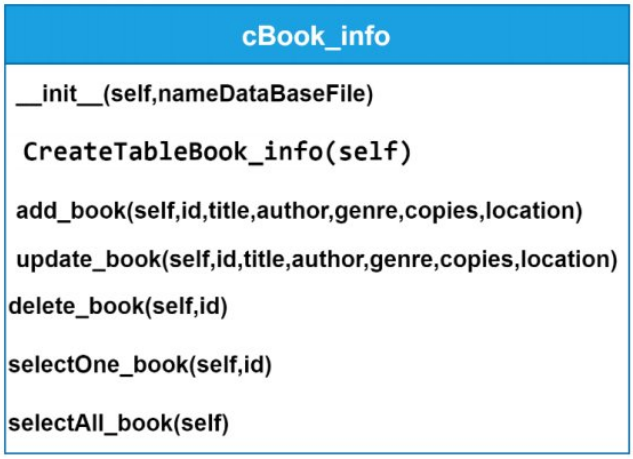
  this destructor used to commit all command and close connection with database.

- **Code:**

```
try:
        self.db.commit()
        self.db.close()
        print("close Connection 'Cgeneral Class'")
    except Exception as e:
        messagebox.showerror(
            "DataBase Error", f"Error in database .\n\t\t{e}
")
        print(
            f"{'='*20} \n > Error in class cGeneral from
Function '__del__' \n{'='*20}")
```

2. cBook_info :

we create this module to be able to perform operations on the table `Book_info`



- **_init(self,nameOfDataBaseFile):**

Constractor used to connect to database and take name of database as input and create Object from cGeneral.

- **Code**:

```
try:
    self.dataBaseName = dataBaseName
    # Create DB and Connection
    self.db = sqlite3.connect(f"{dataBaseName}")
    # Setting Up The Cursor
    self.cr = self.db.cursor()
    # Create Object From Cgeneral Class
    self.generalClass = cGeneral(f"{dataBaseName}")
except Exception as e:
    messagebox.showerror(
        "Connction Filed", f"Connection Failed with database.")
```

```
            print(
                f"{'='*20} \n > Error in class cBook_info from Function
        '__init__' \n{e}\n{'='*20}")
```

- **CreateTableBook_info(self):**

  Create Table `book_info` in DataBase

- **Code**:

```
                try:
                    self.cr.execute(""" CREATE TABLE IF NOT
                EXISTS `Book_info`(
                        `book_id` INTEGER  PRIMARY KEY ,
                        `book_title` TEXT NOT NULL,
                        `book_author` TEXT NOT NULL,
                        `book_genre` TEXT NOT NULL,
                        `book_copies` INTEGER NOT NULL,
                        `book_location` TEXT NOT NULL)""")
                    self.db.commit()
                except Exception as e:
                    messagebox.showerror(
                        "DataBase Error", f"Error in database , You
                can't Create  Book_info. ")
                    print(
                        f"{'='*20} \n > Error in class cBook_info from
                Function 'CreateTableBook_info'\n{e} \n{'='*20}")
```

- **InsertIntoBook_info (self,id,title,author,genre,copies,location):**

  To insert data into table`Book_info`

- **Code:**

```
try :

NumOfCopies=self.generalClass.GetNumOf_book(id)
    if  NumOfCopies != None:
        NewNumOfCopies = NumOfCopies[0]+copies
        cBook_info.UpdateIntoBook_info(self,id, title,
author, gener, NewNumOfCopies, location)

    else:
        self.cr.execute(f""" INSERT INTO `Book_info`
(book_id, book_title ,book_author ,book_genre
,book_copies,book_location)
            VALUES ({id},
'{title}','{author}','{gener}',{copies},'{location}')""")
        self.db.commit()
        messagebox.showinfo("Succeeded", "Successfully
added ")
  except Exception as e:
    messagebox.showerror(
```

```
                "DataBase Error", f"Error in database , You can't
add a book. ")
            print(
                f"{'='*20} \n > Error in class cBook_info from
Function 'UpdateIntoBook_info' \n{e}\n{'='*20}")
```

- **UpdateIntoBook_info**(self, id, title, author, gener, copies, location):
  To update data into `Book_info`

  - **Code:**

```
            try:

                self.cr.execute(
                    f""" UPDATE `Book_info` SET
book_title='{title}',book_author='{author}',book_genre='
{gener}',book_copies={copies} ,
book_location='{location}' WHERE book_id={id}""")
                self.db.commit()
                messagebox.showinfo("Succeeded",
"Successfully update ")

            except Exception as e:
```

```
                messagebox.showerror(
                    "DataBase Error", f"Error in database , You
            can't Update a book. ")
                print(
                    f"{'='*20} \n > Error in class cBook_info from
            Function 'UpdateIntoBook_info' \n{e}\n{'='*20}")
```

- **DeleteBook(self,id):**

  To delete date from table `Book_info`

  - **Code:**

```
        checkDlelete
    =self.generalClass.DeletePK_isInt('Book_info','book_id',id)


        if  checkDlelete:
            messagebox.showinfo("Delete succeeded", " deleted")
        else:
            messagebox.showerror("Delete  failed","book not found")
```

- **SelectOneBook(self, id):**

  To select one data from `Book_info`

  - **Code:**

    return self.generalClass.SelectOnePK_isInt('Book_info', 'book_id', id)

- **SelectAllBooks(self):**

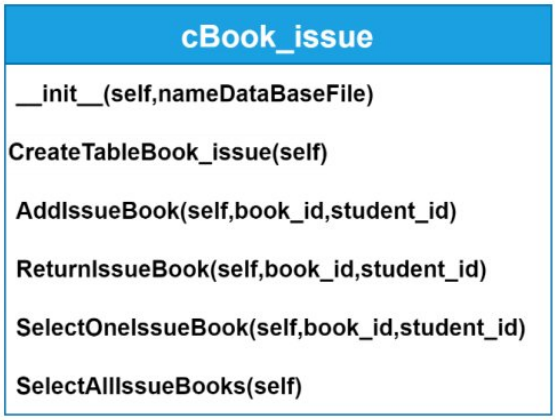
  To select All data from `Book_info`

  - **Code**:

    return self.generalClass.SelectAll('Book_info')

3. cBook_issue:

we create this module to be able to perform operations on the table `Book_issue`



- **_init(Self,nameOfDataBase) :**

  Constractor used to connect to database and take name of database as input and create object from cGeneral

- **Code:**

```
try:
        self.dataBaseName = dataBaseName
        # Create DB and Connection
        self.db = sqlite3.connect(f"{dataBaseName}")
        # Setting Up The Cursor
        self.cr = self.db.cursor()
        # Create Object From Cgeneral Class
        self.generalClass = cGeneral(f"{dataBaseName}")
    except Exception as e:
        messagebox.showerror(
            "Connction Filed", f"Connection Failed with database.")
        print(f"{'='*20} \n > Error in class cBook_issue from Function '__init__' \n{e}\n{'='*20}")
```

- **CreateTableBook_issue(self) :**

  Create Table `book_issue` in DataBase.

- **Code:**

```
try:
        self.cr.execute(
            """Create Table IF NOT EXISTS `Book_issue`
(
                `book_id` INTEGER  NOT NULL,
                `student_id` INTEGER NOT NULL,
                `issue_date` date not null,
                `return_date` date not null,
                FOREIGN KEY(book_id) REFERENCES
Book_info(book_id)
            )""")
        self.db.commit()
    except Exception as e:
        messagebox.showerror(
            "DataBase Error", f"Error in database , You
can't Create  Book_issue. ")
        print(
            f"{'='*20} \n > Error in class cBook_issue from
Function 'CreateTableBook_issue'\n{e} \n{'='*20}")
```

- **AddIssueBook(self,book_id,student_id)**

  to insert into table`Book_issue`

- **Code:**

```
try:
        # Check if exist or not
        book = cBook_info(self.dataBaseName)
        issueBook=book.SelectOneBook(book_id)


        if issueBook !=None  and issueBook[4] > 0 and
self .SelectOneIssueBook(student_id)==None:
            self.cr.execute(
            f"insert into `Book_issue`
values({book_id},{student_id},date('now'),date('now','+7
day'))")
            self.db.commit()
            # update copies -1
            self.cr.execute(
               f""" UPDATE `Book_info` SET
book_copies=book_copies-1 WHERE
book_id={book_id}""")
            self.db.commit()

            messagebox.showinfo(
               "Succeeded", "Successfully added ")
```

```
                else:
                    messagebox.showerror(
                        "Addition failed ", "this book does not exist
or All copies are borrowed ,\n or The same student
borrowed the same book")
            except Exception as e:
                messagebox.showerror(
                    "DataBase Error", f"Error in database , You
can't add into  Book_issue. ")
                print(
                    f"{'='*20} \n > Error in class cBook_issue from
Function 'AddIssueBook'\n{e} \n{'='*20}")
```

- **ReturnIssueBook(self,book_id,student_id):**

  To delete data from table`Book_issue`

- **Code:**

```
            try:
                if self.SelectOneIssueBook(book_id) != None:
                    self.cr.execute(f"DELETE FROM `Book_issue`
WHERE book_id ={book_id} AND student_id ={student_id}")
                    self.db.commit()
                    # update copies +1
                    self.cr.execute(
                        f""" UPDATE `Book_info` SET
book_copies=book_copies+1 WHERE book_id={book_id}""")
                    self.db.commit()
```

```
                messagebox.showinfo(
                    "Succeeded", "The book was returned
successfully")

            else:
                messagebox.showerror(
                    "Book Not Found", "There is no book or student
with this ID to return")
        except Exception as e:
            messagebox.showerror(
                "DataBase Error", f"Error in database , You can't
Return  Book_issue. ")
            print(
                f"{'='*20} \n > Error in class cBook_issue from
Function 'ReturnIssueBook'\n{e} \n{'='*20}")
```

- **SelectOneIssueBook(self, book_id):**

  To select one data from`Book_issue`

  - **Code:**

```
        self.cr.execute(
            f"SELECT * FROM `Book_issue` WHERE  book_id
={book_id}  ")
        return self.cr.fetchone()
```

- **SelectAllIssueBooks(self):**

  To select all data from `Book_issue`

  - **Code:**

    return self.generalClass.SelectAll('Book_issue')