**ABSTRACT**

The project I have developed is known as library manager.

Basically using this program any schools or libraries can maintain their books in more efficient way.

I created this program with thoughts of helping all the libraries and schools to manage their library efficiently my program contains of 6 parts, they include;

* Main file
* Add book file
* View book file
* Delete book file
* Issue book file
* Return book file

The main file includes all the code to create an interface that is intuitive enough and contains the paths to go to other files via buttons the buttons are add, view, delete, issue and return books.

Once any of the button is clicked the program open another window to perform certain task so if the add book button is clicked the program runs the add book file and similarly for other buttons also.

In add book the user can add any books with book id title author and the status (whether issued or available).

In view book the user can view all the books that are available in the library

In delete books the user can remove the books from the library if any of the books were damaged or lost.

In issue book the librarian can issue the books to customer the librarian must enter the book id and the date of book issue

In return book the user can return a book that has been issued already to him/her the user must enter the book id and date of return

Finally with this program any one can manage their library in most efficient way!!

**OVERVIEW OF PYTHON**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Some facts about python**

* Python was developed in the late eighties, i.e., the late 1980's by **Guido van Rossum** at the **National Research Institute for Mathematics and Computer Science** in the Netherlands as a successor of ABC language capable of exception handling and interfacing.
* Python is derived from programming languages such as ABC, Modula 3, small talk, Algol-68.
* Van Rossum picked the name Python for the new language from a TV show, Monty Python's Flying Circus.
* Python page is a file with a **.py** extension that contains could be the combination of HTML Tags and Python scripts.
* In December 1989, the creator developed the 1st python interpreter as a hobby, and then on 16 October 2000, Python 2.0 was released with many new features.
* On 3rd December 2008, **Python 3.0** was released with more testing and included new features.
* Python is an open-source scripting language.
* Python is open-source, which means that anyone can download it freely from **www.python.org** and use it to develop programs. Its source code can be accessed and modified as required in the project.
* Python is one of the official languages at Google.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

## Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

* Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
* Win 9x/NT/2000
* Macintosh (Intel, PPC, 68K)
* OS/2
* DOS (multiple versions)
* PalmOS
* Nokia mobile phones
* Windows CE
* Acorn/RISC OS
* BeOS
* Amiga
* VMS/OpenVMS
* QNX
* VxWorks
* Psion
* Python has also been ported to the Java and .NET virtual machines

## Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org/>

You can download Python documentation from <https://www.python.org/doc/>. The documentation is available in HTML, PDF, and PostScript formats.

## Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

**Unix and Linux Installation**

Here are the simple steps to install Python on Unix/Linux machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link to download zipped source code available for Unix/Linux.
* Download and extract files.
* Editing the *Modules/Setup* file if you want to customize some options.
* run ./configure script
* make
* make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

**Windows installation**

Here are the steps to install Python on Windows machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
* To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
* Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

**Macintosh Installation**

Recent Macs come with Python installed, but it may be several years out of date. See [http://www.python.org/download/mac/](https://www.python.org/download/mac/) for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.

Jack Jansen maintains it and you can have full access to the entire documentation at his website − <http://www.cwi.nl/~jack/macpython.html>. You can find complete installation details for Mac OS installation.

## Setting up PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

## Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix −

* **In the csh shell** − type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
* **In the bash shell (Linux)** − type export PATH="$PATH:/usr/local/bin/python" and press Enter.
* **In the sh or ksh shell** − type PATH="$PATH:/usr/local/bin/python" and press Enter.
* **Note** − /usr/local/bin/python is the path of the Python directory

## Setting path at Windows

To add the Python directory to the path for a particular session in Windows −

**At the command prompt** − type path %path%;C:\Python and press Enter.

**Note** − C:\Python is the path of the Python directory

## Running Python

There are three different ways to start Python −

**Interactive Interpreter**

You can start Python from UNIX, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter **python** the command line.

Start coding right away in the interactive interpreter.

**Integrated Development Environment**

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

* **Unix** − IDLE is the very first Unix IDE for Python.
* **Windows** − Python Win is the first Windows interface for Python and is an IDE with a GUI.
* **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

**Note** − All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavour of Linux.

We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

## First Python Program

Let us execute programs in different modes of programming.

**Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result −

Hello, Python!

**Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file −

print "Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

Let us try another way to execute a Python script. Here is the modified test.py file −

#!/usr/bin/python

print "Hello, Python!"

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

$ chmod +x test.py # This is to make file executable

$./test.py

This produces the following result −

Hello, Python!

### Advantages of Python

Let’s see how Python dominates over other languages.

#### 1. Extensive Libraries

Python downloads with an extensive library and contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.

So, we don’t have to write the complete code for that manually.

#### 2. Extensible

Python can be**extended to other languages**. You can write some of your code in languages like C++ or C.

This comes in handy, especially in projects.

#### 3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++.

This lets us add **scripting capabilities** to our code in the other language.

#### 4. Improved Productivity

The language’s simplicity and extensive libraries render programmers**more productive** than languages like Java and C++ do.

Also, the fact that you need to write less and get more things done.

#### 5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things.

This is a way to connect the language with the real world.

#### 6. Simple and Easy

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do.

It is also quite **easy to learn**,**understand**, and **code**.

#### 7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code.

It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

#### 8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms.

While functions help us with code reusability, classes and objects let us model the real world.

A class allows the **encapsulation of data** and functions into one.

#### 9. Free and Open-Source

Python is **freely available**. But not only can you ***download Python*** for free, but you can also download its source code, make changes to it, and even distribute it.

It downloads with an extensive collection of libraries to help you with your tasks.

#### 10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform.

But it isn’t the same with Python. Here, you need to**code only once**, and you can run it anywhere.

This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

#### 11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

### Advantages of Python Over Other Languages

#### 1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages.

Python also has awesome standard library support, so you don’t have to search for any third-party libraries to get your job done.

This is the reason that many people suggest learning Python to beginners.

#### 2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications.

Python is popular and widely used so it gives you better community support.

**The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.**

#### 3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows.

Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and ***machine learning***, automate things, do web scraping and also build games and powerful visualizations.

It is an all-rounder programming language.

### Disadvantages of Python

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well.

Let’s now see the limitations of Python over another language.

#### 1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**.

This, however, isn’t a problem unless speed is a focal point for the project.

#### 2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**.

Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbon Nelle**.

The reason it is not so famous despite the existence of Bryton is that it isn’t that secure.

#### 3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don’t need to declare the type of variable while writing the code.

It uses **duck-typing**. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck.

While this is easy on the programmers during coding, it can**raise run-time errors**.

#### 4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python’s database access layers are a bit underdeveloped.

Consequently, it is less often applied in huge enterprises.

#### 5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person.

To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

## Summary

advantages and disadvantages of Python, I would say while there are some speed, security, and runtime issues, Python is a great language to pick up.

Its popularity speaks for itself. And this popularity is attributed to its being free, easy, interpreted, object-oriented, extensible, embeddable, portable, and readable.

**PROJECT DESCRIPTION**

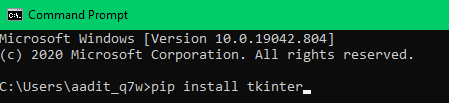
A library management system keeps track of the books present in the library. It is an important piece of software which is a must at schools and colleges. I am building a library management system using Tkinter to make it interactive.

**Its advantages include:**

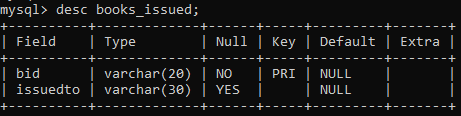
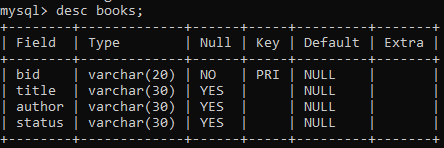
* Speed
* Efficiency
* Easy to use interface

**Beforehand steps:**

* Install MySQL server from: https://www.mysql.com/downloads/
* If on windows go to cmd and type the following:-



* Open my SQL and type the following:
* create database db;
* create table books(bid varchar(20) primary key, title varchar(30), author varchar(30), status varchar(30));
* create table books\_issued(bid varchar(20) primary key, issuedto varchar(30));

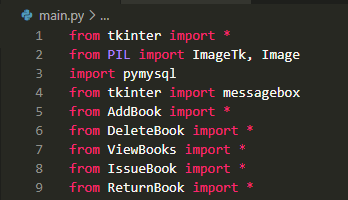


### Code:

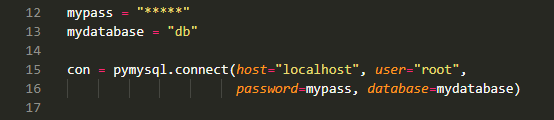
#### 1. main.py

* **Importing the Modules:**

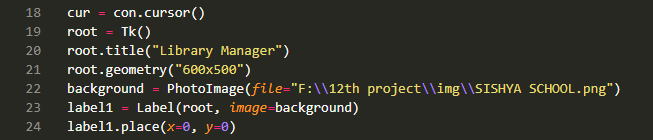
To use Tkinter we need to import the Tkinter module.

 I have imported each file so that we can make function calls from our main file.

* **Connecting to the MySQL server:**

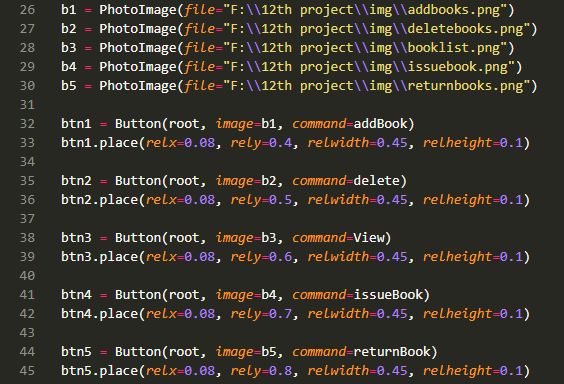
 Now I will connect to the server with the correct credentials associated with the MySQL server installed on my system.

* **Designing the Window:**

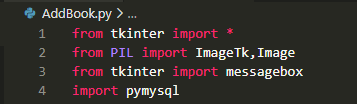
 Now I will design the project window and add a background image. Make sure that the image is in the same directory as the project, in order to avoid confusion.

* **Adding buttons:**

Here I am first using “photoImage” command to extract the data from the image file then I am creating the buttons with appropriate parameter, and finally placing thee buttons in positions



**2. Add book.py**

* **Import modules:**
* **Function – bookRegister():**This function executes an SQL command to insert data into the table and commit the changes.

We fetch the data in the input field by **.get ()** method. Hence after fetching each of the data fields value we can execute an SQL command to insert the data.

Here I am using the try-except block in order to handle the error.

Next, execute the **insertBooks** command by **.execute ()** method associated with cur. Then commit the changes by **.commit ()** method associated with con.

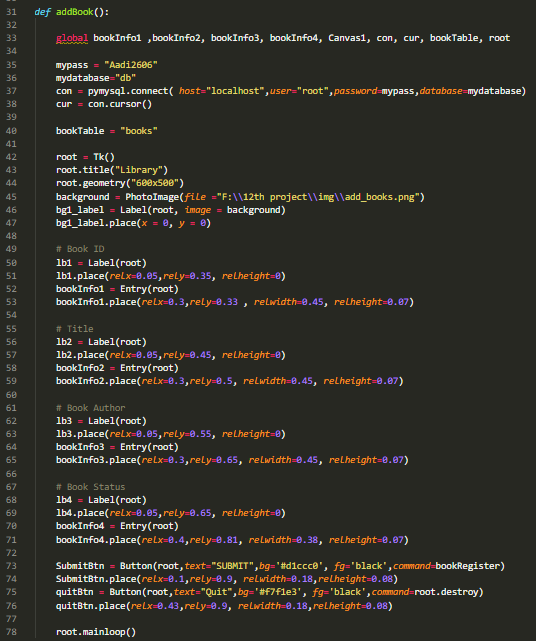
We use **messagebox ()** function to acknowledge the user of success or failure.

* **Function – addBook:**

This function connects to the MySQL server and creates a window for holding new

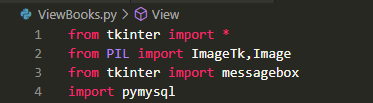
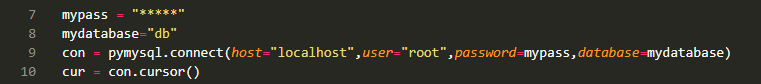
text fields. Then fetch details of a new book from the user and then call

**bookRegister ()** function to register the books into the table.

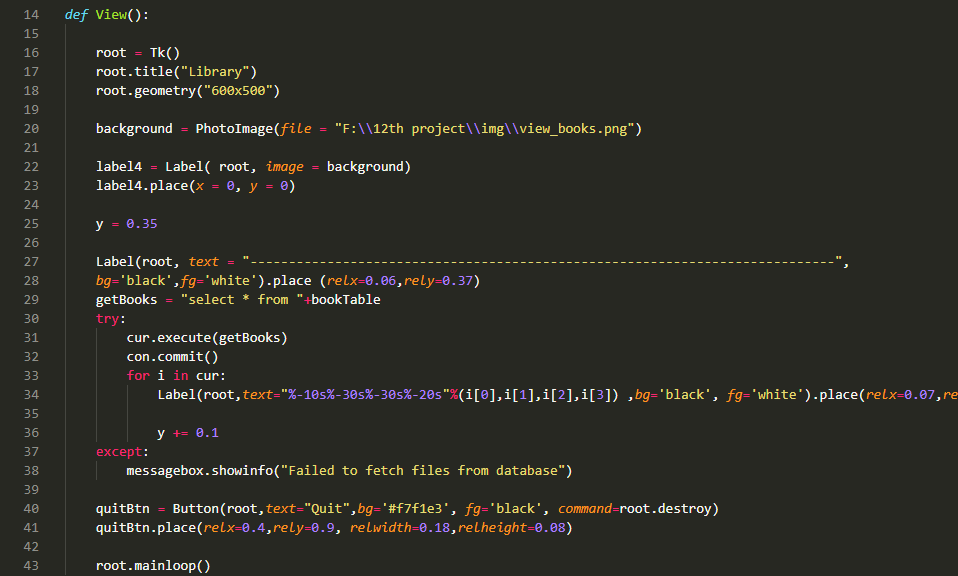


The global variable is used because some of the variables must be used in the function bookregister().Then the MySQL is set up various buttons and labels are created as required

#### 3. ViewBooks.py

* **import modules:**
* **connect to MySQL server:**
* 

#### Function – View():



This function in our library project creates a window for displaying the records in the table.

We create a new window to display the list of books and their status. we create a black box to accommodate the records returned by the **getBooks** query.

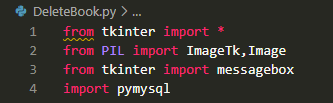
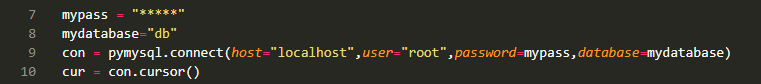
We manually display the name of the columns associated with our books table.

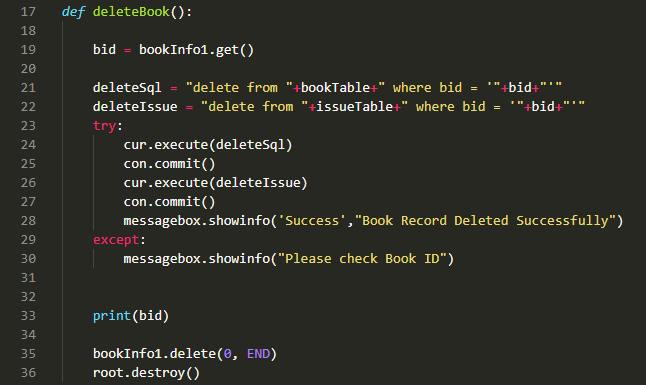
We execute the query stored in **getBooks** using **cur.execute()** display each record one by one as a Label. Each time a record is displayed the value of **y** increases by **‘0.1’**.

In order to handle any discrepancies, we place this code in a try-except block.

In the end, we create and place a button, **quitBtn** to exit from the current window of the library project.

#### 4. DeleteBook.py

* **import modules:**
* **connect to MySQL server:**
* **Function – deleteBook()**

This function primarily checks if the bid (book id) exists in the book table and if it does, it executes the necessary command to remove it.

This is executed in try and except block

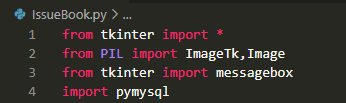
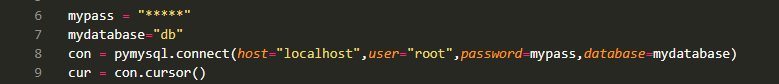
First store the SQL query to delete the record in **deleteSql**. After that, execute this command using **cur.execute ()**.

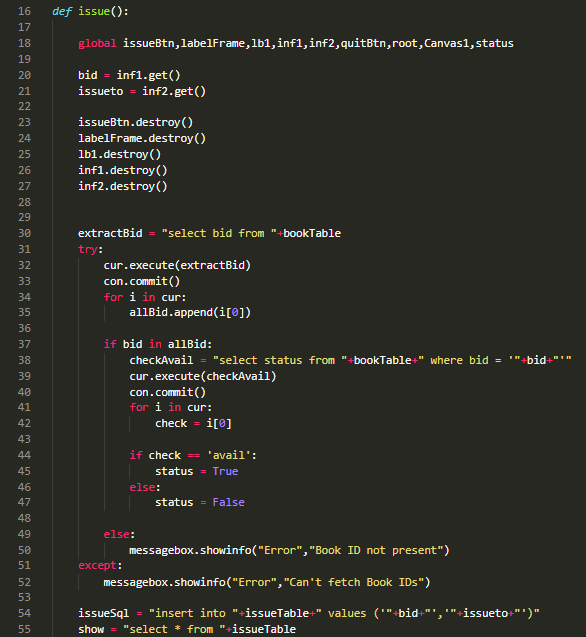
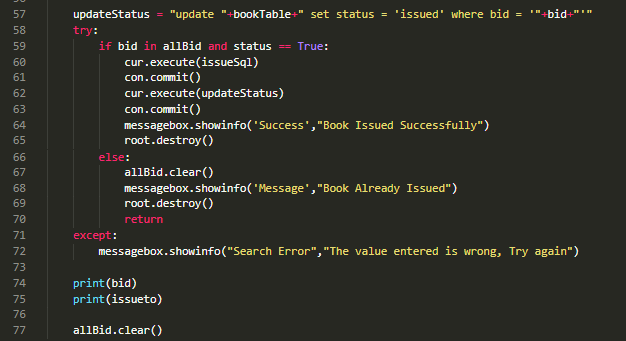
In case someone loses a book, we should delete that book from the **issueTable** in order to prevent errors in the future so store the SQl query to delete the same record from the IssueTable in **deleteIssue**. I execute it along with the **deleteSql**.

* **Function – delete()**

This function creates a window for holding a text input field. We fetch details of a book from the user and then call **deleteBook()** function to delete the book record from the table.

#### 5. IssueBook.py

* **Import modules**
* **2.connect to MySQL**
* **Function – issue()**



Fetch the desired book ID and Issuer’s name and store it into **bid** and **issueto** respectively.

After that we store all the Book ID from the **books** table in **all Bid** by executing the SQL query stored in **extract Bid**.

We check for the existence of the desired **bid** in **all Bid**. If it exists and is available, we set the **status** as **True**.

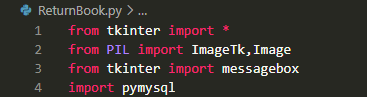
If the book is available we update the **books\_issued** table with the book id (**bid**) and Issuer’s name (**issueto**) and update the **books** table by changing the status of the issued book to ‘**issued**’.

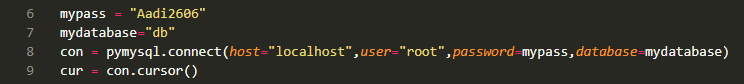
* **Function – issueBook()**

This function creates a window for holding a text input field.

****

#### 6. ReturnBook.py

* **Import module**
* **Connect to MySQL:**



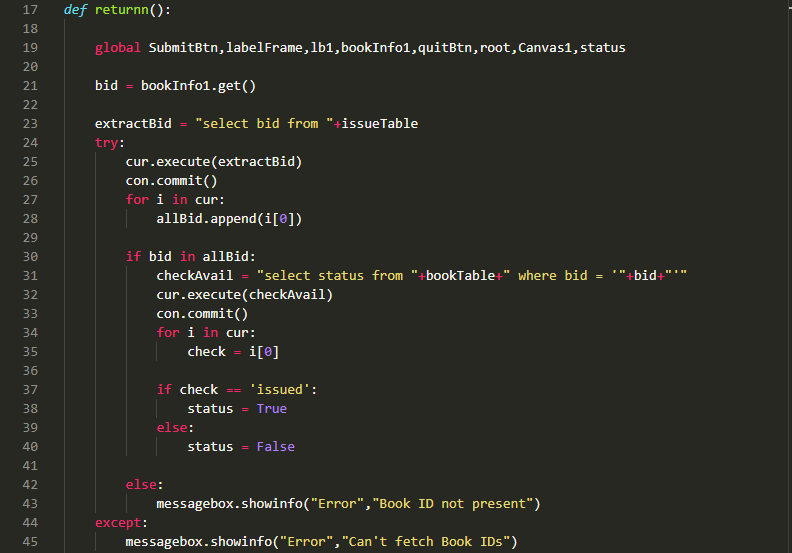
* **Function – returnn():**

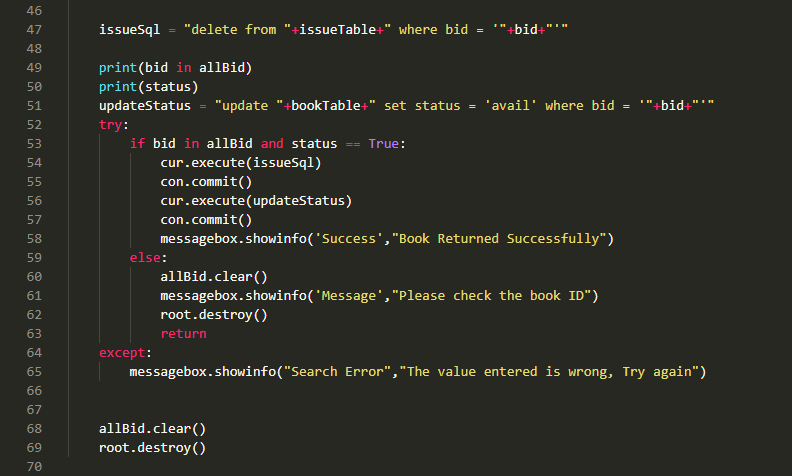
This function is similar to the **issue ()** function we designed for **issueBook.py**.

We fetch the desired book ID and store it into **bid**.

After that we store all the Book IDs from the **books\_issued** table in **allBid** by executing the SQL query stored in **extract Bid**.

We check for the existence of the desired **bid** in **allBid**. Also, we check the status of the same book and if it is ‘**issued**’, we set the **status** as **True**.

Then, we remove the record from **books\_issued** table and update the **books** table by changing the status of the issued book to ‘**avail**’.



* **Function – returnBook()**

This function creates a window for holding a text input field.

**HARDWARE & SOFTWARE REQUIREMENTS**

**Hardware Requirements**

Processor - AMD Athlon(tm) II X2 240 Processor

Speed - 2.81 GHz

RAM - 8 GB

Hard Disk - 256 GB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - LCD/LED

**Software Requirements**

Operating system - Windows 10 Pro

Programming Language - Python

Data Base - MySQL

Tool - VS Code

**HEADER FILES**

from tkinter import \*

from PIL import ImageTk, Image

import pymysql

from tkinter import messagebox

from AddBook import \*

from DeleteBook import \*

from ViewBooks import \*

from IssueBook import \*

from ReturnBook import \*

**FUNCTIONS USED**

bookRegisters ()

addBook ()

deleteBook ()

delete ()

issue ()

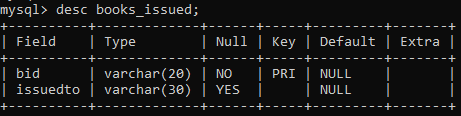
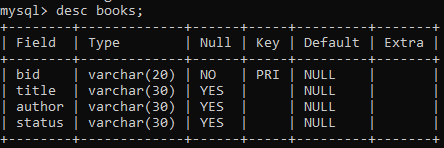
issueBook ()

returnn ()

returnBook ()

View ()

**MYSQL TABLES**



**SOURCE CODE**

## main.py ##

from tkinter import \*

from PIL import ImageTk, Image

import pymysql

from tkinter import messagebox

from AddBook import \*

from DeleteBook import \*

from ViewBooks import \*

from IssueBook import \*

from ReturnBook import \*

mypass = "Aadi2606"

mydatabase = "db"

con = pymysql.connect(host="localhost", user="root",

password=mypass, database=mydatabase)

cur = con.cursor()

root = Tk()

root.title("Library Manager")

root.geometry("600x500")

background = PhotoImage(file="F:\\12th project\\img\\SISHYA SCHOOL.png")

label1 = Label(root, image=background)

label1.place(x=0, y=0)

b1 = PhotoImage(file="F:\\12th project\\img\\addbooks.png")

b2 = PhotoImage(file="F:\\12th project\\img\\deletebooks.png")

b3 = PhotoImage(file="F:\\12th project\\img\\booklist.png")

b4 = PhotoImage(file="F:\\12th project\\img\\issuebook.png")

b5 = PhotoImage(file="F:\\12th project\\img\\returnbooks.png")

btn1 = Button(root, image=b1, command=addBook)

btn1.place(relx=0.08, rely=0.4, relwidth=0.45, relheight=0.1)

btn2 = Button(root, image=b2, command=delete)

btn2.place(relx=0.08, rely=0.5, relwidth=0.45, relheight=0.1)

btn3 = Button(root, image=b3, command=View)

btn3.place(relx=0.08, rely=0.6, relwidth=0.45, relheight=0.1)

btn4 = Button(root, image=b4, command=issueBook)

btn4.place(relx=0.08, rely=0.7, relwidth=0.45, relheight=0.1)

btn5 = Button(root, image=b5, command=returnBook)

btn5.place(relx=0.08, rely=0.8, relwidth=0.45, relheight=0.1)

root.mainloop()

## AddBook.py ##

from tkinter import \*

from PIL import ImageTk,Image

from tkinter import messagebox

import pymysql

def bookRegister():

bid = bookInfo1.get()

title = bookInfo2.get()

author = bookInfo3.get()

status = bookInfo4.get()

status = status.lower()

insertBooks = "insert into "+bookTable+" values('"+bid+"','"+title+"','"+author+"','"+status+"')"

try:

cur.execute(insertBooks)

con.commit()

messagebox.showinfo('Success',"Book added successfully")

except:

messagebox.showinfo("Error","Can't add data into Database")

print(bid)

print(title)

print(author)

print(status)

root.destroy()

def addBook():

global bookInfo1 ,bookInfo2, bookInfo3, bookInfo4, Canvas1, con, cur, bookTable, root

mypass = "Aadi2606"

mydatabase="db"

con = pymysql.connect( host="localhost",user="root",password=mypass,database=mydatabase)

cur = con.cursor()

bookTable = "books"

root = Tk()

root.title("Library")

root.geometry("600x500")

background = PhotoImage(file ="F:\\12th project\\img\\add\_books.png")

bg1\_label = Label(root, image = background)

bg1\_label.place(x = 0, y = 0)

# Book ID

lb1 = Label(root)

lb1.place(relx=0.05,rely=0.35, relheight=0)

bookInfo1 = Entry(root)

bookInfo1.place(relx=0.3,rely=0.33 , relwidth=0.45, relheight=0.07)

# Title

lb2 = Label(root)

lb2.place(relx=0.05,rely=0.45, relheight=0)

bookInfo2 = Entry(root)

bookInfo2.place(relx=0.3,rely=0.5, relwidth=0.45, relheight=0.07)

# Book Author

lb3 = Label(root)

lb3.place(relx=0.05,rely=0.55, relheight=0)

bookInfo3 = Entry(root)

bookInfo3.place(relx=0.3,rely=0.65, relwidth=0.45, relheight=0.07)

# Book Status

lb4 = Label(root)

lb4.place(relx=0.05,rely=0.65, relheight=0)

bookInfo4 = Entry(root)

bookInfo4.place(relx=0.4,rely=0.81, relwidth=0.38, relheight=0.07)

SubmitBtn = Button(root,text="SUBMIT",bg='#d1ccc0', fg='black',command=bookRegister)

SubmitBtn.place(relx=0.1,rely=0.9, relwidth=0.18,relheight=0.08)

quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black',command=root.destroy)

quitBtn.place(relx=0.43,rely=0.9, relwidth=0.18,relheight=0.08)

root.mainloop()

## DeleteBook.py ##

from tkinter import \*

from PIL import ImageTk,Image

from tkinter import messagebox

import pymysql

mypass = "Aadi2606"

mydatabase="db"

con = pymysql.connect(host="localhost",user="root",password=mypass,database=mydatabase)

cur = con.cursor()

issueTable = "books\_issued"

bookTable = "books"

def deleteBook():

bid = bookInfo1.get()

deleteSql = "delete from "+bookTable+" where bid = '"+bid+"'"

deleteIssue = "delete from "+issueTable+" where bid = '"+bid+"'"

try:

cur.execute(deleteSql)

con.commit()

cur.execute(deleteIssue)

con.commit()

messagebox.showinfo('Success',"Book Record Deleted Successfully")

except:

messagebox.showinfo("Please check Book ID")

print(bid)

bookInfo1.delete(0, END)

root.destroy()

def delete():

global bookInfo1,bookInfo2,bookInfo3,bookInfo4,Canvas1,con,cur,bookTable,root

root = Tk()

root.title("Library")

root.geometry("600x500")

background = PhotoImage(file ="F:\\12th project\\img\\delete\_books.png")

bg1\_label = Label(root, image = background)

bg1\_label.place(x = 0, y = 0)

bookInfo1 = Entry(root)

bookInfo1.place(relx=0.27,rely=0.53, relwidth=0.45, relheight=0.067)

SubmitBtn = Button(root,text="SUBMIT",bg='#d1ccc0', fg='black',command=deleteBook)

SubmitBtn.place(relx=0.1,rely=0.87, relwidth=0.18,relheight=0.08)

quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)

quitBtn.place(relx=0.43,rely=0.87, relwidth=0.18,relheight=0.08)

root.mainloop()

## IssueBook.py ##

from tkinter import \*

from PIL import ImageTk,Image

from tkinter import messagebox

import pymysql

mypass = "Aadi2606"

mydatabase="db"

con = pymysql.connect(host="localhost",user="root",password=mypass,database=mydatabase)

cur = con.cursor()

issueTable = "books\_issued"

bookTable = "books"

allBid = []

def issue():

global issueBtn,labelFrame,lb1,inf1,inf2,quitBtn,root,Canvas1,status

bid = inf1.get()

issueto = inf2.get()

issueBtn.destroy()

labelFrame.destroy()

lb1.destroy()

inf1.destroy()

inf2.destroy()

extractBid = "select bid from "+bookTable

try:

cur.execute(extractBid)

con.commit()

for i in cur:

allBid.append(i[0])

if bid in allBid:

checkAvail = "select status from "+bookTable+" where bid = '"+bid+"'"

cur.execute(checkAvail)

con.commit()

for i in cur:

check = i[0]

if check == 'avail':

status = True

else:

status = False

else:

messagebox.showinfo("Error","Book ID not present")

except:

messagebox.showinfo("Error","Can't fetch Book IDs")

issueSql = "insert into "+issueTable+" values ('"+bid+"','"+issueto+"')"

show = "select \* from "+issueTable

updateStatus = "update "+bookTable+" set status = 'issued' where bid = '"+bid+"'"

try:

if bid in allBid and status == True:

cur.execute(issueSql)

con.commit()

cur.execute(updateStatus)

con.commit()

messagebox.showinfo('Success',"Book Issued Successfully")

root.destroy()

else:

allBid.clear()

messagebox.showinfo('Message',"Book Already Issued")

root.destroy()

return

except:

messagebox.showinfo("Search Error","The value entered is wrong, Try again")

print(bid)

print(issueto)

allBid.clear()

def issueBook():

global issueBtn,labelFrame,lb1,inf1,inf2,quitBtn,root,Canvas1,status

root = Tk()

root.title("Library")

root.geometry("600x500")

background = PhotoImage(file ="F:\\12th project\\img\\issue\_books.png")

bg2\_label = Label(root, image = background)

bg2\_label.place(x = 0, y = 0)

inf1 = Entry(root)

inf1.place(relx=0.3,rely=0.44, relwidth=0.45, relheight=0.055)

inf2 = Entry(root)

inf2.place(relx=0.32,rely=0.635, relwidth=0.45, relheight=0.055)

issueBtn = Button(root,text="Issue",bg='#d1ccc0', fg='black',command=issue)

issueBtn.place(relx=0.1,rely=0.87, relwidth=0.18,relheight=0.08)

quitBtn = Button(root,text="Quit",bg='#aaa69d', fg='black', command=root.destroy)

quitBtn.place(relx=0.43,rely=0.87, relwidth=0.18,relheight=0.08)

root.mainloop()

## ReturnBook.py ##

from tkinter import \*

from PIL import ImageTk,Image

from tkinter import messagebox

import pymysql

mypass = "Aadi2606"

mydatabase="db"

con = pymysql.connect(host="localhost",user="root",password=mypass,database=mydatabase)

cur = con.cursor()

issueTable = "books\_issued"

bookTable = "books"

allBid = []

def returnn():

global SubmitBtn,labelFrame,lb1,bookInfo1,quitBtn,root,Canvas1,status

bid = bookInfo1.get()

extractBid = "select bid from "+issueTable

try:

cur.execute(extractBid)

con.commit()

for i in cur:

allBid.append(i[0])

if bid in allBid:

checkAvail = "select status from "+bookTable+" where bid = '"+bid+"'"

cur.execute(checkAvail)

con.commit()

for i in cur:

check = i[0]

if check == 'issued':

status = True

else:

status = False

else:

messagebox.showinfo("Error","Book ID not present")

except:

messagebox.showinfo("Error","Can't fetch Book IDs")

issueSql = "delete from "+issueTable+" where bid = '"+bid+"'"

print(bid in allBid)

print(status)

updateStatus = "update "+bookTable+" set status = 'avail' where bid = '"+bid+"'"

try:

if bid in allBid and status == True:

cur.execute(issueSql)

con.commit()

cur.execute(updateStatus)

con.commit()

messagebox.showinfo('Success',"Book Returned Successfully")

else:

allBid.clear()

messagebox.showinfo('Message',"Please check the book ID")

root.destroy()

return

except:

messagebox.showinfo("Search Error","The value entered is wrong, Try again")

allBid.clear()

root.destroy()

def returnBook():

global bookInfo1,SubmitBtn,quitBtn,Canvas1,con,cur,root,labelFrame, lb1

root = Tk()

root.title("Library")

root.geometry("600x500")

background = PhotoImage(file = "F:\\12th project\\img\\return\_books.png")

label5 = Label( root, image = background)

label5.place(x = 0, y = 0)

bookInfo1 = Entry(root)

bookInfo1.place(relx=0.27,rely=0.53, relwidth=0.45, relheight=0.067)

SubmitBtn = Button(root,text="Return",bg='#d1ccc0', fg='black',command=returnn)

SubmitBtn.place(relx=0.28,rely=0.9, relwidth=0.18,relheight=0.08)

quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)

quitBtn.place(relx=0.53,rely=0.9, relwidth=0.18,relheight=0.08)

root.mainloop()## ReturnBook.py ##

from tkinter import \*

from PIL import ImageTk,Image

from tkinter import messagebox

import pymysql

mypass = "Aadi2606"

mydatabase="db"

con = pymysql.connect(host="localhost",user="root",password=mypass,database=mydatabase)

cur = con.cursor()

issueTable = "books\_issued"

bookTable = "books"

allBid = []

def returnn():

global SubmitBtn,labelFrame,lb1,bookInfo1,quitBtn,root,Canvas1,status

bid = bookInfo1.get()

extractBid = "select bid from "+issueTable

try:

cur.execute(extractBid)

con.commit()

for i in cur:

allBid.append(i[0])

if bid in allBid:

checkAvail = "select status from "+bookTable+" where bid = '"+bid+"'"

cur.execute(checkAvail)

con.commit()

for i in cur:

check = i[0]

if check == 'issued':

status = True

else:

status = False

else:

messagebox.showinfo("Error","Book ID not present")

except:

messagebox.showinfo("Error","Can't fetch Book IDs")

issueSql = "delete from "+issueTable+" where bid = '"+bid+"'"

print(bid in allBid)

print(status)

updateStatus = "update "+bookTable+" set status = 'avail' where bid = '"+bid+"'"

try:

if bid in allBid and status == True:

cur.execute(issueSql)

con.commit()

cur.execute(updateStatus)

con.commit()

messagebox.showinfo('Success',"Book Returned Successfully")

else:

allBid.clear()

messagebox.showinfo('Message',"Please check the book ID")

root.destroy()

return

except:

messagebox.showinfo("Search Error","The value entered is wrong, Try again")

allBid.clear()

root.destroy()

def returnBook():

global bookInfo1,SubmitBtn,quitBtn,Canvas1,con,cur,root,labelFrame, lb1

root = Tk()

root.title("Library")

root.geometry("600x500")

background = PhotoImage(file = "F:\\12th project\\img\\return\_books.png")

label5 = Label( root, image = background)

label5.place(x = 0, y = 0)

bookInfo1 = Entry(root)

bookInfo1.place(relx=0.27,rely=0.53, relwidth=0.45, relheight=0.067)

SubmitBtn = Button(root,text="Return",bg='#d1ccc0', fg='black',command=returnn)

SubmitBtn.place(relx=0.28,rely=0.9, relwidth=0.18,relheight=0.08)

quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)

quitBtn.place(relx=0.53,rely=0.9, relwidth=0.18,relheight=0.08)

root.mainloop()

## ViewBook.py ##

from tkinter import \*

from PIL import ImageTk,Image

from tkinter import messagebox

import pymysql

mypass = "Aadi2606"

mydatabase="db"

con = pymysql.connect(host="localhost",user="root",password=mypass,database=mydatabase)

cur = con.cursor()

bookTable = "books"

def View():

root = Tk()

root.title("Library")

root.geometry("600x500")

background = PhotoImage(file = "F:\\12th project\\img\\view\_books.png")

label4 = Label( root, image = background)

label4.place(x = 0, y = 0)

y = 0.35

Label(root, text = "----------------------------------------------------------------------------",bg='black',fg='white').place (relx=0.06,rely=0.5)

getBooks = "select \* from "+bookTable

try:

cur.execute(getBooks)

con.commit()

for i in cur:

Label(root,text="%-10s%-30s%-30s%-20s"%(i[0],i[1],i[2],i[3]) ,bg='black', fg='white').place(relx=0.07,rely=y)

y += 0.1

except:

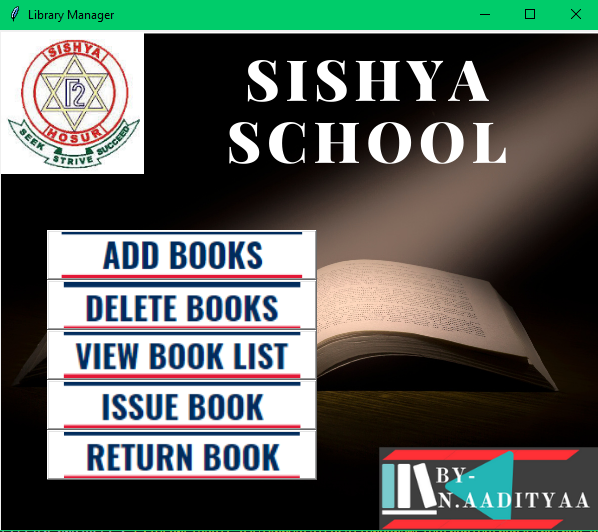
messagebox.showinfo("Failed to fetch files from database")

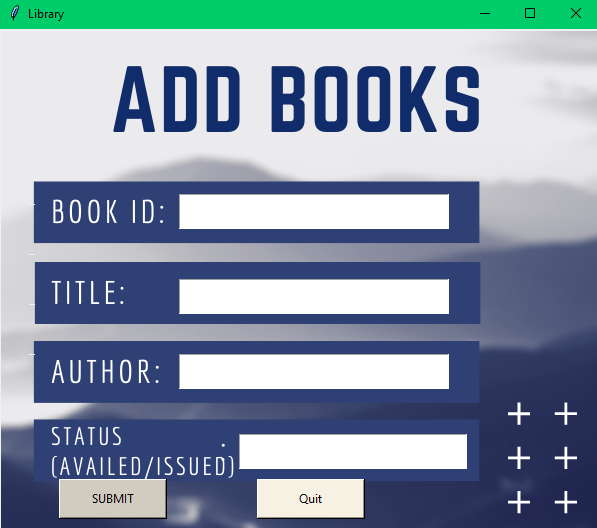
quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)

quitBtn.place(relx=0.4,rely=0.9, relwidth=0.18,relheight=0.08)

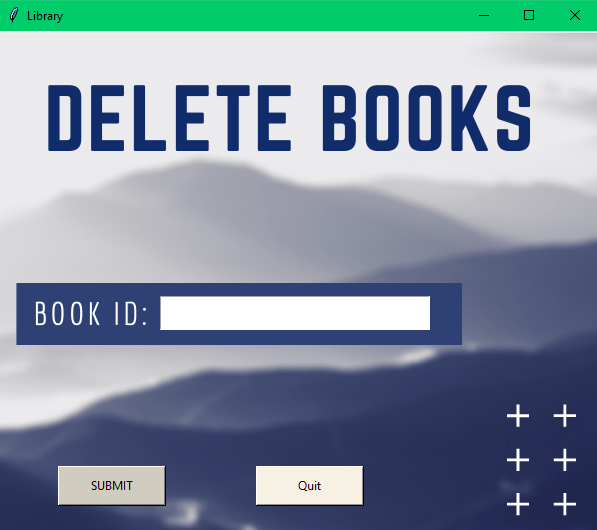
root.mainloop()

**OUTPUT**

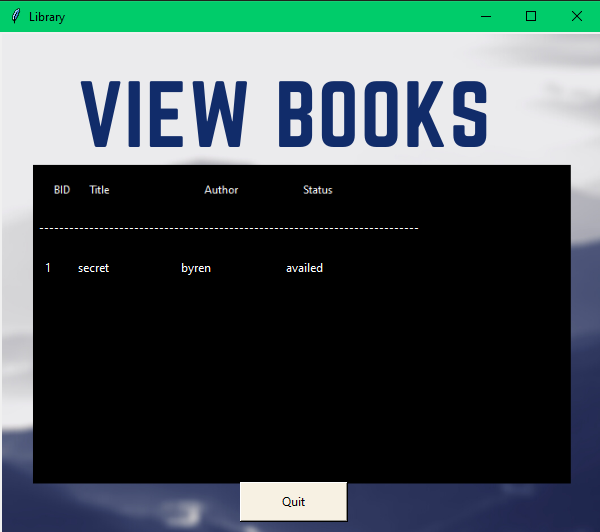


When the button “ADD BOOKS” clicked

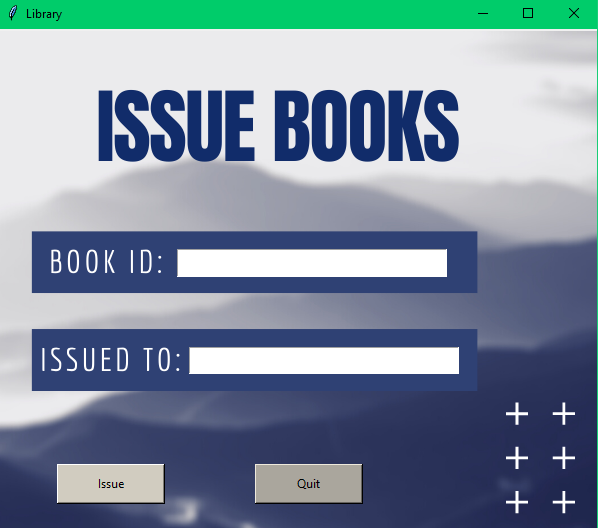
When the button “DELETE BOOKS” clicked



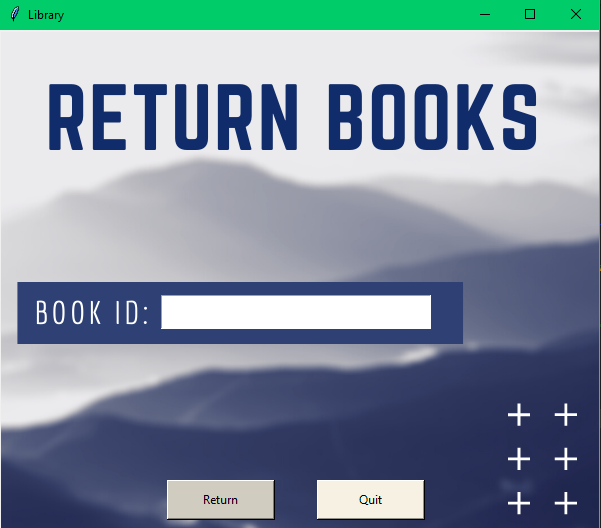
When the button “VIEW BOOK LIST” clicked



When the button “ISSUE BOOK” clicked

****

When the button “RETURN BOOK” clicked

****

**FUTURE ENHANCEMENT**

The program can be enhanced by the following ways:-

* An additional feature can be added like a camera so that the user need not enter the book id each time to add book instead can show thee book through the camera and the program would deduct the title of the book.
* A virtual library can be created where all book’s front page image can be displayed.
* The program can be also made to be used in mobile phones and not only on windows.
* Another additional feature can be added like when book is issued to a person and once the due date is reached a automatic message can be sent to the person informing him/her to return the book.
* The program should also calculate the fine to be paid due to delay of submission.
* Instead of book id the program can read a QR code so that each book would have unique QR code and so each time if the book is rented or returned or added the QR code can be scanned this would make the proses quick and efficient

**REFERENCES**

Textbook Reference:

* Computer science with python for class XII by Sumitra Arora

Website Reference:

* <https://en.wikipedia.org/wiki/Wikipedia>
* <https://stackoverflow.com/>
* <https://www.techwithtim.net/>

Video Reference:

* <https://youtu.be/YXPyB4XeYLA>
* <https://youtube.com/playlist?list=PLB5jA40tNf3tRMbTpBA0N7lfDZNLZAa9G>
* <https://youtu.be/3vsC05rxZ8c>