



LOVELY
PROFESSIONAL
UNIVERSITY

SIX WEEKS SUMMER TRAINING REPORT

On

**“LPU - Learn Advanced Programming
using Python ”**

Submitted by:

Harsha Vardhan Tamarana

Registration No: 12005900

Programme Name : BTech(CSE).

Under the Guidance of E-box

School of Computer Science & Engineering
Lovely Professional University , Phagwara
(May –Aug 2022)

DECLARATION

I here by declare that I have completed my twelve weeks summer training at E- box from May 2022 to July,2022. I have declare that I have worked with full dedication during these six weeks of training and my learning outcomes full fill the requirements of training for the award of degree of B.Tech(CSE) Lovely Professional University, Phagwara.

`Signature of Student

Name: Harsha Vardhan Tamarana

Reg.No: 12005900

Start Date : May 2022

End Date : Aug 2022

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to my mentor whose contribution in stimulating suggestions and encouragement helped me to coordinate my project especially in writing this report. I express my thanks to my institution Lovely Professional University for giving me an opportunity to learn this interesting topic. I also convey my regards to my faculty assistance all through this training named “Object Oriented Programming”. Once again I would like to thank all my supporters from the core of my heart.

Training certificate from organization

INTRODUCTION TO PYTHON

Python Language Introduction

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently. 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.

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.

TRAINING CONTENTS

1. Introduction to Python

Learn how to install Python, distinguish between important data types and use basic features of the Python interpreter, IDLE.

2. Using Variables in Python

Learn about numeric, string, sequence and dictionary data types and relevant operations while practicing Python syntax.

3. Basics of Programming in Python

Learn how to write programs using conditionals, loops, iterators and generators, functions and modules and packages.

4. Principles of Object-oriented Programming (OOP)

Learn about the important features of Object-oriented Programming while using Classes and Objects, two main aspects of the OOP paradigm.

5. Connecting to SQLite Database

Learn about relational databases while learning how to store and retrieve data from an SQLite database through Python.

6. Developing a GUI with PyQt

Learn how to install PyQt5 toolkit, Qt Designer and create a graphical user interface using common widgets and menu systems.

7. Application of Python in Various Disciplines

Learn about various resources to extend your learning for the Python programming language.

PROBLEM IDENTIFICATION AND CAUSE OF THE PROBLEM

management system is the framework that enables companies to achieve their operational and business objectives through a process of continuous improvement. In its simplest form, a management system implements the Plan, Do, Check, Act/Adjust cycle. Several choices are available for management systems (ISO is commonly applied), whether they are certified by third-party registrars and auditors, self-certified, or used as internal guidance and for potential certification readiness.

Business Benefits of a Well-Documented Management System

The connection between management systems and compliance is vital in avoiding recurring compliance issues and in reducing variation in compliance performance. In fact, reliable and effective regulatory compliance is commonly an outcome of consistent and reliable implementation of a management system.

Beyond that, there are a number of business reasons for implementing a well-documented management system (environmental, safety, quality, food safety, other) and associated support methods and tools:

Establishes a common documented framework to achieve more consistent implementation of compliance policies and processes—addressing the eight core functions of compliance:

Inventories

Permits and authorizations

Plans

Training

Practices in place

Monitoring and inspection

Records

Reporting

OBJECTIVE TO BE ACHIEVED

Create a Book Management System in Python. The should should have all the features displayed in the mock-up screens in the scenario.

Features

- > Add Book**
- > Delete Student**
- > Update Book**
- > View All Book**

LIBRARIES AND MODULES

Tkinter

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets

- Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.
- There are currently 15 types of widgets in Tkinter.

these widgets as well as a brief description in the following table –

Sr.No.	Operator & Description
1	<u>Button</u> The Button widget is used to display buttons in your application.
2	<u>Canvas</u> The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	<u>Checkbutton</u>

	<p>The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.</p>
4	<p><u>Entry</u></p> <p>The Entry widget is used to display a single-line text field for accepting values from a user.</p>
5	<p><u>Frame</u></p> <p>The Frame widget is used as a container widget to organize other widgets.</p>
6	<p><u>Label</u></p> <p>The Label widget is used to provide a single-line caption for other widgets. It can also contain images.</p>
7	<p><u>Listbox</u></p> <p>The Listbox widget is used to provide a list of options to a user.</p>
8	<p><u>Menubutton</u></p> <p>The Menubutton widget is used to display menus in your application.</p>
9	<p><u>Menu</u></p> <p>The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.</p>
10	<p><u>Message</u></p> <p>The Message widget is used to display multiline text fields for accepting values from a user.</p>
11	<p><u>Radiobutton</u></p> <p>The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.</p>
12	<p><u>Scale</u></p> <p>The Scale widget is used to provide a slider widget.</p>
13	<p><u>Scrollbar</u></p> <p>The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.</p>
14	<p><u>Text</u></p> <p>The Text widget is used to display text in multiple lines.</p>
15	<p><u>Toplevel</u></p>

	The Toplevel widget is used to provide a separate window container.
16	<u>Spinbox</u> The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	<u>PanedWindow</u> A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	<u>LabelFrame</u> A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	<u>tkMessageBox</u> This module is used to display message boxes in your applications.

Standard attributes

Let us take a look at how some of their common attributes. such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- **The *pack()* Method** – This geometry manager organizes widgets in blocks before placing them in the parent widget.
- **The *grid()* Method** – This geometry manager organizes widgets in a table-like structure in the parent widget.
- **The *place()* Method** – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

Random import

Sometimes we want the computer to pick a random number in a given range, pick a random element from a list, pick a random card from a deck, flip a coin, etc. The random module provides access to functions that support these types of operations. The **random** module is another library of functions that can extend the basic features of python. Other modules we have seen so far are **string**, **math**, **time** and **graphics**. With the exception of the graphics module, all of these modules are built into python. To get access to the **random** module, we add **from random import *** to the top of our program (or type it into the python shell).

Date time

In Python, date and time are not a data type of their own, but a module named datetime can be imported to work with the date as well as time. Python Datetime module comes built into Python, so there is no need to install it externally. Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

Tk message box

The `tkMessageBox` module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

Some of these functions are `showinfo`, `showwarning`, `showerror`, `askquestion`, `askokcancel`, `askyesno`, and `askretryignore`.

Here is the simple syntax to create this widget –

```
tkMessageBox.FunctionName(title, message [, options])
```

Parameters

- **FunctionName** – This is the name of the appropriate message box function.
- **title** – This is the text to be displayed in the title bar of a message box.
- **message** – This is the text to be displayed as a message.
- **options** – options are alternative choices that you may use to tailor a standard message box. Some of the options that you can use are `default` and `parent`. The `default` option is used to specify the default button, such as `ABORT`, `RETRY`, or `IGNORE` in the message box. The `parent` option is used to specify the window on top of which the message box is to be displayed.

You could use one of the following functions with dialogue box –

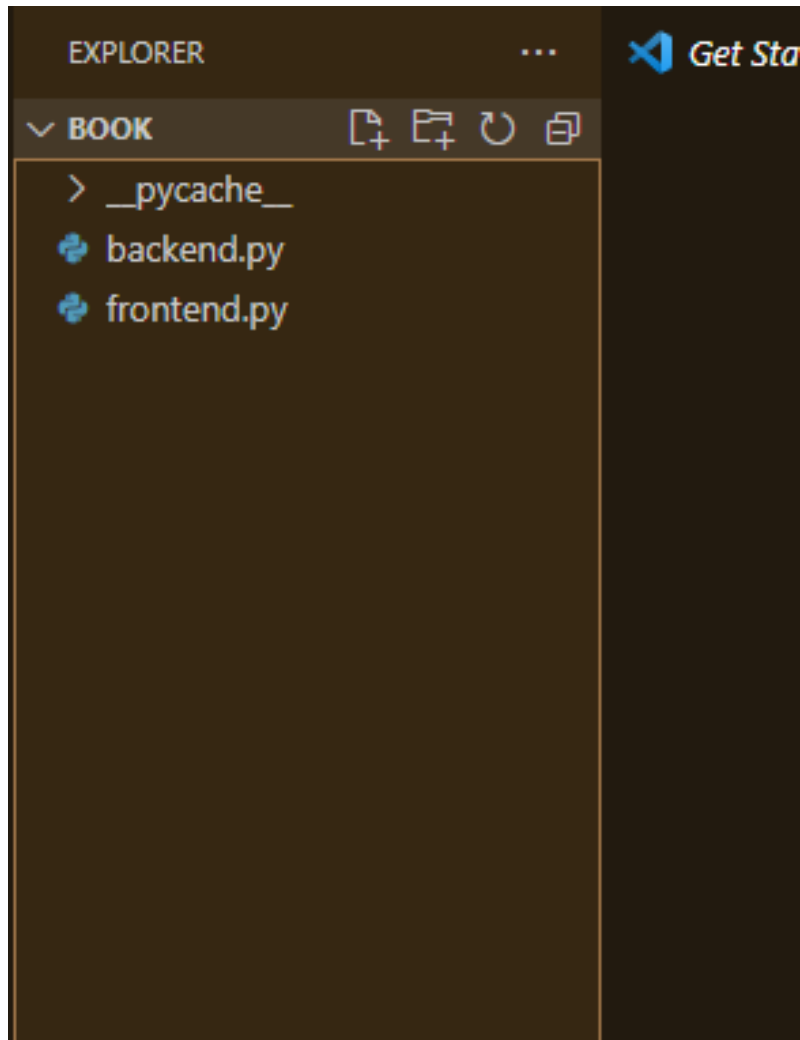
- `showinfo()`
- `showwarning()`
- `showerror ()`
- `askquestion()`
- `askokcancel()`
- `askyesno ()`
- `askretrycancel ()`

Database sqlite3

SQLite is an in-process library that implements a [self-contained, serverless, zero-configuration, transactional](#) SQL database engine. The code for SQLite is in the [public domain](#) and is thus free for use for any purpose, commercial or private. SQLite is the [most widely deployed](#) database in the world with more applications than we can count, including several [high-profile projects](#).

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database [file format](#) is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between [big-endian](#) and [little-endian](#) architectures. These features make SQLite a popular choice as an [Application File Format](#). SQLite database files are a [recommended storage format](#) by the US Library of Congress. Think of SQLite not as a replacement for [Oracle](#) but as a replacement for [fopen\(\)](#)

Project Structure




CODE

LANDING PAGE

The screenshot shows a Java Swing window titled "Bookstore". The window has a standard title bar with a close button. The main content area is divided into several sections. On the left, there are two input fields labeled "Title" and "Year". To the right of these are two more input fields labeled "Author" and "ISBN". Below the "Title" and "Year" fields is a large, empty text area. To the right of this text area are two small, light gray buttons with upward and downward arrows. On the far right side of the window, there is a vertical stack of five buttons: "View all", "Search entry", "Add entry", "Update", and "Delete". At the bottom right corner, there is a "Close" button.

ADD BOOK



The screenshot shows a Java Swing window titled "Bookstore". The window has a standard title bar with a close button (X) and a maximize button (square). The main content area is divided into two parts. On the left, there is a table with two rows and two columns. The first row has "Title" and "The earth", and the second row has "Year" and "1910". On the right, there are two labels with text fields: "Author" with "The earth" and "ISBN" with "987654321". Below these fields is a large empty rectangular area. To the right of this area are five buttons stacked vertically: "View all", "Search entry", "Add entry", "Update", and "Delete". At the bottom right, there is a "Close" button. The window is styled with a light gray background and black text.

Title	Author
The earth	The earth
Year	ISBN
1910	987654321

View all

Search entry

Add entry

Update

Delete

Close

BOOK ADDED

Bookstore

Title	The earth	Author	1910
Year	987654321	ISBN	

{The earth} {The earth} 1910 987654321

View all

Search entry

Add entry

Update

Delete

Close

ADD MORE ENTRY

Bookstore

Title	The earth2	Author	The earth2
Year	1912	ISBN	987654320

1 {The earth} {The earth} 1910 987654321

2 {The earth2} {The earth2} 1912 987654320

View all

Search entry

Add entry

Update

Delete

Close

UPDATE BOOK

Bookstore

Title

The earth update

Author

The earth

Year

1910

ISBN

987654321

1 {The earth update} {The earth} 1910 987654321

2 {The earth2} {The earth2} 1912 987654321

^

v

View all

Search entry

Add entry

Update

Delete

Close

UPDATE SUCCESS

Bookstore

Title

The earth update

Author

The earth

Year

1910

ISBN

987654321

1 {The earth update} {The earth} 1910 987654321

2 {The earth2} {The earth2} 1912 987654321

^

v

View all

Search entry

Add entry

Update

Delete

Close

DELETE BOOK

Bookstore

Title

The earth update

Author

The earth

Year

1910

ISBN

987654321

1 {The earth update} {The earth} 1910 987654321

2 {The earth2} {The earth2} 1912 987654320

^

v

View all

Search entry

Add entry

Update

Delete

Close

DELETE SUCCESS

Bookstore

Title

The earth2

Author

The earth2

Year

1912

ISBN

987654320

2 {The earth2} {The earth2} 1912 987654320

^

v

View all

Search entry

Add entry

Update

Delete

Close

SEARCH BY AUTHOR NAME

Bookstore

Title

Year

Author

ISBN

The earth3

3 {The earth3} {The earth3} 1912 9876543

^

v

View all

Search entry

Add entry

Update

Delete

Close

SEARCH SUCCESS

Bookstore

Title

Year

Author

ISBN

The earth

^

v

View all

Search entry

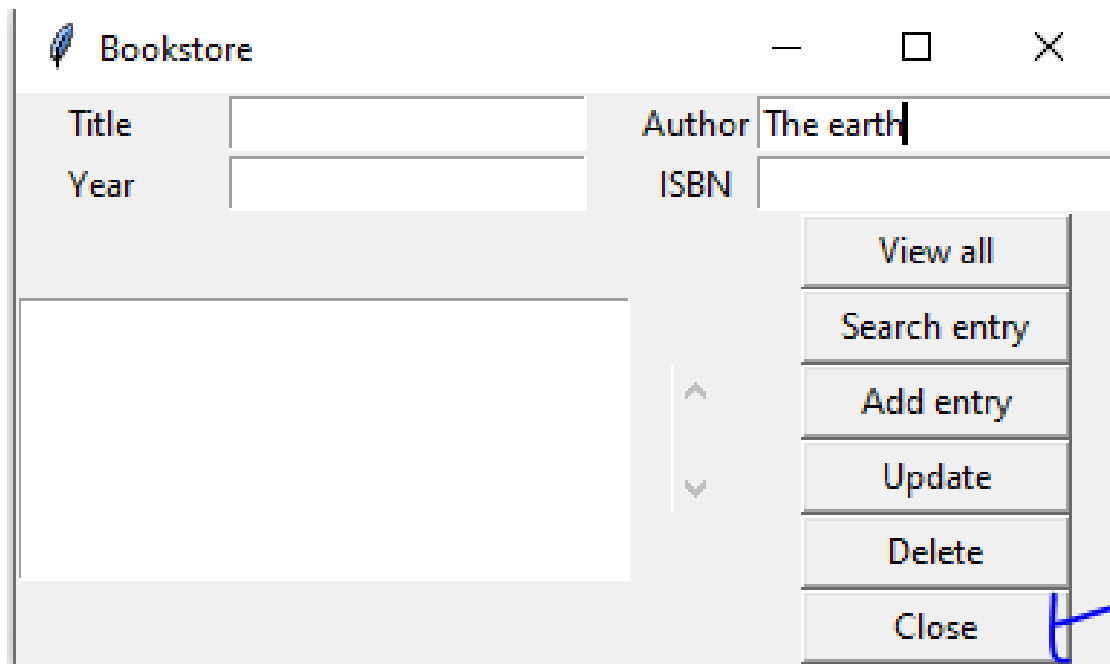
Add entry

Update

Delete

Close

CLOSE BUTTON

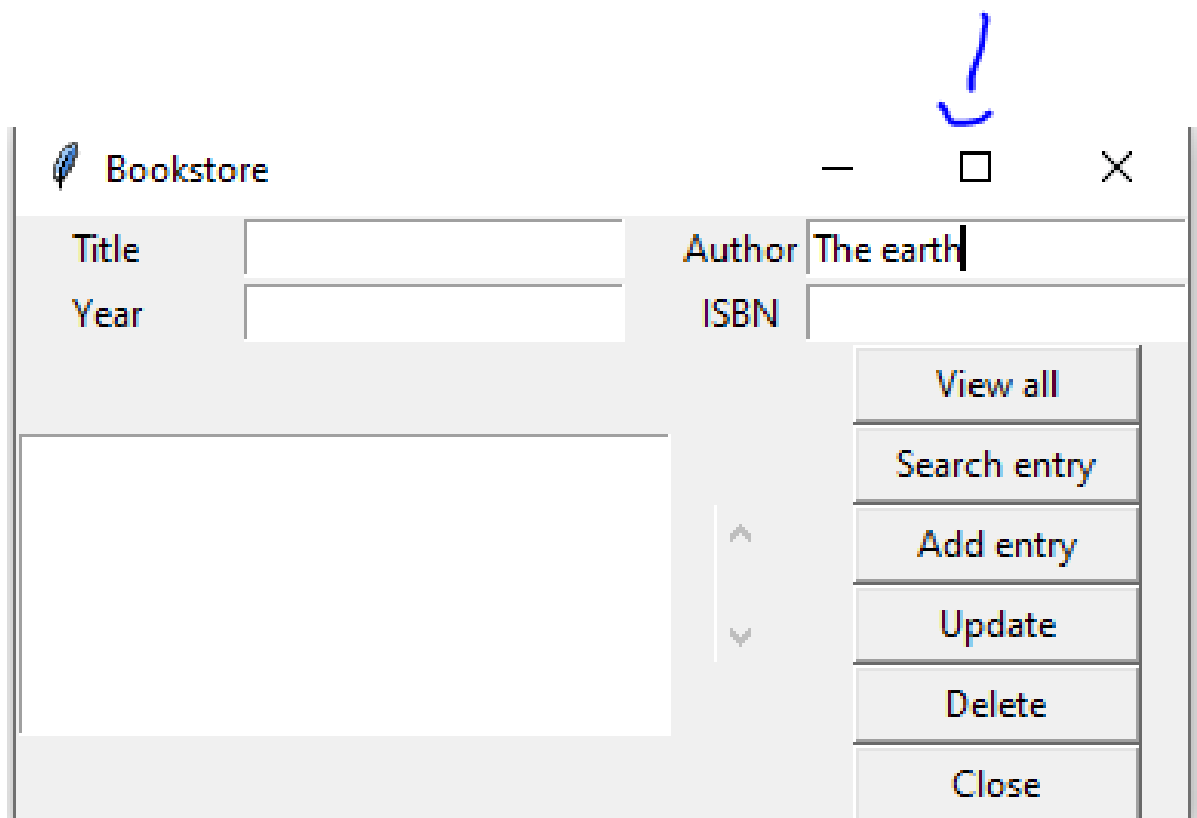


The screenshot shows a window titled "Bookstore" with a feather icon. The window contains a form with fields for "Title", "Year", "Author", and "ISBN". The "Author" field is filled with "The earth". Below the form is a large empty rectangular area. To the right of this area are two small arrows, one pointing up and one pointing down. On the far right, there is a vertical stack of six buttons: "View all", "Search entry", "Add entry", "Update", "Delete", and "Close". A blue line points to the "Close" button.

Title		Author	The earth
Year		ISBN	

View all
Search entry
Add entry
Update
Delete
Close

MAXIMIZE WINDOW



The screenshot shows the same "Bookstore" window as before, but the "Maximize" button (represented by a square icon) in the window's title bar is highlighted with a blue arrow. The rest of the window's content is identical to the previous screenshot.

Title		Author	The earth
Year		ISBN	

View all
Search entry
Add entry
Update
Delete
Close

CODE

backend.py

SETUP DB

```
import sqlite3

def connect():
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, title text, author text, year integer, isbn integer)")
    conn.commit()
    conn.close()
```

INSERT

```
✓ def insert(title,author,year,isbn):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("INSERT INTO book VALUES (NULL,?,?,?,?,?)",(title,author,year,isbn))
    conn.commit()
    conn.close()
```

VIEW BOOK

```
def view():  
    conn=sqlite3.connect("books.db")  
    cur=conn.cursor()  
    cur.execute("SELECT * FROM book")  
    rows=cur.fetchall()  
    conn.close()  
    return rows
```

SEARCH

```
def search(title="",author="",year="",isbn=""):  
    conn=sqlite3.connect("books.db")  
    cur=conn.cursor()  
    cur.execute("SELECT * FROM book WHERE title=? OR author=? OR year=? OR isbn=?",  
                (title,author,year,isbn))  
    rows=cur.fetchall()  
    conn.close()  
    return rows
```


DELETE BY ID

```
def delete(id):  
    conn=sqlite3.connect("books.db")  
    cur=conn.cursor()  
    cur.execute("DELETE FROM book WHERE id=?", (id,))  
    conn.commit()  
    conn.close()
```

UPDATE BOOK

```
def update(id,title,author,year,isbn):  
    conn=sqlite3.connect("books.db")  
    cur=conn.cursor()  
    cur.execute("UPDATE book SET title=?, author=?,year=?,isbn=? WHERE id=?", (title,author,year,isbn,id))  
    conn.commit()  
    conn.close()
```

CONNECT DB

```
connect()  
#insert("The earth","The earth",1910,987654321)  
#delete(4)  
#update(1,"THE Moon","Dork",2000,564789123)  
#print(view())  
#print(search(author="Dork"))
```

IMPORT BACKEND AND GUI

```
from tkinter import *  
import backend
```

SETUP GUI

```
def get_selected_row(event):  
    try:  
        global selected_tuple  
        index=list1.curselection()[0]  
        selected_tuple=list1.get(index)  
        e1.delete(0,END)  
        e1.insert(END,selected_tuple[1])  
        e2.delete(0,END)  
        e2.insert(END,selected_tuple[2])  
        e3.delete(0,END)  
        e3.insert(END,selected_tuple[3])  
        e4.delete(0,END)  
        e4.insert(END,selected_tuple[4])  
    except IndexError:  
        pass
```

SETUP COMMAND

```
def view_command():  
    list1.delete(0,END)  
    for row in backend.view():  
        list1.insert(END,row)
```

SEARCH COMMAND

```
def search_command():  
    list1.delete(0,END)  
    for row in backend.search(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()):  
        list1.insert(END,row)  
  
def add_command():  
    list1.delete(0,END)
```

ADD COMMAND

```
def add_command():  
    list1.delete(0,END)  
    backend.insert(title_text.get(),author_text.get(),year_text.get(),isbn_text.get())  
    list1.insert(END,(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()))
```

```
def delete_command():
```

DELETE COMMAND

```
def delete_command():  
    backend.delete(selected_tuple[0])
```

UPDATE COMMAND

```
def update_command():  
    backend.update(selected_tuple[0],title_text.get(),author_text.get(),year_text.get(),isbn_text.get())
```

GUI

```
window=Tk()

window.wm_title("Bookstore")
l1=Label(window,text="Title")
l1.grid(row=0,column=0)

title_text=StringVar()
e1=Entry(window,textvariable=title_text)
e1.grid(row=0,column=1)

l2=Label(window,text="Author")
l2.grid(row=0,column=2)

author_text=StringVar()
e2=Entry(window,textvariable=author_text)
e2.grid(row=0,column=3)
```

LABEL

```
l3=Label(window,text="Year")
l3.grid(row=1,column=0)

year_text=StringVar()
e3=Entry(window,textvariable=year_text)
e3.grid(row=1,column=1)

l4=Label(window,text="ISBN")
l4.grid(row=1,column=2)

isbn_text=StringVar()
e4=Entry(window,textvariable=isbn_text)
e4.grid(row=1,column=3)
```

BUTTON

```
b1=Button(window,text="View all",width=12,command=view_command)
b1.grid(row=2,column=3)

b2=Button(window,text="Search entry",width=12,command=search_command)
b2.grid(row=3,column=3)

b3=Button(window,text="Add entry",width=12,command=add_command)
b3.grid(row=4,column=3)

b4=Button(window,text="Update",width=12,command=update_command)
b4.grid(row=5,column=3)

b5=Button(window,text="Delete",width=12,command=delete_command)
b5.grid(row=6,column=3)

b6=Button(window,text="Close",width=12,command=window.destroy)
b6.grid(row=7,column=3)
```

LIST

```
list1=Listbox(window,height=6,width=35)
list1.grid(row=2,column=0,rowspan=6,columnspan=2)

sb1=Scrollbar(window)
sb1.grid(row=2,column=2,rowspan=6)

list1.configure(yscrollcommand=sb1.set)
sb1.configure(command=list1.yview)

list1.bind('<<ListboxSelect>>',get_selected_row)
window.mainloop()
```

CONCLUSION

1. Increase efficiency

Maintaining daily statistics on the total number of volumes issued, reissued, unreturned, and available can be a time-consuming task for a librarian. A School library management system improves the efficiency of a library's whole life cycle by allowing all tasks to be completed with a single click, making a librarian's job easier.

Students can browse the catalog, their book status, and other information by signing into their accounts. Not only that, but they can be reminded by SMS about due dates for returning books, notifications to pay late fines, and so on.

2. Maintain data

Data is an essential component of any educational institution, and library books are a valuable asset. Manual data management entails hazards such as data misplacement, data entry errors, and so on. We must properly document all library data to be easily accessible. They can save the whole catalog, details of books issued, reissued, unreturned, and available in a single system, and retrieve it anytime needed with a few simple clicks by employing a library management system. They lower the chance of data loss while simultaneously ensuring security.

3. Increase productivity

Having such management systems in libraries can greatly streamline the overall business. With book records available with a single click, a dashboard for real-time analysis, and direct communication with students, the system can handle the majority of the activities, saving the team a significant amount of time. As a result, they may devote their attention to more important tasks.

4. Time-saving

The traditional method of managing library activities might be time-consuming. During exam season, the number of students accessing library books increases, which may cause students to wait even longer than usual. Using Library Management Software can be extremely beneficial at this time. The library team can promptly issue books to pupils while also using their track record to distribute the books effectively. Students can also check the catalog to determine if the book they require is currently available. All of this can save both students and the institute library personnel a lot of time.

5. Cost-effective

For technology investments, educational institutions have a set budget. As a result, some institutes may consider such systems to be a large financial investment. However, it is a very cost-effective alternative in practice. Managing a library necessitates a crew that cannot guarantee seamless operation, whereas these solutions are a one-time investment. They save money on labor, have low maintenance costs, and are more efficient and effective. With so many possibilities on the market, they may select the best solution for them.

REFERENCE

- <https://docs.python.org/3/>
- <https://www.sqlite.org/index.html>
- <https://www.sqlitetutorial.net/sqlite-python/>