Advanced Topics

Unit-8

Contents

Working with Database and Using SQL Statements

Basics of GUI Programming

Basics of Web Development.

Introduction

In today's world, where the volume of information is rapidly growing, its effective processing becomes a key point in any field of human activity. In this context, databases act as a repository, allowing information not only to be stored but also to be managed efficiently.

SQL (Structured Query Language) plays a significant role in the organization of databases. This query language provides a convenient way to interact with data, allowing you to create, modify, and retrieve information. The combination of Python and SQL provides convenient tools for automating processes and will enable developers to create flexible and scalable applications.

Working with SQL in Python is done through database management systems (DBMS). There are many DBMS, each with its own characteristics and purposes.

From classical relational databases such as MySQL and PostgreSQL to NoSQL solutions such as MongoDB, the choice of a suitable DBMS depends on the specific project requirements.

- SQLite
- MySQL
- PostgreSQL

Installing the Libraries

Before working with SQL databases in Python, you need to install the appropriate libraries. Each database has its own package; let's look at some of them.

To work with **SQLite**, you need the sqlite3 library. This library is included in the standard Python library, so there's no need to install it separately. However, for working with MySQL and PostgreSQL, we need to install the libraries.

For **MySQL**:

pip install mysql-connector-python

After installing the appropriate libraries, we will be able to establish a connection to the SQL database directly from the Python runtime environment.

A lightweight SQL library called SQLite is supplied with Python. SQLite is fast since it is written in C. Furthermore, it produces the database in a single file, which simplifies implementation and eliminates the need to manage several servers' worth of problems. It does, however, imply that SQLite is more appropriate for tiny, standalone apps or for use in development. If your Python software is going to be used on large-scale systems, you should switch to a more reliable database, such PostgreSQL or MySQL.

Despite aside, SQLite is still valuable. Before implementing a full-fledged database, it's beneficial to prototype your application. This ensures that your software functions as intended and that any issues are most likely related to the DB implementation. Small programs that don't require a full database package and all its overhead can also benefit from it.

Installing this module is not necessary as it comes installed by default with Python versions 2.5.x and above.

The following code demonstrate how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.

```
# connecting SQLite Database

import sqlite3

conn = sqlite3.connect('test.db')

print ('Opened database successfully.')
```

OUTPUT

```
Opened database successfully.
```

More Examples

Create

Insert

Update

Delete

We will see their example practically

MySQL Database

MySQL is a popular open-source relational database management system (RDBMS). It is widely used for its reliability, ease of use, and performance. MySQL is a central component of the LAMP (Linux, Apache, MySQL, PHP/Perl/Python) stack, which is a popular web development platform.

Key Features of MySQL

- **Open-Source:** MySQL is open-source software, which means it is free to use, modify, and distribute.
- **Cross-Platform:** MySQL can run on various operating systems, including Linux, Windows, and macOS.
- **High Performance:** MySQL is known for its speed and efficiency in handling large databases.
- **Scalability:** MySQL can handle large amounts of data and support numerous concurrent users.
- **Security:** MySQL provides robust security features, including user authentication and encryption.
- **Replication:** MySQL supports master-slave replication, allowing data to be copied from one server to another.
- **Support for Large Databases:** MySQL can handle databases in the terabyte range.
- **Transactions and ACID Compliance:** MySQL supports transactions and ensures data integrity with ACID (Atomicity, Consistency, Isolation, Durability) compliance.

1. First Install MySQL

   After installing MySQL we cannot connect it to python but we can use it from terminal, to create database and create table

2. Install MySQL Connector/Python

   A database driver is a piece of software that allows an application to connect and interact with a database system. Programming languages like Python need a special driver before they can speak to a database from a specific vendor.

3. Connecting to Database

We will learn how to install **MySQL**, install **MySQL connector** and **connect python code with MySQL using connector.**

Then ,

We will see example of :

- Connecting MySQL from python code
- Create table
- Insert into table
- Update table
- Delete from table
- Fetch from table

Note: We will see example of these practically

Basic of GUI Programming in Python

A graphical user interface (GUI) is a desktop interface that allows you to communicate with computers.

Python has a variety of libraries, but these four stand out, especially in terms of GUI.

- Tkinter
- Kivy
- Python QT
- wxPython

Out of all the GUI methods, tkinter is the most commonly used method.

Tkinter is the first option for a lot of learners and developers because it is quick and convenient to use.

Tkinter is a Python library that can be used to construct basic graphical user interface (GUI) applications.

In Python, it is the most widely used module for GUI applications.

Remember : The tkinter library is primarily designed for creating desktop graphical user interfaces (GUIs) in Python and is not suitable for web development.

Building Your First Python GUI Application With Tkinter

To create a Tkinter Python app, you follow these basic steps:

Import the tkinter module: This is done just like importing any other module in Python. Note that in Python 2.x, the module is named ‘Tkinter’, while in Python 3.x, it is named ‘tkinter’.

Create the main window (container): The main window serves as the container for all the GUI elements you’ll add later.

Add widgets to the main window: You can add any number of widgets like buttons, labels, entry fields, etc., to the main window to design the interface as desired.

Apply event triggers to the widgets: You can attach event triggers to the widgets to define how they respond to user interactions.

Let’s install tkinter module and create basic GUI application:

To install tkinter in python go to command prompt and

- pip install tkinter

After that we can create basic gui application using tkinter

First create window using following command

To create a main window, tkinter offers a method ‘Tk(screenName=None, baseName=None, className=’Tk’, useTk=1)’. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

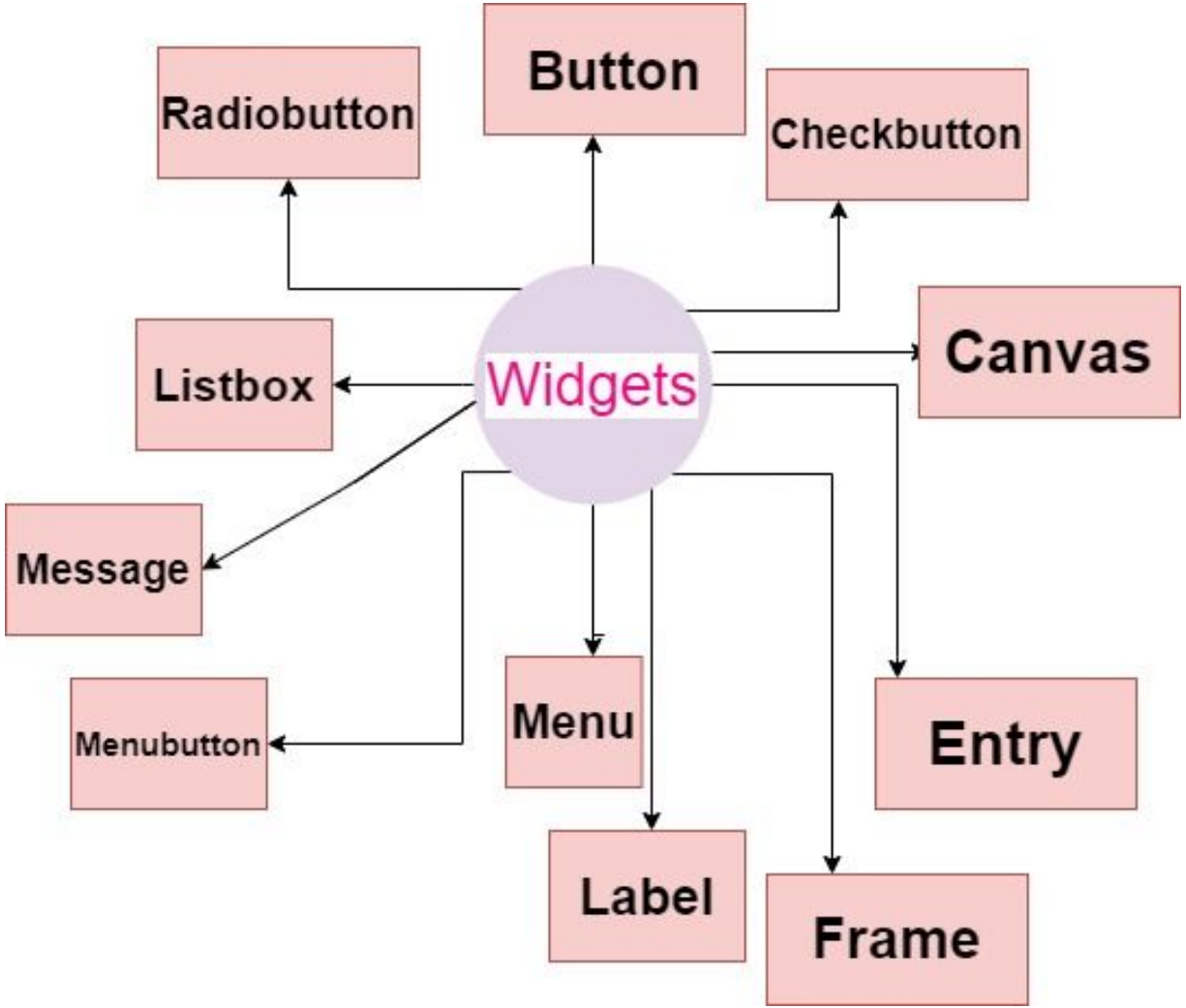
```
import tkinter

window = tkinter.Tk()

window.mainloop()
```

Note: There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur, and process the event as long as the window is not closed.

Widgets


Radiobutton
Button
Checkbutton
Listbox
Widgets
Canvas
Message
Menubutton
Menu
Entry
Label
Frame

Basic tkinter widgets

- Label: Display text or images.
- Button: Create a clickable button to trigger a function or action.
- Entry: Single-line text entry widget for user input.
- Frame: Container widget to group other widgets.
- Checkbutton: Create a checkbox that the user can toggle on or off.
- Radiobutton: Create a set of mutually exclusive options (radio buttons).
- Listbox: Display a list of items for the user to select from.
- Scrollbar: Add vertical or horizontal scrolling to another widget.
- Canvas: Draw shapes, lines, or display images.
- Menu: Create menus and menu items for the application.

Text: Multi-line text entry widget for more extensive text input.

Scale: Create a slider for selecting a numerical value within a range.

Combobox: Provide a dropdown list with editable text entry

Spinbox: Allow the user to select from a fixed set of values or type a value.

Toplevel: Create additional top-level windows separate from the main window.

Menubutton : Display a menu when clicked.

Message: Display multi-line text, similar to Label but for longer messages.

OptionMenu: Create a drop-down menu for selecting options.

Progressbar (requires ttk module) : Display a progress bar to indicate the progress of a task.

Advanced Tkinter widgets

ScrolledText: Create a text widget with built in scrollbars

Treeview: Display hierarchical data in a tree like structure

MessageBox: Display dialog boxes for messages, warning

Treeview Scrollbar: Add scrollbar to Treeview widgets

Others

LabelFrame : A frame with a label, used as a container for related widgets.

PanedWindow : Create a container with resizable panes.

Geometry management

Creating a new widget doesn’t mean that it will appear on the screen. To display it, we need to call a special method: **pack**, **grid**, and **place**. Each of these managers offers different ways to control the layout of widgets.

pack

The pack geometry manager organizes widgets in blocks before placing them in the parent widget.

Suitable for simple layouts where widgets are stacked horizontally or vertically.

grid

The grid geometry manager organizes widgets in a table-like structure.

Suitable for more complex layouts where widgets need to be placed in rows and columns.

place

The place geometry manager places widgets at an absolute position you specify.

Suitable for precise control over widget placement, using absolute positioning.

Let’s see example of various widget