CHAPTER-4

DEVELOPMENT OF Graphical User Interface (GUI)

www.ilmkidunya.com

Introduction

In this chapter, students will learn to develop user-friendly Graphical User Interfaces (GUIs) using Tkinter, Python's built-in GUI toolkit. They will create windows, frames, labels, buttons, entry fields, and manage layouts using pack(), grid(), and place(). The chapter also introduces database concepts, SQL, Python database connectivity, and basic CRUD operations using SQLite or MySQL.

4.1 Graphical User Interface (GUI) Development with Tkinter

Ex.Q.1 Explain the concept of Graphical User Interface (GUI) and discuss its importance in application development.

Graphical User Interface (GUI)

A Graphical User Interface (GUI) is a visual interface that allows users to interact with software by using windows, buttons, labels, text boxes, menus, and forms.

GUI Development

GUI development is the process of creating applications with visual elements such as windows, buttons, menus, and text boxes. GUI development allows programs to interact with users through visual elements instead of commands. GUI makes applications more user-friendly, attractive, and easier to operate.

Python provides tools to develop GUI applications easily.

Tkinter is Python's built-in library for creating GUI programs. It is simple, widely used, and helps developers create interactive applications quickly.

Short Question

What is GUI development?

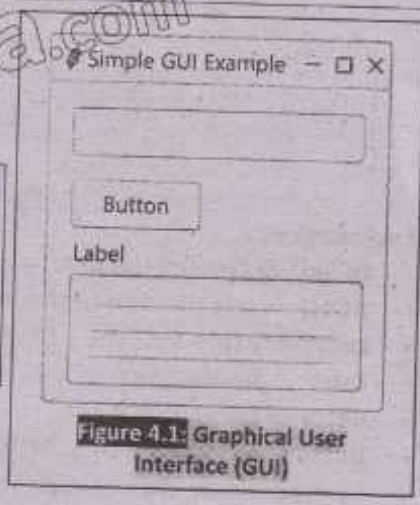


Figure 4.1: Graphical User Interface (GUI)

Importance of GUI in Application development/ Why is GUI Important?

i. **Easy to Understand and Use**
   A GUI makes software easy to understand and use, especially for non-technical users.
ii. **Easy Interaction**
   It allows users to interact with programs by clicking buttons, selecting menus, and filling forms instead of typing commands.
iii. **Visual Feedback**
   A GUI provides visual feedback, making it easier for users to operate software correctly.
iv. **Widely Used**
   GUI applications are widely used in desktop software, web applications, and mobile app

Ex.Q.2 Describe Tkinter as Python's built-in GUI toolkit.

Tkinter

Tkinter is Python's standard GUI toolkit that comes pre-installed with the language.
It provides a collection of classes and functions to create GUI applications. It allows programmers to create windows, buttons, labels, entry fields, and other widgets.

Features of Tkinter

- It is easy to learn and lightweight.
- It supports event-driven programming, where actions are performed when the user interacts with the application.
- It is suitable for developing small to medium-sized GUI applications quickly and efficiently.

Example:

```
import tkinter as tk
# Create the main window
window = tk.Tk()
window.title("Simple GUI Example")
window.geometry("300x200")
# Function to handle button click
def on_click():
    label.config(text="Button Clicked!")
# Create a label
label = tk.Label(window, text="Welcome to Tkinter")
label.pack(pady=10)
# Create an entry widget
entry = tk.Entry(window)
entry.pack(pady=10)
# Create a button
button = tk.Button(window, text="Click Me", command=on_click)
button.pack(pady=10)
# Start the GUI application
window.mainloop()
```

Explanation:

- **tk.Tk()** → creates the main application window.
- **title()** → sets the title of the window.
- **geometry ()** → sets the size of the window.
- **Label** → displays text.
- **Entry** → takes user input.
- **Button** → creates a clickable button.
- **command=on_click** → calls the `on_click()` function when the button is clicked.
- **label.config()** → changes the text of the label.
- **pack()** → arranges widgets in the window.
- **mainloop()** → starts and runs the GUI application.

Output:

Q. What is a window? How is a window created using Tkinter in Python?

Window

A window is the main screen of a GUI (Graphical User Interface) application. It acts as a container that holds widgets such as labels, buttons, entry fields, menus, and listboxes.

Creation of Window

A window is created using the **Tk() function** of the Tkinter library. First, the Tkinter library is imported. Then, the Tk() function is used to create the main application window.

Short Question

What is window and how is a window created using Tkinter?

Example:

```
import tkinter as tk
window = tk.Tk()
window.mainloop()
```

www.ilmkidunya.com

Q. What is the use of pack() and mainloop() in Tkinter.

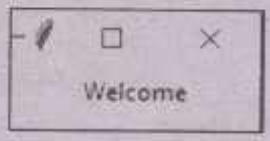
pack()

The pack() method is a geometry manager in Tkinter. It is used to place and display a widget inside a window. Without pack(), the widget is not displayed.

Short Question
Describe the use of pack() method in Tkitner in Pyhton?

Example:

```
import tkinter as tk
window = tk.Tk()
label = tk.Label(window, text="Welcome")
label.pack()
window.mainloop()
```



mainloop()

The mainloop() function starts and runs the GUI application. It keeps the window open and waits for user actions such as button clicks, keyboard input, or mouse clicks. Without mainloop(), the window opens and closes immediately.

Short Question
Why mainloop() function is used in GUI development?

Example:

```
import tkinter as tk
window = tk.Tk()
window.mainloop()
```

Ex.Q.3 What are the key components and widgets in Tkinter?

Tkinter Components and Widgets

Tkinter widgets are the visual components used to create graphical user interfaces (GUIs). Tkinter provides many widgets to build interactive GUI applications. A program can contain different visual elements such as windows, frames, buttons, and text fields.

Short Question
What are Tkinter widgets?

These widgets allow users to enter data, select options, and perform different actions. Each widget has a specific purpose. By combining different widgets, a complete and user-friendly interface can be created. Tkinter also provides layout management to arrange widgets neatly on the screen.

Common widgets used in Tkinter

Common Widgets: Labels, Buttons, Entry Fields, Menus, and Listboxes

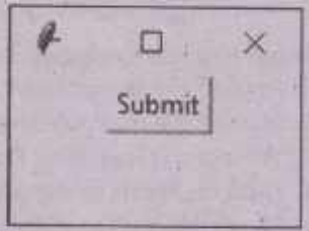
Widgets are the basic elements used to create a graphical user interface (GUI). Tkinter provides different widgets to design interactive applications. Each widget has a specific purpose and helps users interact with the program.

1. **Label**
   A Label is a widget used to display text or messages on the screen.
   Example: A label displaying **"Welcome to School"**.
   Code:

```
import tkinter as tk
window = tk.Tk()
label = tk.Label(window, text="Welcome to School")
label.pack()
window.mainloop()
```

2. **Button**
   A Button is a widget that allows the user to perform an action when it is clicked.
   Example: A **"Submit"** button.
   Code:

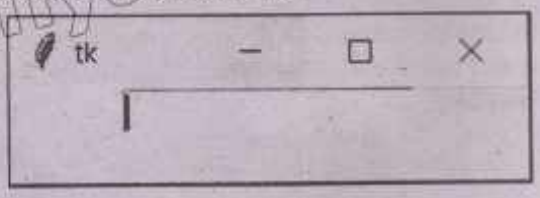
```
import tkinter as tk
window = tk.Tk()
button = tk.Button(window, text="Submit")
button.pack()
window.mainloop()
```



www.ilmkidunya.com

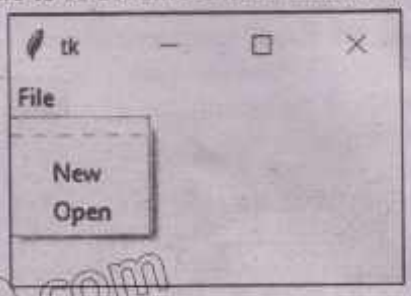
3. **Entry Field**
An Entry field is a widget used to enter text data such as a name, password, or other information.
Example: A text box where the user enters their **name**.
Code:

```
import tkinter as tk
window = tk.Tk()
entry = tk.Entry(window)
entry.pack()
window.mainloop()
```



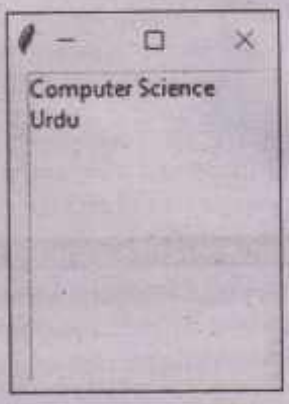
4. **Menu**
A Menu provides a list of options displayed at the top of a window. It allows users to select different commands or options.
Example: A **File** menu with **New, Open, Save,** and **Exit** options.
Code:

```
import tkinter as tk
window = tk.Tk()
menu_bar = tk.Menu(window)
file_menu = tk.Menu(menu_bar)
file_menu.add_command(label="New")
file_menu.add_command(label="Open")
menu_bar.add_cascade(label="File", menu=file_menu)
window.config(menu=menu_bar)
window.mainloop()
```



5. **Listbox**
A Listbox displays a list of items and allows users to select one or more items from the list.
Example: A list of **subjects** from which the user selects one.
Code:

```
import tkinter as tk
window = tk.Tk()
listbox = tk.Listbox(window)
listbox.pack()
listbox.insert(0, "Computer Science")
listbox.insert(1, "Urdu")
window.mainloop()
```



Q. How is a window is created and frames are added using tkinter in Python?

Creating Windows and Adding Frames

Window
A main window is the primary window of a Tkinter application that holds all other widgets. Every Tkinter program starts by creating a main window. This window acts as the base of the application. It can be resized, given a title, and used to display different widgets.

Frames
Frames are sections inside the main window that help organize the interface. They divide the window into smaller parts so that related widgets can be grouped together. This makes the interface clear, organized, and easy to manage.

Steps to create windows and frames using Tkinter in Python

1. Import the tkinter module.
2. Create the main window using Tk().
3. Create a Frame using Frame(window).
4. Add the frame to the window using pack(), grid(), or place().
5. Add widgets (Label, Button, Entry, etc.) inside the frame.
6. Run the application using mainloop().

Short Question
Write steps to create window and frames using Tkinter.

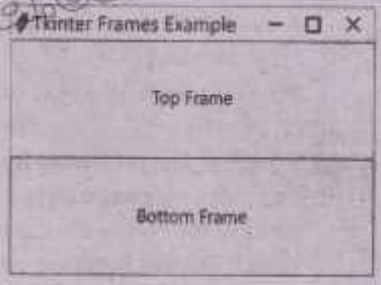
www.ilmkidunya.com

Example:

```
import tkinter as tk
window = tk.Tk()
window.title("Frame Activity")
window.geometry("400x300")
# Create the top frame with a light blue background
top_frame = tk.Frame(window, bg="lightblue", height=100)
# Display the top frame
top_frame.pack(fill="x")

# Create the bottom frame with a light green background
bottom_frame = tk.Frame(window, bg="lightgreen")
# Display the bottom frame and allow it to expand
bottom_frame.pack(fill="both", expand=True)
# Create a label inside the top frame
label = tk.Label(top_frame, text="Student Information", bg="lightblue")
label.pack(pady=10)
# Create an entry field inside the bottom frame
entry = tk.Entry(bottom_frame)
entry.pack(pady=10)
# Create a button inside the bottom frame
button = tk.Button(bottom_frame, text="Submit")
button.pack(pady=10)
window.mainloop()
```

Output:



Explanation:

- **Frame** → creates a container to organize widgets.
- **bg** → sets the background color of the frame.
- **height** → sets the height of the frame.
- **pack(fill="x")** → makes the frame fill the window horizontally.
- **pack(fill="both", expand=True)** → makes the frame fill the available space.
- **Label** → displays text inside the frame.
- **pack()** → places the widget inside the frame.
- **mainloop()** → starts and runs the GUI application.

CLASS ACTIVITY

Create a main window and divide it into two frames. Add different widgets in each frame to organize layout.

Sol:

```
import tkinter as tk
window = tk.Tk()
window.title("Frame Activity")
window.geometry("400x300")
top_frame = tk.Frame(window, bg="lightblue", height=100)
top_frame.pack(fill="x")
bottom_frame = tk.Frame(window, bg="lightgreen")
bottom_frame.pack(fill="both", expand=True)
label = tk.Label(top_frame, text="Student Information", bg="lightblue")
label.pack(pady=10)
entry = tk.Entry(bottom_frame)
entry.pack(pady=10)
button = tk.Button(bottom_frame, text="Submit")
button.pack(pady=10)
window.mainloop()
```

www.ilmkidunya.com

Additional Multiple Choice Questions

1. **A visual interface that allows users to interact with software using windows, buttons, and menus is called:**
   (a) Command Line (b) Graphical User Interface (c) Compiler (d) Database
2. **Which Python library is used to create Graphical User Interface (GUI) applications?**
   (a) NumPy (b) Pandas (c) Tkinter (d) Matplotlib
3. **Which of the following is an advantage of a GUI?**
   (a) Uses only commands (b) Requires programming knowledge
   (c) Easy for non-technical users (d) Works without windows
4. **Which Tkinter function creates the main application window?**
   (a) mainloop() (b) Label() (c) Button() (d) Tk()
5. **Which Tkinter method starts and keeps the GUI application running?**
   (a) geometry() (b) title() (c) mainloop() (d) config()
6. **A widget used to display text or messages on the screen is called:**
   (a) Button (b) Entry (c) Label (d) Listbox
7. **Which widget is used to enter text such as a name or password?**
   (a) Frame (b) Entry Field (c) Menu (d) Label
8. **A programmer wants to divide a window into different sections to organize widgets. Which Tkinter component should be used?**
   (a) Frame (b) Button (c) Entry (d) Listbox
9. **Which statement about Tkinter is correct?**
   (a) It is a database library. (b) It comes pre-installed with Python.
   (c) It is used only for web development. (d) It creates command-line programs only.
10. **A widget that displays a list of items and allows users to select one or more items is called:**
    (a) Label (b) Menu (c) Button (d) Listbox

Answers:

1	b	2	c	3	c	4	d	5	c	6	c	7	b	8	a	9	b	10	d

Ex.0.4 What is layout management in Tkinter, and how are elements organized using pack(), grid(), and place()?

Layout Management

Short Question
What is layout management in Tkinter?

Layout management is used to arrange widgets inside a window in a proper way. It decides the position and size of each widget on the screen. Good layout management makes the interface clear and easy to use. It helps in placing widgets properly inside a window.

Tkinter provides different layout management methods, and each method has its own way of arranging widgets. These methods help in creating a well-organized and user-friendly interface.

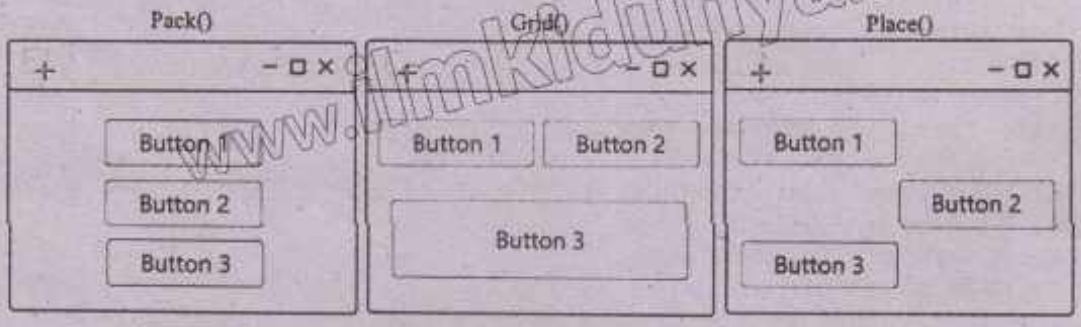


Figure 4.2 Diagram showing Tkinter Layout Managers: pack(), grid(), place()

www.ilmkidunya.com

Organizing Elements using pack(), grid(), and place()

1. pack() Method

The pack() method places widgets in a simple vertical or horizontal order. It is easy to use and is suitable for small applications. It places widgets automatically by arranging them in a specific direction, such as **top, bottom, left, or right.**

Example1:

```
label = tk.Label(window, text="Hello")
label.pack()
```

Example2:

```
button = tk.Button(window, text="Click")
button.pack(side="left")
```

Example3:

```
label = tk.Label(window, text="Hello")
label.pack(side="top")
```

2. grid() Method

The grid() method arranges widgets in rows and columns like a table. It is useful for creating forms and structured layouts.

Example:

```
label = tk.Label(window, text="Name")
label.grid(row=0, column=0)
```

Short Question

State purpose of the grid() method in Tkinter?

3. place() Method

The place() method positions widgets at exact locations on the window. It provides more control over the position of widgets but requires careful adjustment.

Example:

```
label = tk.Label(window, text="Hello")
label.place(x=50, y=30)
```

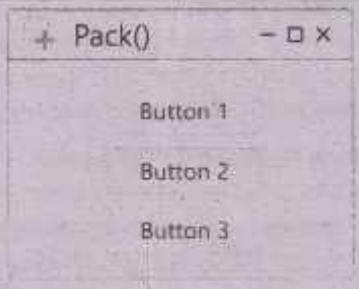
Short Question

What is the place() method in Tkinter?

Example pack():

```
import tkinter as tk
window = tk.Tk()
window.title("pack()")
window.geometry("300x220")
#Create buttons
button1 = tk.Button(window, text="Button 1", width=12)
button2 = tk.Button(window, text="Button 2", width=12)
button3 = tk.Button(window, text="Button 3", width=12)
#Arrange buttons using pack()
button1.pack(pady=10)
button2.pack(pady=10)
button3.pack(pady=10)
window.mainloop()
```

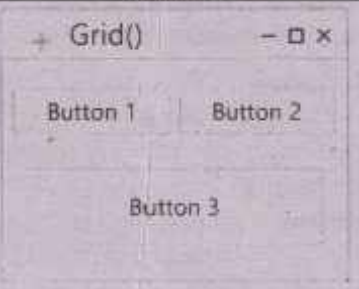
Output:



Example grid():

```
import tkinter as tk
window = tk.Tk()
window.title("grid()")
window.geometry("300x220")
#Create buttons
button1 = tk.Button(window, text="Button 1", width=10)
button2 = tk.Button(window, text="Button 2", width=10)
button3 = tk.Button(window, text="Button 3", width=24)
#Arrange buttons using grid()
button1.grid(row=0, column=0, padx=10, pady=20)
```

Output:



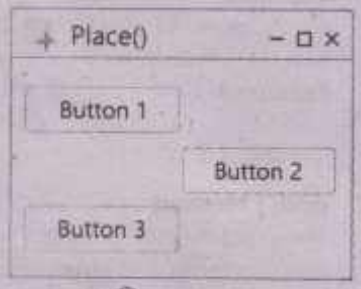
www.ilmkidunya.com

```
button2.grid(row=0, column=1, padx=10, pady=20)
#Button 3 spans two columns.
button3.grid(row=1, column=0, columnspan=2,
padx=10, pady=10)
#Keep content centred
window.grid_columnconfigure(0, weight=1)
window.grid_columnconfigure(1, weight=1)
window.mainloop()
```

Example place():

```
import tkinter as tk
window = tk.Tk()
window.title("place()")
window.geometry("300x220")
#Create buttons
button1 = tk.Button(window, text="Button 1", width=10)
button2 = tk.Button(window, text="Button 2", width=10)
button3 = tk.Button(window, text="Button 3", width=10)
#Arrange buttons using place()
button1.place(x=30, y=40)
button2.place(x=170, y=85)
button3.place(x=30, y=140)
window.mainloop()
```

Output:



TIDBIT

- Proper layout management helps make applications look neat and work well on different screen sizes.

Ex.Q.5 How can you design clean and responsive user interfaces in Tkinter?

Clean Interface

A clean interface has well-arranged widgets with proper spacing between different elements. Text, buttons, labels, and other widgets should be clear and easy to read. Similar widgets should be grouped together to make the interface organized. A simple and neat layout makes the application easy to understand and use.

Using Geometry Managers

Tkinter provides geometry managers such as **pack()**, **grid()**, and **place()** to arrange widgets.

- **pack()** manager arranges widgets automatically.
- **grid()** places widgets in rows and columns.
- **place()** positions widgets at fixed locations.

Using the appropriate geometry manager helps create a well-organized interface.

Responsive Interface

A responsive interface adjusts its layout when the window size changes. Tkinter supports this by using options such as **fill**, **expand**, and **weight**. These options allow widgets to resize and use the available space properly. This helps the application work well on different screen sizes.

Fill Option

The fill option allows a widget to expand and fill the available space in a specified direction. It can fill horizontally (fill="x"), vertically (fill="y"), or in both directions (fill="both").

Example:

```
label = tk.Label(window, text="Hello")
label.pack(fill="x")
```

Short Question

Describe purpose of fill option in responsive interface in Tkinter?

www.ilmkidunya.com

Expand Option

The expand option allows a widget to use any extra space available in the window when its size increases. It is commonly set as expand=True.

Example:
```
button = tk.Button(window, text="Click")
button.pack(expand=True)
```

Short Question

Describe purpose of expand option in responsive interface in Tkinter?

Weight Option

The weight option is used with the grid() layout manager to control how extra space is shared among rows and columns. A row or column with a higher weight receives more of the available space when the window is resized.

Example:
```
window.grid_columnconfigure(0, weight=1)
```

Short Question

Describe purpose of weight option in responsive interface in Tkinter?

Benefits of Clean and Responsive User Interfaces

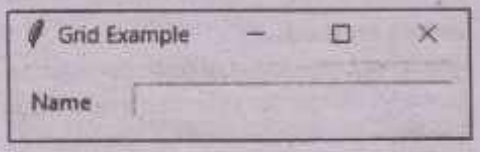
- Proper alignment and spacing improve the appearance of an application.
- A clean and responsive interface makes the program attractive, user-friendly, and easy to use.

Example 1: Using pack(), fill, and expand

```
import tkinter as tk
window = tk.Tk()
window.title("Pack Example")
window.geometry("300x200")
label = tk.Label(window, text="Tkinter",
bg="lightblue")
label.pack(fill="x", pady=10)
button = tk.Button(window, text="Click Me")
button.pack(fill="both", expand=True, padx=10, pady=10)
window.mainloop()
```

Example 2: Using grid() and weight

```
import tkinter as tk
window = tk.Tk()
window.title("Grid Example")
window.geometry("300x150")
window.columnconfigure(1, weight=1)
tk.Label(window, text="Name").grid(row=0, column=0, padx=10, pady=10)
tk.Entry(window).grid(row=0, column=1, padx=10, pady=10, sticky="ew")
window.mainloop()
```



Ex.Q.6 Explain the concept of event-driven programming in Tkinter.

Event-Driven Programming

Event-driven programming is a programming approach based on events. In this method, the program remains idle until an event occurs. Common events include button clicks, key presses, mouse clicks, and mouse movements. Each event is connected to a specific function called an event handler. The function runs only when the related event happens. This approach saves system resources and is widely used in GUI applications.

Event Handling

Event handling allows a program to respond to user actions. An event is an action performed by the user, such as clicking a button or typing text. When an event occurs, Tkinter automatically calls the connected function. This makes the application interactive and useful.

Handling User Input

Tkinter accepts user input through widgets such as **Entry** and **Text**. Buttons are used to trigger actions. These widgets are connected to functions using events. When the user interacts with a widget, the related function is executed. This allows the program to process user input and perform the required task correctly.

www.ilmkidunya.com

Role of mainloop()

The mainloop() function keeps the GUI application running and waits for user events. It continuously checks for actions such as button clicks or key presses. When an event occurs, it calls the appropriate event handler.

Benefits of Event-driven programming

- Event-driven programming makes GUI applications interactive, efficient, and easy to use.
- It allows the program to respond only when needed, improving performance and providing a better user experience.

Ex.Q.7 How do you handle user input and connect widgets to functions in Tkinter?

Handling User Input

Tkinter accepts user input through widgets such as **Entry**, **Text**, and **Button**. The **Entry** widget is commonly used to enter text such as a username or password. The program reads the entered data and processes it when required.

Connecting Widgets to Functions

Widgets are connected to functions using the **command option** or the **bind() method**. When the user interacts with a widget, such as clicking a button, the connected function is executed automatically. This allows the program to process user input and perform the required task. Proper connections between widgets and functions ensure correct program behavior and improve the user experience.

Benefits

- Connecting widgets to functions makes GUI applications interactive and user-friendly.
- It allows the program to respond to user actions and perform tasks correctly.

Command Option

The command option is used with widgets such as Button and Menu. It connects a widget to a function. When the user clicks the widget, the specified function is executed.

Example:

```
button= tk.Button(window, text="Click", command=show_message)
```

Short Question

What is the purpose of the command option in event-driven programming?

bind() Method

The bind() method is used to connect an event, such as a mouse click or key press, to a function. When the specified event occurs, the related function is executed.

Example: `entry.bind("<Key>", show_text)`

Short Question

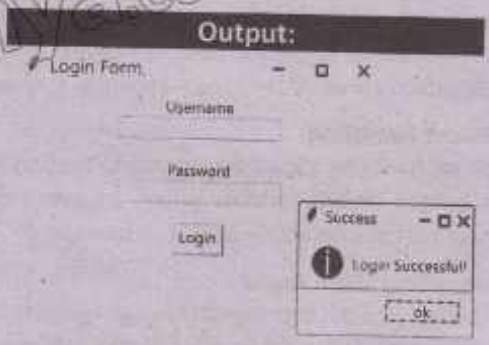
What is the purpose of the bind() method in event-driven programming?

Example: Simple Login Form

A simple login form allows users to enter a username and password. Entry widgets are used to take input from the user, and a Button is used to submit the information. When the button is clicked, the check_login() function is called. The function checks whether the username is "admin" and the password is "1234". If both are correct, the program displays a "Login Successful!" message. Otherwise, it displays an "Invalid Username or Password" error message.

```
import tkinter as tk
from tkinter import messagebox
window = tk.Tk()
window.title("Login Form")
window.geometry("300x200")
#Function to check login
def check_login():
    username = entry_user.get()
    password = entry_pass.get()
      # simple check
    if username == "admin" and password == "1234":
        messagebox.showinfo("Success", "Login Successful!")
```

Output:



www.ilmkidunya.com

```
    else:
        messagebox.showerror("Error", "Invalid Username or Password")
# Username label and entry
tk.Label(window, text="Username").pack()
entry_user = tk.Entry(window)
entry_user.pack()
#Password label and entry
tk.Label(window, text="Password").pack()
entry_pass = tk.Entry(window, show="*")
entry_pass.pack()
# Login button
tk.Button(window, text="Login", command=check_login).pack(pady=10)
window.mainloop()
```

- **Entry** → takes username and password
- **Button** → submits data
- **check_login()** → checks input
- **messagebox** → shows success or error
- Works on event-driven programming

CLASS ACTIVITY

Explain event-driven concepts and have students add a button that triggers a label update on click.

Sol: **Event-Driven Concept**

Event-driven programming is a programming style in which a program waits for a user action, called an **event**, such as clicking a button or pressing a key. When an event occurs, a specific function is executed to perform the required task.

Program

```
import tkinter as tk
def update_label():
    label.config(text="Button Clicked!")
window = tk.Tk()
window.title("Event Example")
window.geometry("300x150")
label = tk.Label(window, text="Click the Button")
label.pack(pady=10)
button = tk.Button(window, text="Click Me", command=update_label)
button.pack(pady=10)
window.mainloop()
```

www.ilmkidunya.com

Additional Multiple Choice Questions

1. **Layout management used to arrange widgets inside a window is called:**
   (a) Event handling (b) Layout management (c) Data processing (d) Widget control
2. **Which layout manager arranges widgets in rows and columns?**
   (a) place() (b) grid() (c) pack() (d) bind()
3. **Which layout manager is most suitable when exact widget positions are required?**
   (a) grid() (b) pack() (c) place() (d) fill()
4. **Which Tkinter option helps widgets to resize when the window size changes?**
   (a) geometry (b) command (c) title (d) expand
5. **A programming approach in which the program waits for user actions is called:**
   (a) Sequential programming (b) Event-driven programming
   (c) Modular programming (d) Object-oriented programming
6. **Which function keeps a Tkinter application running and waiting for user events?**
   (a) config() (b) bind() (c) mainloop() (d) geometry()
7. **Which method is commonly used to connect a Button widget to a function?**
   (a) place() (b) command (c) title() (d) pack()
8. **A user wants to create a registration form with labels and text fields arranged neatly in rows and columns. Which layout manager should be used?**
   (a) place() (b) pack() (c) grid() (d) bind()
9. **Which widget is mainly used to receive text input from the user?**
   (a) Label (b) Entry (c) Menu (d) Listbox
10. **Which statement best describes a responsive interface?**
    (a) Uses fixed positions only (b) Adjusts to window size
    (c) Displays one widget only (d) Prevents user input

Answers:

1	B	2	B	3	C	4	D	5	B	6	C	7	B	8	C	9	B	10	B

4.2 Working with Databases in Python

Q. What is a database? Also explain the concepts of database

Database

A database is an organized collection of data stored in a structured format. A database allows data to be accessed, stored, and managed easily. It reduces data duplication and improves reliability. Databases provide secure storage of information. They are commonly used in different software applications.

> ***Short Question***
> ***What is database and why databases are used?***

Python programs can connect to databases to perform different operations such as inserting, retrieving, updating, and deleting data. Databases help applications manage large amounts of information efficiently. Using databases improves data accuracy and consistency. Python supports database operations through built-in and external libraries.

i. **Entity**
An entity is a real-world object, person, place, event, or concept about which data is stored in a database.
Example:
`Student, Teacher, Course, Book, Employee`

ii. **Attribute**
An attribute is a property or characteristic of an entity. It describes specific information about an entity.
Example:
For the entity Student, attributes can be: `RollNo, Name, Class, Age`

www.ilmkidunya.com

iii. Relationship

A relationship shows how two or more entities are connected with each other.

Example:

A Student enrols in a Course. "enrols in" is the relationship between Student and Course.

iv. Identifier

An identifier is an attribute that uniquely identifies each record or object of an entity.

Example:

For the entity Student, Roll Number can be used as an identifier because every student has a unique roll number.

? DO YOU KNOW?

➢ Databases allow programs to store large amounts of information safely and retrieve it quickly when needed.

Q. What is SQLite3 and MySQL

SQLite3

Short Question
What do you know about SQLite3?

- SQLite3 is a **built-in Python library** used to create and manage small databases.
- It stores all database data in a single file on the computer.
- It does not require a separate database server.
- It is easy to use and is suitable for small applications, testing, and learning.
- It is included with Python, so no separate installation is required.

MySQL

Short Question
What is MySQL?

- MySQL is a **Database Management System** (DBMS) used to create and manage databases.
- It is suitable for large and complex database applications.
- It stores and manages data for multiple users.
- It requires a database server to store and access data.
- It is widely used in websites, business applications, and large information systems.

Ex.Q.8: What is a relational database? Discuss the process of connecting Python to a database using SQLite or MySQL.

Relational Database

Short Question
What is relational database?

A relational database is a database that **stores data in tables** consisting of rows and columns. Each table stores related data in a structured way. These tables are connected with each other through relationships.
Relationships between different tables are maintained using keys.

Components of a Relational Database

i. Table

A table is a collection of related data organized in rows and columns. Rows represent records, while columns represent fields.

Example: A **Student** table stores student information.

ii. Column (Field)

A column represents a property or characteristic of an entity.

Example:

Name and Age are columns in a Student table.

www.ilmkidunya.com

iii. Row (Record)

A row represents a single record in a table. Each row contains complete information about one item.

Example: One row contains the data of one student.

iv. Primary Key

A primary key is a unique identifier for each record in a table.

It ensures that every record can be identified separately.

Example: Student_ID

v. Foreign Key

A foreign key is a field that connects one table with another table. It refers to the primary key of another table.

Example: Student_ID in the **Marks** table refers to Student_ID in the **Student** table.

Figure 4.3: Tables, Columns, Records, Fields, Primary Key, and Foreign Key Relationship

Connecting Python to a Database

Connecting Python to a database allows programs to work with stored data. A database connection is required before performing any operation on data. Python provides libraries that help create a connection with databases easily. After establishing a connection, data can be added, retrieved, or modified. Proper connection handling ensures smooth communication between Python programs and databases.

Using sqlite3 or MySQL to Establish a Database Connection

A connection is created by providing required database details such as database name and credentials. A successful connection allows Python programs to access and manage data stored in the database.

```
# Import sqlite3 module
import sqlite3
#Connect to the database file named "school.db"
# If the file does not exist, SQLite will create it automatically
connection = sqlite3.connect("school.db")
# Create a cursor object
#Cursor is used to execute SQL queries
cursor = connection.cursor()
# Create "students" table if it does not exist
cursor.execute("""
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY,
    name TEXT,
    age INTEGER,
    grade TEXT
)
```

```
""")
# (Optional) Insert sample data
cursor.execute("INSERT INTO students(name,age,grade)VALUES ('Ali', 14, '8th')")
cursor.execute("INSERT INTO students(name,age,grade)VALUES ('Sara', 15, '9th')")
connection.commit()
#This query selects all columns (*) from the table named "students"
cursor.execute("SELECT * FROM students")
# Fetch all records returned by the query
# fetchall() returns a list of rows/records
records = cursor.fetchall()
# Display all records
#Loop through each row in the records list
for row in records:
    print(row)#Print each row of the table
#Close the database connection
connection.close()
```

Output:

```
(1, 'Ali', 14, '8th')
(2, 'Sara', 15, '9th')
```

- **Connect to Database:** Establishes a connection to the school.db SQLite database.
- **Create Cursor**: Creates a cursor object to execute SQL commands.
- **Execute Query:** Executes a SQL query to fetch all records from the students table.
- **Fetch and Display Data:** Fetches all records and prints each row in the result.
- **Close Connection:** Closes the database connection after the operation.

Q. What is SQL? How are SQL commands are executed from Python.

SQL

SQL stands for **Structured Query Language**. It is used to communicate with relational databases. It allows you to store, retrieve, modify, and manage data.

An SQL command is any instruction written in SQL. SQL commands are used to create, read, update, and delete data.

Executing SQL Commands From Python

SQL commands are used to interact with databases from Python programs. Python sends SQL commands to the database using a **database cursor**. These commands can be used to create tables and insert data into the database.

Short Question
How SQL commands are executed from Python?

Data can also be retrieved using SELECT queries. Executing SQL commands allows Python programs to store, access, and manage database data effectively.

Q. What are CRUD operations? Explain Create, Read, Update, and Delete operations.

CRUD Operations

CRUD stands for **Create, Read, Update, and Delete.** These are the basic operations used to manage data stored in a database.

CRUD operations are used in almost every database-based application. They help maintain accurate and updated information. Python uses SQL commands to perform these operations on databases.

www.ilmkidunya.com

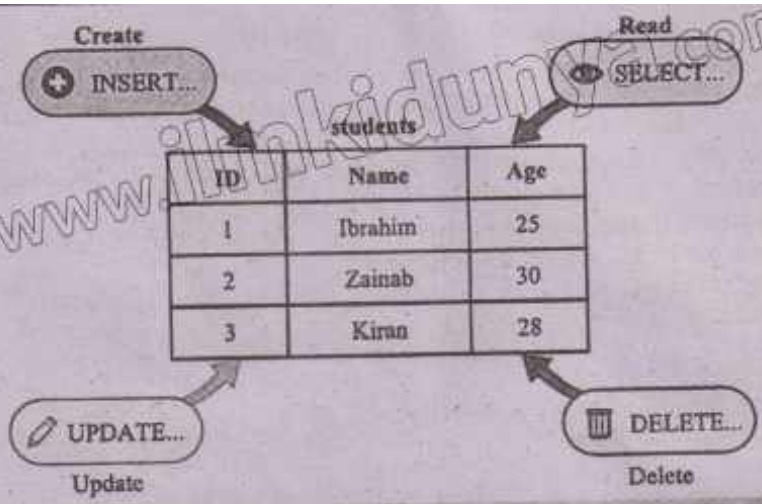


Figure 4.4: Diagram showing Create, Read, Update, and Delete operations on a table

Table 4.1: CRUD operations

Command	Command Example	Description
INSERT	`INSERT INTO students (ID, Name, Age) VALUES (1, 'John', 20);`	Adds a new record to a table.
READ	`SELECT * FROM students;`	Retrieves data from a table.
UPDATE	`UPDATE students SET Age = 21 WHERE ID = 1;`	Modifies an existing record in a table.
DELETE	`DELETE FROM students WHERE ID = 1;`	Removes a record from a table.

Create: Adding New Records

The Create operation is used to add new data to a database table.
It stores new information such as names, values, or records. SQL INSERT commands are used for creating new records. Python sends these commands to the database. After execution, the new records become part of the stored data.

Read: Retrieving and Displaying Data

The Read operation is used to retrieve data from a database.
It allows users to view stored records when needed. SQL SELECT commands are commonly used for reading data. Python retrieves the results and processes them. This operation does not make any changes to the stored data.

Update: Modifying Existing Records

The Update operation is used to change existing data in a database.
It is used when stored information needs correction or modification. SQL UPDATE commands are used to modify selected records. Python executes these commands according to given conditions. Only the specified data is changed.

Delete: Removing Data Safely

The Delete operation is used to remove unwanted data from a database.
It helps keep the database clean and accurate. SQL DELETE commands are used with conditions to remove selected records. Python ensures that only required records are deleted. Careful use of this operation prevents the loss of important data.

www.ilmkidunya.com

Example:

```
import sqlite3
connection = sqlite3.connect("school.db")
cursor = connection.cursor()
cursor.execute("DELETE FROM students WHERE id = 1")
connection.commit()
print("Record deleted successfully")
connection.close()
```

Output: Record deleted successfully

- **DELETE FROM students WHERE ID = 1** → removes a specific record
- **connection.commit()**→ saves the changes in database
- **Without commit(),** deletion will not be permanent

CLASS ACTIVITY

Update a student's marks and display updated data.

Sol:

```
import sqlite3
# Connect to database
connection = sqlite3.connect("school.db")
cursor = connection.cursor()
# Create table
cursor.execute("""
CREATE TABLE IF NOT EXISTS students2 (
    roll_no INTEGER PRIMARY KEY,
    name TEXT,
    marks INTEGER
)
""")
# Insert sample data
cursor.execute("INSERT OR IGNORE INTO students2 VALUES (1, 'Ali', 75)")
cursor.execute("INSERT OR IGNORE INTO students2 VALUES (2, 'Sara', 82)")
connection.commit()

# Update student's marks
cursor.execute("UPDATE students2 SET marks = 90 WHERE roll_no = 1")
connection.commit()

# Display updated data
cursor.execute("SELECT * FROM students2")
print("Updated Student Records:")

for row in cursor.fetchall():
    print(row)
# Close connection
connection.close()
```

Output

```
Updated Student Records:
(1, 'Ali', 90)
(2, 'Sara', 82)
```

www.ilmkidunya.com

Additional Multiple Choice Questions

1. **An organized collection of data stored in a structured format is called:**
   (a) Spreadsheet (b) Database (c) Program (d) Network
2. **A real-world object about which data is stored in a database is called:**
   (a) Attribute (b) Entity (c) Relationship (d) Record
3. **A property or characteristic of an entity is called:**
   (a) Attribute (b) Column (c) Identifier (d) Table
4. **An attribute that uniquely identifies each record is called:**
   (a) Foreign Key (b) Column (c) Primary Key (d) Identifier
5. **Which database key connects one table with another?**
   (a) Candidate Key (b) Primary Key (c) Foreign Key (d) Composite Key
6. **A database that stores data in tables of rows and columns is called:**
   (a) Flat Database (b) Relational Database (c) Distributed Database (d) Object Database
7. **SQL stands for:**
   (a) Structured Query Language (b) System Query Logic (c) Standard Question Language (d) Simple Query List
8. **Which Python library is built-in for creating and managing SQLite databases?**
   (a) pandas (b) tkinter (c) sqlite3 (d) matplotlib
9. **A Python program needs to retrieve all student records from a database. Which SQL command should be used?**
   (a) INSERT (b) UPDATE (c) DELETE (d) SELECT
10. **Which CRUD operation is used to change an existing record in a database?**
    (a) Create (b) Read (c) Update (d) Delete

Answers:

1	B	2	B	3	A	4	D	5	C	6	B	7	A	8	C	9	D	10	C

TEXTBOOK EXERCISE (SOLVED)

MULTIPLE CHOICE QUESTIONS

1. **The main purpose of a GUI in Python programming is to:**
   (a) Create a user-friendly interface for interacting with software (b) Manage files in Python
   (c) Handle network operations (d) Store and retrieve data from databases
2. **Python's built-in GUI toolkit is:**
   (a) wxPython (b) Tkinter (c) PyQt (d) Kivy
3. **The Tkinter widget used to display text is:**
   (a) Button (b) Label (c) Entry (d) Listbox
4. **The pack() method in Tkinter is used to:**
   (a) Align widgets vertically or horizontally (b) Organize widgets in rows and columns
   (c) Set widget sizes manually (d) Create pop-up windows
5. **The methods used to organize widgets in Tkinter include:**
   (a) grid() (b) pack() (c) place() (d) All of the above
6. **Event-driven programming in Tkinter means:**
   (a) Writing code that runs without user input
   (b) Writing code that responds to user actions such as clicks or key presses
   (c) Writing code to manage database queries
   (d) Writing code to handle network requests

www.ilmkidunya.com

7. **The Tkinter widget used to get user input is:**
   (a) Label (b) Button (c) Entry (d) Listbox
8. **The Tkinter layout manager that allows precise positioning of widgets using coordinates is:**
   (a) pack() (b) grid() (c) place() (d) align()
9. **The Tkinter option that connects a button click to a function is:**
   (a) bind() (b) configure() (c) command= (no parentheses) (d) execute()
10. **CRUD operations in databases stand for:**
   (a) Create, Read, Update, Delete (b) Create, Run, Upload, Delete
   (c) Copy, Retrieve, Upload, Download (d) Create, Remove, Update, Manage

Answers:

1	A	2	B	3	B	4	A	5	D	6	B	7	C	8	C	9	C	10	A

SHORT QUESTIONS

1. **What is a GUI and why is it important in application development?**

Ans: A Graphical User Interface (GUI) is a visual interface that allows users to interact with software by using windows, buttons, labels, text boxes, menus, and forms.

- A GUI makes software easy to understand and use, especially for non-technical users.
- It allows users to interact with programs by clicking buttons, selecting menus, and filling forms instead of typing commands.

2. **What is Tkinter and why is it Python's built-in GUI toolkit?**

Ans: Tkinter is Python's standard GUI toolkit. Tkinter provides a collection of classes and functions to create GUI applications. Tkinter is easy to learn and lightweight. Tkinter is called Python's built-in GUI toolkit because it comes pre-installed with Python and does not require a separate installation.

3. **How do you create a window and add frames in Tkinter?**

Ans: A window is created using the Tk() function. Frames are created using the Frame() widget and are added to the window using geometry managers such as pack(), grid(), or place(). Frames help organize widgets into different sections of the window.

4. **Name two common widgets used in Tkinter.**

Ans: Two common widgets used in Tkinter are:

- **Label:** It displays text.
- **Button :** It performs an action when clicked.

5. **What is the purpose of layout management in Tkinter?**

Ans: Layout management is used to arrange widgets inside a window in a proper way. It decides the position and size of each widget on the screen. Good layout management makes the interface clear and easy to use. It helps in placing widgets properly inside a window.

6. **How does the pack() method organize elements in Tkinter?**

Ans: The pack() method places widgets in a simple vertical or horizontal order. It is easy to use and is suitable for small applications. It places widgets automatically by arranging them in a specific direction, such as **top, bottom, left, or right**.

7. **What is event-driven programming and how is it used in Tkinter?**

Ans: Event-driven programming is a programming approach based on events. In Tkinter, events such as button clicks, key presses, and mouse clicks are connected to functions. When an event occurs, the related function runs automatically to perform the required task.

www.ilmkidunya.com

8. **How do you handle user input in Tkinter?**

Ans: Tkinter accepts user input through widgets such as **Entry** and **Text**. The **Entry** widget is commonly used to enter text such as a username or password. The program reads the entered data and processes it when required.

9. **What is the CRUD operation in database management?**

Ans: CRUD stands for Create, Read, Update, and Delete. These are the four basic operations used to manage data in a database. They help maintain accurate and updated information. In Python, CRUD operations are performed using SQL commands.

10. **How do you connect Python to a database like SQLite?**

Ans: Python connects to an SQLite database using the sqlite3 module. A connection is created by providing the database file name with the connect() function. After the connection is established, Python can access and manage the data stored in the database.

For example: `connection = sqlite3.connect("school.db")`

LONG QUESTIONS

All Qs are Solved in chapter

1. Explain the concept of Graphical User Interface (GUI) and discuss its importance in application development.
2. Describe Tkinter as Python's built-in GUI toolkit.
3. What are the key components and widgets in Tkinter?
4. What is layout management in Tkinter, and how are elements organized using pack(), grid(), and place()?
5. How can you design clean and responsive user interfaces in Tkinter?
6. Explain the concept of event-driven programming in Tkinter.
7. How do you handle user input and connect widgets to functions in Tkinter?
8. Discuss the process of connecting Python to a database using SQLite or MySQL.

ADDITIONAL SHORT QUESTIONS

Tkinter

1. **Describe the features of Tkinter.**

Ans: Tkinter has many useful features for GUI development. It is easy to learn and lightweight. It supports event-driven programming, where actions are performed when users interact with the application. It is suitable for developing small to medium-sized GUI applications.

2. **What is the purpose of tk.Tk() in Tkinter?**

Ans: The tk.Tk() function is used to create the main application window in Tkinter. Every Tkinter program starts by creating a main window that holds other widgets.

3. **What is the purpose of mainloop() in Tkinter?**

Ans: The mainloop() function starts and runs the GUI application. It keeps the window open and waits for user actions such as button clicks, keyboard input, or mouse clicks. Without mainloop(), the window opens and closes immediately.

4. **Why is `import tkinter as tk` used?**

Ans: `import tkinter as tk`

This statement imports the Tkinter library. The name tk is an alias that makes it shorter and easier to write commands such as tk.Tk() and tk.Label().

www.ilmkidunya.com

Tkinter Components and Widgets

5. Difference between Window and Widget.

Ans:

Window	Widget
A window is the main screen of a GUI (Graphical User Interface) application.	Tkinter widgets are the visual components used to create graphical user interfaces (GUIs).
It acts as a container that holds widgets such as labels, buttons, entry fields, menus, and listboxes.	Widgets allow users to enter data, select options, and perform different actions.

6. List common widgets used in Tkinter.

Ans: Common widgets are Labels, Buttons, Entry Fields, Menus, and Listboxes.

7. Explain the importance of widgets in Tkinter.

Ans: Widgets are important components of Tkinter because they are used to create graphical user interfaces. Different widgets such as buttons, labels, and text fields help users interact with the application and perform different tasks.

8. Explain the functions of Label and Entry widgets in Tkinter.

Ans: A Label widget is used to display text or messages on the screen. An Entry widget is used to take input from users, such as names, passwords, or other information.

9. What is the function of Button widget in Tkinter?

Ans: A Button widget is used to create a clickable button in a GUI application. It allows the user to perform an action when the button is clicked.

10. What are frames in Tkinter? Explain their purpose.

Ans: Frames are sections inside the main window that help organize the interface. They divide the window into smaller parts so that related widgets can be grouped together. This makes the interface clear, organized, and easy to manage.

11. What is a Menu widget in Tkinter?

Ans: A Menu widget provides a list of options displayed at the top of a window. It allows users to select different commands or options from the available list.

12. What is purpose of a Listbox widget in Tkinter?

Ans: A Listbox widget displays a list of items on the screen. It allows users to select one or more items from the given list according to their requirements.

13. What is the place() method in Tkinter?

Ans: The place() method positions widgets at exact locations on the window. It provides more control over the position of widgets but requires careful adjustment.

14. How can you design a clean interface in Tkinter?

Ans: A clean interface can be designed by arranging widgets properly with suitable spacing between different elements. Text, buttons, labels, and other widgets should be clear and easy to read. Similar widgets should be grouped together to make the interface organized.

15. What is the role of geometry managers in Tkinter?

Ans: Geometry managers help arrange widgets inside a window. Tkinter provides pack(), grid(), and place() geometry managers. The pack() manager arranges widgets automatically, grid() places widgets in rows and columns, and place() positions widgets at fixed locations.

16. What is a responsive interface in Tkinter?

Ans: A responsive interface adjusts its layout when the window size changes. It allows widgets to resize and use the available space properly so that the application works well on different screen sizes.

17. Which options help create responsive interfaces in Tkinter?

Ans: Tkinter supports options such as fill, expand, and weight to create responsive interfaces. These options allow widgets to resize and adjust according to the available space.

www.ilmkidunya.com

Event-Driven Programming in Tkinter

18. What are common events in Tkinter?

Ans: Common events in Tkinter include button clicks, key presses, mouse clicks, and mouse movements. These events allow the program to respond to user actions.

19. What is event handling in Tkinter.

Ans: Event handling allows a program to respond to user actions. When an event occurs, Tkinter automatically calls the connected function. This makes the application interactive and useful.

20. What is the role of mainloop() in event-driven programming?

Ans: The mainloop() function keeps the GUI application running and waits for user events. It continuously checks for actions such as button clicks or key presses. When an event occurs, it calls the appropriate event handler.

21. What are the benefits of event-driven programming?

Ans: Event-driven programming makes GUI applications interactive, efficient, and easy to use.
It allows the program to respond only when needed, improving performance and providing a better user experience.

22. How are widgets connected to functions in Tkinter?

Ans: Widgets are connected to functions using the **command option** or the **bind() method**. When a user interacts with a widget, such as clicking a button, the connected function is executed automatically.

Working with Databases in Python

23. List components of relational databases

Ans: The Components of relational databases are tables, fields, records, primary key and foreign key.

24. What is an entity in a database? Give examples.

Ans: An entity is a real-world object, person, place, event, or concept about which data is stored in a database. Examples of entities are Student, Teacher, Course, Book, and Employee.

25. What is an identifier in a database? Give an example.

Ans: An identifier is an attribute that uniquely identifies each record or object of an entity. For example, Roll Number can be used as an identifier for a Student because every student has a unique roll number.

26. What is a primary key?

Ans: A primary key is a unique identifier for each record in a table. It ensures that every record can be identified separately and no two records have the same primary key value.

27. What is a foreign key?

Ans: A foreign key is a field that connects one table with another table. It refers to the primary key of another table and helps maintain relationships between tables.

28. What is the use of sqlite3 library in Python?

Ans: The sqlite3 library is a built-in Python library used for creating and managing small databases. It stores database information in a single file on the computer.

29. Differentiate between SQLite3 and MySQL

Ans:

SQLite3	MySQL
SQLite3 is a file-based database management system.	MySQL is a server-based database management system.
It does not require a separate database server.	It requires a separate database server.
It is suitable for small applications.	It is suitable for medium and large applications.

30. What is the Create operation in CRUD.

Ans: The Create operation is used to add new records to a database. The SQL INSERT command is commonly used to perform the Create operation.

31. Explain the Update operation in CRUD.

Ans: The Update operation is used to modify existing data in a database. SQL UPDATE commands are used to change selected records according to given conditions. Only the specified data is modified.

www.ilmkidunya.com

32. **Which Python library is used to connect with an SQLite database?**

Ans: The sqlite3 library.

33. **Which SQL command is used to retrieve all records from the students table in Python SQLite3?**

Ans: `cursor.execute("SELECT * FROM students")`

34. **Why should `connection.close()` be called after completing database operations?**

Ans: It closes the database connection and releases system resources properly.

35. **A program should execute a function only when the user clicks a button. Which programming approach is being used?**

Ans: Event-driven programming.

Write the Code Short Questions

1. **Write a Python code to create a window with the Title of "Tkinter"**

Ans:
```
import tkinter as tk
window = tk.Tk()
window.title("Tkinter")
window.mainloop()
```

2. **Write a Python code to create a window with a label showing "Welcome to Tkinter"**

Ans:
```
import tkinter as tk
window = tk.Tk()
label = tk.Label(window,text="Welcome to Tkinter")
label.pack()
window.mainloop()
```

3. **Write a Python code to create a window having a text box.**

Ans:
```
import tkinter as tk
window = tk.Tk()
entry = tk.Entry(window)
entry.pack()
window.mainloop()
```

4. **Write a Python code to create a button in a window.**

Ans:
```
import tkinter as tk
window = tk.Tk()
button = tk.Button(window)
button.pack()
window.mainloop()
```

5. **Write a Python code to create and connect to the *school.db* database in SQLite3.**

Ans:
```
import sqlite3
connection = sqlite3.connect("school.db")
connection.close()
```

6. **Write a SQL command to create a table named students in the *school.db* database in Python SQLite3.**

Ans:
```
cursor.execute("""
CREATE TABLE IF NOT EXISTS students (
roll_no INTEGER PRIMARY KEY,
name TEXT, marks INTEGER )""")
```

7. **Write a SQL command to insert a record (1, "Ali", 85) into the students table.**

Ans: `cursor.execute("INSERT INTO students VALUES (1, 'Ali', 85)")`

8. **Write a SQL command to update the 90 marks of the student whose roll number is 1.**

Ans: `cursor.execute("UPDATE students SET marks = 90 WHERE roll_no = 1")`

9. **Write a SQL command to delete all records from the students table in Python SQLite3.**

`cursor.execute("DELETE FROM students")`

www.ilmkidunya.com

CONCEPTUAL SHORT QUESTIONS

1. **You want to clear an Entry widget after the user clicks the Submit button. Which Tkinter method is used to remove the text from the Entry widget?**

Ans: The `entry.delete(0, tk.END)` method is used to remove all text from an Entry widget. It clears the text box so the user can enter new data.

2. **Write a single line of Tkinter code to create a main window named root.**

Ans:
```
root = tk.Tk()
```
This statement creates the main Tkinter window and stores it in the variable root.

3. **A programmer wants to display the message "Welcome" on a window. Which Tkinter widget should be used?**

Ans: The Label widget should be used. It is used to display text or messages on a Tkinter window.

4. **You are creating a student registration form with labels and text boxes arranged in rows and columns. Which layout manager is most suitable?**

Ans: The grid() layout manager is the most suitable. It arranges widgets neatly in rows and columns.

5. **You want a button to expand and fill the available space when the window is resized. Which pack() option should be used?**

Ans: The **expand=True** option should be used. It allows the button to expand when the window size increases.

6. **A Python program shows the error "no such table: students" while inserting data. What step did the programmer forget?**

Ans: The programmer forgot to create the students table before inserting the records. The table must exist before data can be added.

7. **After inserting records into an SQLite database, the data disappears when the program closes. What statement is missing?**

Ans: The connection.commit() statement is missing. It saves the changes permanently in the database.

8. **A programmer wants to change a student's marks in the database. Which SQL command should be used?**

Ans: The UPDATE command should be used. It modifies the existing data in a table.

9. **A programmer creates a Tkinter window with a button, but clicking the button does not produce any output. What is the most likely reason?**

Ans: The button is not linked to a function using the command option. Therefore, clicking the button does not execute any action.

10. **A programmer inserts new student records into an SQLite database. The program runs without errors, but the new records are not saved after the program closes. What statement is missing?**

Ans: The connection.commit() statement is missing. It saves the inserted records permanently in the database before the program closes.

11. **When a user clicks a "Save" button, your program crashes with an error saying the database table does not exist. Analyze the problem: what step did the programmer forget to do first?**

Ans: The programmer forgot to **create the database table** before trying to save the data. The table must exist before records can be inserted into it.

www.ilmkidunya.com

TEST YOUR SKILLS

Total Marks: 40 **Time: 1:30 Hour**

Q.1 MULTIPLE CHOICE QUESTIONS **8 x 1 = 8**

1. **Which Tkinter function creates the main application window?**
   (a) mainloop() (b) Label() (c) Button() (d) Tk()
2. **Which Tkinter method starts and keeps the GUI application running?**
   (a) geometry() (b) title() (c) mainloop() (d) config()
3. **The Tkinter widget used to display text is:**
   (a) Button (b) Label (c) Entry (d) Listbox
4. **The methods used to organize widgets in Tkinter include:**
   (a) grid() (b) pack() (c) place() (d) All of the above
5. **The Tkinter widget used to get user input is:**
   (a) Label (b) Button (c) Entry (d) Listbox
6. **Which statement best describes a responsive interface?**
   (a) Uses fixed positions only (b) Adjusts to window size
   (c) Displays one widget only (d) Prevents user input
7. **Which Python library is built-in for creating and managing SQLite databases?**
   (a) pandas (b) tkinter (c) sqlite3 (d) matplotlib
8. **CRUD operations in databases stand for:**
   (a) Create, Read, Update, Delete (b) Create, Run, Upload, Delete
   (c) Copy, Retrieve, Upload, Download (d) Create, Remove, Update, Manage

Q.2 SHORT QUESTIONS: **8 x 2 = 16**

Note: *Attempt any 8 short questions*

i. What is the purpose of mainloop() in Tkinter?
ii. What is the purpose of tk.Tk() in Tkinter?
iii. What is Tkinter and why is it Python's built-in GUI toolkit?
iv. How do you create a window and add frames in Tkinter?
v. How does the pack() method organize elements in Tkinter?
vi. How are widgets connected to functions in Tkinter?
vii. Differentiate between SQLite3 and MySQL.
viii. Write a Python code to create a button in a window.
ix. Write a SQL command to insert a record (1, "Ali", 85) into the students table
x. A programmer inserts new student records into an SQLite database. The program runs without errors, but the new records are not saved after the program closes. What statement is missing?

LONG QUESTIONS: **2 x 8 = 16**

Note: *Attempt any 2 long questions*

3. Explain the concept of Graphical User Interface (GUI) and discuss its importance in application development.
4. Explain the concept of event-driven programming in Tkinter.
5. Discuss the process of connecting Python to a database using SQLite or MySQL.

www.ilmkidunya.com