**Advance Python**

**Module 11: Writing GUIs in Python**

* **Introduction**

A **GUI** (Graphical User Interface) is a visual way for users to interact with a program. Instead of typing commands in a console, users click buttons, select menus, enter text in boxes, and see results visually.

**Why Use GUIs in Python?**

* **User-friendly** – Easier for non-technical users.
* **Visual feedback** – Makes interaction more intuitive.
* **Better experience** – Professional look and feel.
* **Cross-platform** – Works on Windows, macOS, and Linux.

**Example: Simple Tkinter GUI**

import tkinter as tk

# Create main window

root = tk.Tk()

root.title("My First GUI")

root.geometry("300x200")

# Add a label

label = tk.Label(root, text="Hello, Python GUI!", font=("Arial", 14))

label.pack(pady=20)

# Add a button

def say\_hello():

label.config(text="You clicked the button!")

button = tk.Button(root, text="Click Me", command=say\_hello)

button.pack(pady=10)

# Run the window

root.mainloop()

* **Components and Events**
* **GUI Components (Widgets)**

Widgets are the building blocks of a GUI—things you see and interact with.

|  |  |  |
| --- | --- | --- |
| **Component** | **Purpose** | **Example in Tkinter** |
| **Label** | **Displays text or images** | **tk.Label(root, text="Hello")** |
| **Button** | **Clickable element to trigger actions** | **tk.Button(root, text="Click")** |
| **Entry** | **Single-line text input field** | **tk.Entry(root)** |
| **Text** | **Multi-line text area** | **tk.Text(root)** |
| **Checkbutton** | **Checkbox to toggle options** | **tk.Checkbutton(root, text="Agree")** |
| **Radiobutton** | **Select one from multiple options** | **tk.Radiobutton(root, text="Option 1")** |
| **Listbox** | **List of selectable items** | **tk.Listbox(root)** |
| **Canvas** | **Draw shapes, images, or graphics** | **tk.Canvas(root, width=200, height=200)** |
| **Frame** | **Container to group widgets** | **tk.Frame(root)** |

* **Events**

An event is an action performed by the user or system that the program can respond to.

Common GUI Events

|  |  |
| --- | --- |
| **Event** | **Trigger** |
| **<Button-1>** | **Left mouse button click** |
| **<Button-2>** | **Middle mouse button click** |
| **<Button-3>** | **Right mouse button click** |
| **<Double-Button-1>** | **Double-click left mouse button** |
| **<Enter>** | **Mouse enters the widget area** |
| **<Leave>** | **Mouse leaves the widget area** |
| **<Key>** | **Any key press** |
| **<KeyPress-a>** | **Pressing a specific key (e.g.,** a) |
| **<Return>** | **Enter/Return key** |
| **<Motion>** | **Mouse movement** |

* **An Example GUI**

import tkinter as tk

# Create the main window

root = tk.Tk()

root.title("Simple GUI Example")

root.geometry("300x200")

# Function to greet user

def greet():

name = entry.get()

label\_result.config(text=f"Hello, {name}!")

# Label

label = tk.Label(root, text="Enter your name:")

label.pack(pady=5)

# Entry box

entry = tk.Entry(root, width=25)

entry.pack(pady=5)

# Button

button = tk.Button(root, text="Greet", command=greet)

button.pack(pady=5)

# Result label

label\_result = tk.Label(root, text="", fg="blue")

label\_result.pack(pady=5)

# Run the GUI loop

root.mainloop()

* **The root Component**

In Python GUI programming—especially when using **Tkinter**—the **root component** is the **main window** of your application.

It’s like the "foundation" of a house: everything else (buttons, labels, text boxes, etc.) is placed **inside** this root window.

**In Tkinter:**

* The root component is created with:

import tkinter as tk

root = tk.Tk() # This is the root window

tk.Tk() initializes:

The main application window

The connection to the Tkinter event loop

A container for all other widgets

Example:

import tkinter as tk

# Create root window

root = tk.Tk()

# Title and size

root.title("My First GUI")

root.geometry("400x300")

# Add a label inside root

label = tk.Label(root, text="Hello, Tkinter!", font=("Arial", 16))

label.pack(pady=20)

# Run the event loop

root.mainloop()

* **Adding a Button**

In Python’s **Tkinter**, adding a button is straightforward—you just create a Button widget and place it inside the **root component** (or any other frame).

Example: Adding a Button

import tkinter as tk

# Create root window

root = tk.Tk()

root.title("Button Example")

root.geometry("300x200")

# Function to run when button is clicked

def on\_click():

print("Button clicked!")

# Create a button

btn = tk.Button(root, text="Click Me", command=on\_click)

# Place the button inside the window

btn.pack(pady=20)

# Run the event loop

root.mainloop()

**Key Notes**

* **text** → Label shown on the button.
* **command** → Function to run when the button is clicked (no parentheses here).
* **pack()** (or grid() / place()) → Method to arrange the button in the window.
* You can style it:

btn = tk.Button(root, text="Click Me", bg="blue", fg="white", font=("Arial", 14))

Multiple Buttons Example:-

import tkinter as tk

# Create root window

root = tk.Tk()

root.title("Multiple Buttons Example")

root.geometry("400x300")

# Functions for buttons

def greet():

label.config(text="Hello, Amit! 👋")

def change\_color():

root.configure(bg="lightblue")

def exit\_app():

root.destroy()

# Label to show messages

label = tk.Label(root, text="Click a button", font=("Arial", 14))

label.pack(pady=20)

# Buttons

btn1 = tk.Button(root, text="Greet", command=greet, bg="green", fg="white", width=10)

btn1.pack(pady=5)

btn2 = tk.Button(root, text="Change Color", command=change\_color, bg="orange", fg="white", width=10)

btn2.pack(pady=5)

btn3 = tk.Button(root, text="Exit", command=exit\_app, bg="red", fg="white", width=10)

btn3.pack(pady=5)

# Run event loop

root.mainloop()

* **Entry Widgets**

In Python **Tkinter**, an **Entry widget** is a small text box that lets the user type in single-line input (like a username, password, or number).

Basic Example: Entry Widget

import tkinter as tk

# Create root window

root = tk.Tk()

root.title("Entry Widget Example")

root.geometry("400x200")

# Function to display entered text

def show\_text():

entered\_text = entry.get() # Get text from entry widget

label.config(text=f"You entered: {entered\_text}")

# Label

label = tk.Label(root, text="Enter something:", font=("Arial", 14))

label.pack(pady=10)

# Entry widget

entry = tk.Entry(root, font=("Arial", 14))

entry.pack(pady=5)

# Button to get text

btn = tk.Button(root, text="Submit", command=show\_text, bg="blue", fg="RED", font=("Arial", 12))

btn.pack(pady=10)

# Output label

label = tk.Label(root, text="", font=("Arial", 14))

label.pack(pady=10)

# Run event loop

root.mainloop()

**Key Notes**

* **Entry(root)** → Creates an input box.
* **get()** → Reads the text inside the box.
* **delete(start, end)** → Clears the box (example: entry.delete(0, tk.END)).
* **insert(index, text)** → Inserts text at a position (example: entry.insert(0, "Default")).
* You can set:
  + **show="\*"** → Mask input for passwords.
  + **width=20** → Set box size.

Tkinter Login Form Example:-

import tkinter as tk

# Create root window

root = tk.Tk()

root.title("Login Form")

root.geometry("400x450")

# Function for login

def login():

    username = entry\_username.get()

    password = entry\_password.get()

    if username == "Amit" and password == "1234":

        label\_message.config(text="✅ Login Successful!", fg="green")

    else:

        label\_message.config(text="❌ Invalid Username or Password", fg="red")

# Username label & entry

tk.Label(root, text="Username:", font=("Arial", 12)).pack(pady=5)

entry\_username = tk.Entry(root, font=("Arial", 12))

entry\_username.pack(pady=5)

# Password label & entry (masked)

tk.Label(root, text="Password:", font=("Arial", 12)).pack(pady=5)

entry\_password = tk.Entry(root, font=("Arial", 12), show="\*")

entry\_password.pack(pady=5)

# Login button

btn\_login = tk.Button(root, text="Login", command=login, bg="blue", fg="white", font=("Arial", 12))

btn\_login.pack(pady=10)

# Message label

label\_message = tk.Label(root, text="", font=("Arial", 12))

label\_message.pack(pady=5)

root.mainloop()

* **Text Widgets**

In Python’s **Tkinter**, a **Text widget** is a **multi-line text box**—unlike the Entry widget, which is only for **single-line input**.

It’s useful for:

* Writing paragraphs or notes
* Displaying logs or output
* Editing longer text

Basic Text Widget Example:-

import tkinter as tk

# Create root window

root = tk.Tk()

root.title("Text Widget Example")

root.geometry("400x300")

# Function to get text from widget

def get\_text():

    text\_content = text\_box.get("1.0", tk.END)  # From line 1, char 0 to end

    label.config(text=f"You typed:\n{text\_content}")

# Create a Text widget

text\_box = tk.Text(root, height=5, width=40, font=("Arial", 12))

text\_box.pack(pady=10)

# Button to read text

btn = tk.Button(root, text="Get Text", command=get\_text, bg="blue", fg="white")

btn.pack(pady=5)

# Label to display content

label = tk.Label(root, text="", font=("Arial", 12), wraplength=350, justify="left")

label.pack(pady=10)

root.mainloop()

**Key Points**

* **Text(root, height, width)** → Creates the widget.
* **get("start", "end")** → Retrieves text.
  + "1.0" → line 1, character 0 (start).
  + tk.END → the end of the text box.
* **insert(index, text)** → Adds text at a position (e.g., "1.0" for start).
* **delete(start, end)** → Removes text.
* Can handle **scrollbars** for large content.
* **Check buttons**

A Checkbutton lets the user select ON / OFF (True/False) options, like a checkbox in forms.  
It can be used for multiple selections.

Basic Checkbutton Example:-

import tkinter as tk

root = tk.Tk()

root.title("Checkbutton Example")

root.geometry("300x200")

# Variables to store checkbox states

var1 = tk.BooleanVar()

var2 = tk.BooleanVar()

var3 = tk.BooleanVar()

# Function to display selected options

def show\_selection():

    selected = []

    if var1.get():

        selected.append("Python")

    if var2.get():

        selected.append("Java")

    if var3.get():

        selected.append("C++")

    label.config(text=f"Selected: {', '.join(selected)}")

# Checkbuttons

cb1 = tk.Checkbutton(root, text="Python", variable=var1, onvalue=True, offvalue=False, command=show\_selection)

cb1.pack(anchor="w")

cb2 = tk.Checkbutton(root, text="Java", variable=var2, onvalue=True, offvalue=False, command=show\_selection)

cb2.pack(anchor="w")

cb3 = tk.Checkbutton(root, text="C++", variable=var3, onvalue=True, offvalue=False, command=show\_selection)

cb3.pack(anchor="w")

# Label to show output

label = tk.Label(root, text="Selected: None", font=("Arial", 12))

label.pack(pady=10)

root.mainloop()

**Key Points**

* **variable=var** → Links the Checkbutton to a Tkinter variable (IntVar, BooleanVar, or StringVar).
* **onvalue / offvalue** → Values when checked/unchecked.
* **command** → Function to run when the checkbox state changes.
* You can check state using:

**Real-world Example:-**

**🍕 Pizza Order Form in Tkinter**

import tkinter as tk

# Create root window

root = tk.Tk()

root.title("Pizza Order Form")

root.geometry("350x350")

# Variables for checkbuttons

cheese\_var = tk.BooleanVar()

pepperoni\_var = tk.BooleanVar()

mushrooms\_var = tk.BooleanVar()

# Function to process the order

def place\_order():

name = name\_entry.get()

toppings = []

if cheese\_var.get():

toppings.append("Cheese")

if pepperoni\_var.get():

toppings.append("Pepperoni")

if mushrooms\_var.get():

toppings.append("Mushrooms")

if not name:

output\_label.config(text="❌ Please enter your name", fg="red")

elif not toppings:

output\_label.config(text="❌ Please select at least one topping", fg="red")

else:

order = f"✅ {name}, your pizza with {', '.join(toppings)} is on the way!"

output\_label.config(text=order, fg="green")

# Name field

tk.Label(root, text="Customer Name:", font=("Arial", 12)).pack(pady=5)

name\_entry = tk.Entry(root, font=("Arial", 12))

name\_entry.pack(pady=5)

# Toppings checkbuttons

tk.Label(root, text="Select Toppings:", font=("Arial", 12)).pack(pady=5)

tk.Checkbutton(root, text="Cheese", variable=cheese\_var).pack(anchor="w", padx=50)

tk.Checkbutton(root, text="Pepperoni", variable=pepperoni\_var).pack(anchor="w", padx=50)

tk.Checkbutton(root, text="Mushrooms", variable=mushrooms\_var).pack(anchor="w", padx=50)

# Place order button

tk.Button(root, text="Place Order", command=place\_order, bg="orange", fg="white", font=("Arial", 12)).pack(pady=15)

# Output label

output\_label = tk.Label(root, text="", font=("Arial", 12), wraplength=300)

output\_label.pack(pady=10)

root.mainloop()

**Project To Create Inventory Management Using GUI**

import tkinter as tk

from tkinter import ttk

from datetime import date

import pandas as pd

import os

root = tk.Tk()

root.title("Dynamic Parts Order Form with Excel Storage")

root.geometry("750x550")

parts\_list = [f"Part {i}" for i in range(1, 51)]

part\_rows = []

# Function to add part row

def add\_part\_row():

    row\_frame = tk.Frame(parts\_frame)

    row\_frame.pack(fill="x", pady=2)

    part\_combo = ttk.Combobox(row\_frame, values=parts\_list, font=("Arial", 10), state="readonly", width=25)

    part\_combo.grid(row=0, column=0, padx=5)

    qty\_entry = tk.Entry(row\_frame, width=10)

    qty\_entry.grid(row=0, column=1, padx=5)

    amt\_entry = tk.Entry(row\_frame, width=10)

    amt\_entry.grid(row=0, column=2, padx=5)

    part\_rows.append((part\_combo, qty\_entry, amt\_entry))

# Function to save order to Excel

def save\_to\_excel(order\_data):

    file\_name = "orders.xlsx"

    # Create DataFrame from order data

    df\_new = pd.DataFrame(order\_data, columns=["Customer Name", "Date", "Part Name", "Quantity", "Amount", "Line Total"])

    if os.path.exists(file\_name):

        df\_existing = pd.read\_excel(file\_name)

        df\_combined = pd.concat([df\_existing, df\_new], ignore\_index=True)

        df\_combined.to\_excel(file\_name, index=False)

    else:

        df\_new.to\_excel(file\_name, index=False)

# Function to submit order

def submit\_order():

    name = entry\_name.get()

    order\_date = entry\_date.get()

    if not name or not order\_date:

        output\_label.config(text="❌ Please enter Customer Name and Date", fg="red")

        return

    total\_amount = 0

    order\_details = []

    excel\_data = []

    for part\_combo, qty\_entry, amt\_entry in part\_rows:

        part = part\_combo.get()

        qty = qty\_entry.get()

        amt = amt\_entry.get()

        if not part or not qty or not amt:

            continue

        try:

            qty = int(qty)

            amt = float(amt)

            line\_total = qty \* amt

            total\_amount += line\_total

            order\_details.append(f"{part} ({qty} × ₹{amt:.2f}) = ₹{line\_total:.2f}")

            excel\_data.append([name, order\_date, part, qty, amt, line\_total])

        except ValueError:

            output\_label.config(text="❌ Quantity & Amount must be numbers", fg="red")

            return

    if not order\_details:

        output\_label.config(text="❌ Please enter at least one part", fg="red")

    else:

        save\_to\_excel(excel\_data)

        details\_text = "\n".join(order\_details)

        output\_label.config(

            text=f"✅ Order Saved!\n\nOrder Summary for {name} ({order\_date}):\n{details\_text}\n\nTotal = ₹{total\_amount:.2f}",

            fg="green"

        )

# GUI Layout

tk.Label(root, text="Customer Name:", font=("Arial", 12)).pack(pady=5)

entry\_name = tk.Entry(root, font=("Arial", 12))

entry\_name.pack(pady=5)

tk.Label(root, text="Date (YYYY-MM-DD):", font=("Arial", 12)).pack(pady=5)

entry\_date = tk.Entry(root, font=("Arial", 12))

entry\_date.insert(0, date.today().strftime("%Y-%m-%d"))

entry\_date.pack(pady=5)

tk.Label(root, text="Parts List (Select Part, Qty, Amount):", font=("Arial", 12)).pack(pady=5)

parts\_frame = tk.Frame(root)

parts\_frame.pack(pady=5)

btn\_add\_part = tk.Button(root, text="Add Part", command=add\_part\_row, bg="orange", fg="white")

btn\_add\_part.pack(pady=5)

add\_part\_row()

btn\_submit = tk.Button(root, text="Submit Order", command=submit\_order, bg="blue", fg="white", font=("Arial", 12))

btn\_submit.pack(pady=10)

output\_label = tk.Label(root, text="", font=("Arial", 11), justify="left", wraplength=700)

output\_label.pack(pady=10)

root.mainloop()