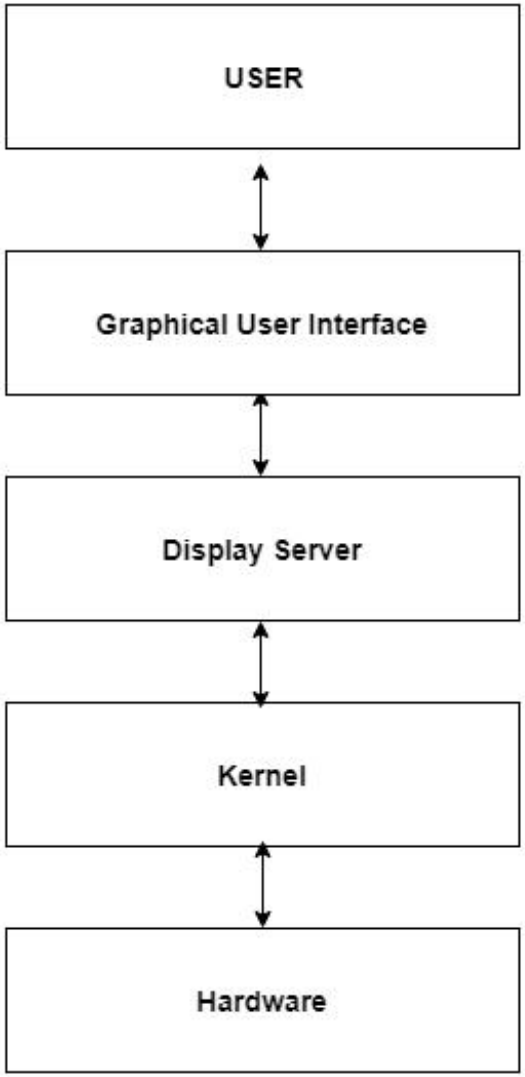
Module 5

PPT1

Graphical user interfaces

- GUI is an interface that allows users to interact with different electronic devices using icons and other visual indicators. The graphical user interfaces were created because command line interfaces were quite complicated and it was difficult to learn all the commands in it.
- In today's times, graphical user interfaces are used in many devices such as mobiles, MP3 players, gaming devices, smartphones etc.


USER
Graphical User Interface
Display Server
Kernel
Hardware

Elements in Graphical User Interface

- Graphical User Interface makes use of visual elements mostly. These elements define the appearance of the GUI. Some of these are described in detail as follows−
- **Window-**This is the element that displays the information on the screen. It is very easy to manipulate a window. It can be opened or closed with the click of an icon. Moreover, it can be moved to any area by dragging it around.In a multitasking environment, multiple windows can be open at the same time, all of them performing different tasks.
- There are multiple types of windows in a graphical user interface, such as container window, browser window, text terminal window, child window, message window etc.
- **Menu -**A menu contains a list a choices and it allows users to select one from them. A menu bar is displayed horizontally across the screen such as pull down menu. When any option is clicked in this menu, then the pull down menu appears.
- Another type of menu is the context menu that appears only when the user performs a specific action. An example of this is pressing the right mouse button. When this is done, a menu will appear under the cursor.

- **Icons**- Files, programs, web pages etc. can be represented using a small picture in a graphical user interface. This picture is known as an icon. Using an icon is a fast way to open documents, run programs etc. because clicking on them yields instant access.
- **Controls**- Information in an application can be directly read or influences using the graphical control elements. These are also known as widgets. Normally, widgets are used to display lists of similar items, navigate the system using links, tabs etc. and manipulating data using check boxes, radio boxes etc.
- **Tabs**- A tab is associated with a view pane. It usually contains a text label or a graphical icon. Tabs are sometimes related to widgets and multiple tabs allow users to switch between different widgets. Tabs are used in various web browsers such as Internet Explorer, Firefox, Opera, Safari etc. Multiple web pages can be opened in a web browser and users can switch between them using tabs.

Python - GUI Programming (Tkinter)

- Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.
- **Tkinter** – Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython** – This is an open-source Python interface for wxWindows http://wxpython.org.
- **JPython** – JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org

Tkinter Programming

- 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.

```
import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

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

1 Button- The Button widget is used to display buttons in your application.

2 Canvas-The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

3 Checkbutton-The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

4 Entry-The Entry widget is used to display a single-line text field for accepting values from a user.

5 Frame- The Frame widget is used as a container widget to organize other widgets.

6 Label-The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

7 Listbox-The Listbox widget is used to provide a list of options to a user.

8 Menubutton-The Menubutton widget is used to display menus in your application.

9 Menu-The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

10 Message-The Message widget is used to display multiline text fields for accepting values from a user.

11 Radiobutton-The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

12 Scale-The Scale widget is used to provide a slider widget.

13 Scrollbar-The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.

14 Text-The Text widget is used to display text in multiple lines.

15 Toplevel-The Toplevel widget is used to provide a separate window container.

16 Spinbox-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 PanedWindow-A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

18 LabelFrame-A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

19 tkMessageBox- This module is used to display message boxes in your applications.

BUTTON

- The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

w = Button (master, option=value, ...)

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.

Sr.No.	Option & Description
1	**activebackground** Background color when the button is under the cursor.
2	**activeforeground** Foreground color when the button is under the cursor.
3	**bd** Border width in pixels. Default is 2.
4	**bg** Normal background color.
5	**command** Function or method to be called when the button is clicked.
6	**fg** Normal foreground (text) color.
7	**font** Text font to be used for the button's label.
8	**height** Height of the button in text lines (for textual buttons) or pixels (for images).
9	**highlightcolor** The color of the focus highlight when the widget has focus.
10	**image** Image to be displayed on the button (instead of text).

Sr.No.	Option & Description
11	**justify** How to show multiple text lines: LEFT to left-justify each line; CENTER to center them; or RIGHT to right-justify.
12	**padx** Additional padding left and right of the text.
13	**pady** Additional padding above and below the text.
14	**relief** Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE.
15	**state** Set this option to DISABLED to gray out the button and make it unresponsive. Has the value ACTIVE when the mouse is over it. Default is NORMAL.
16	**underline** Default is -1, meaning that no character of the text on the button will be underlined. If nonnegative, the corresponding text character will be underlined.
17	**width** Width of the button in letters (if displaying text) or pixels (if displaying an image).
18	**wraplength** If this value is set to a positive number, the text lines will be wrapped to fit within this length.

Methods

Sr.No.	Method & Description
1	**flash()** Causes the button to flash several times between active and normal colors. Leaves the button in the state it was in originally. Ignored if the button is disabled.
2	**invoke()** Calls the button's callback, and returns what that function returns. Has no effect if the button is disabled or there is no callback.

Adding Label

```
import tkinter as tk
window = tk.Tk()
greeting = tk.Label(text="Hello, Tkinter")
greeting.pack()#to add widgets to a window, you can use the Label widget’s .pack() method:
window.mainloop()
#When you .pack() a widget into a window, Tkinter sizes the window as small as
# it can while still fully encompassing the widget.
```

You can control Label text and background colors using the foreground and background parameters:

```
import tkinter as tk
window = tk.Tk()
label = tk.Label(
    text="Hello, Tkinter",
    foreground="white",  # Set the text color to white
    background="black"  # Set the background color to black
)
label.pack()#to add widgets to a window, you can use the Label widget’s .pack() method:
window.mainloop()
```

- You can also specify a color using hexadecimal RGB values:

```
label = tk.Label(text="Hello, Tkinter",
background="#34A2FE")
```

- If you don’t feel like typing out foreground and background all the time, then you can use the shorthand fg and bg parameters to set the foreground and background colors:

```
label = tk.Label(text="Hello, Tkinter", fg="white",
bg="black")
```

- You can also control the width and height of a label with the width and height parameters:

```
import tkinter as tk
window = tk.Tk()
label = tk.Label(
    text="Hello, Tkinter",
    fg="white",
    bg="black",
    width=10,
    height=10
)
label.pack()#to add widgets to a window, you can use the Label widget’s .pack() method:
window.mainloop()
```

Displaying Clickable Buttons With Button Widgets#

```
import tkinter as tk
window = tk.Tk()
button = tk.Button(
    text="Click me!",
    width=25,
    height=5,
    bg="blue",
    fg="yellow",
)
button.pack()#to add widgets to a window, you can use the Label widget’s .pack() method:
window.mainloop()
```

Getting User Input With Entry Widgets#

```
import tkinter as tk
window = tk.Tk()
entry = tk.Entry(fg="yellow", bg="blue", width=50)
entry.pack()#to add widgets to a window, you can use the Label widget's .pack() method:
window.mainloop()
```

- The interesting bit about Entry widgets isn't how to style them, though. It's how to use them to **get input from a user**. There are three main operations that you can perform with Entry widgets:
  - **Retrieving text** with .get()
  - **Deleting text** with .delete()
  - **Inserting text** with .insert()

```
import tkinter as tk
def write_slogan():
    print("Tkinter is easy to use!")
root = tk.Tk()
frame = tk.Frame(root)
frame.pack()
button = tk.Button(frame,
                   text="QUIT",
                   fg="red",
                   command=quit())
button.pack(side=tk.LEFT)
slogan = tk.Button(frame,
                   text="Hello",
                   command=write_slogan)
slogan.pack(side=tk.LEFT)
root.mainloop()
```

Dynamical Content in a Label

```
import tkinter as tk
counter = 0
def counter_label(label):
  counter = 0
  def count():
    global counter
    counter += 1
    label.config(text=str(counter))
    label.after(1000, count)
  count()
root = tk.Tk()
root.title("Counting Seconds")
label = tk.Label(root, fg="dark green")
label.pack()
counter_label(label)
button = tk.Button(root, text='Stop', width=25, command=root.destroy)
button.pack()
root.mainloop()
```

Message widget

```
import tkinter as tk
master = tk.Tk()
whatever_you_do = "Whatever you do will be insignificant, but it is very important that you do it.\n(Mahatma Gandhi)"
msg = tk.Message(master, text = whatever_you_do)
msg.config(bg='lightgreen', font=('times', 24, 'italic'))
msg.pack()
tk.mainloop()
```

Option	Meaning
anchor	The position, where the text should be placed in the message widget: N, NE, E, SE, S, SW, W, NW, or CENTER. The Default is CENTER.
aspect	Aspect ratio, given as the width/height relation in percent. The default is 150, which means that the message will be 50% wider than it is high. Note that if the width is explicitly set, this option is ignored.
background	The background color of the message widget. The default value is system specific.
bg	Short for background.
borderwidth	Border width. Default value is 2.
bd	Short for borderwidth.
cursor	Defines the kind of cursor to show when the mouse is moved over the message widget. By default the standard cursor is used.
font	Message font. The default value is system specific.
foreground	Text color. The default value is system specific.
fg	Same as foreground.

Option	Meaning
highlightcolor	See highlightbackground.
highlightthickness	See highlightbackground.
justify	Defines how to align multiple lines of text. Use LEFT, RIGHT, or CENTER. Note that to position the text inside the widget, use the anchor option. Default is LEFT.
padx	Horizontal padding. Default is -1 (no padding).
pady	Vertical padding. Default is -1 (no padding).
relief	Border decoration. The default is FLAT. Other possible values are SUNKEN, RAISED, GROOVE, and RIDGE.
takefocus	If true, the widget accepts input focus. The default is false.
text	Message text. The widget inserts line breaks if necessary to get the requested aspect ratio. (text/Text)
textvariable	Associates a Tkinter variable with the message, which is usually a StringVar. If the variable is changed, the message text is updated.
width	Widget width given in character units. A suitable width based on the aspect setting is automatically chosen, if this option is not given.

Radio Buttons

- A radio button, sometimes called option button, is a graphical user interface element of Tkinter, which allows the user to choose (exactly) one of a predefined set of options. Radio buttons can contain text or images.

```
import tkinter as tk
root = tk.Tk()
v = tk.IntVar()
tk.Label(root,
      text="""Choose a
programming language:""",
      justify = tk.LEFT,
      padx = 20).pack()
tk.Radiobutton(root,
           text="Python",
           padx = 20,
           variable=v,
           value=1).pack(anchor=tk.W)
tk.Radiobutton(root,
           text="Perl",
           padx = 20,
           variable=v,
           value=2).pack(anchor=tk.W)
root.mainloop()
```

In many cases, there are more than two radio buttons. It would be cumbersome, if we have to define and write down each button. The solution is shown in the following example. We have a list "languages", which contains the button texts and the corresponding values. We can use a for loop to create all the radio buttons.

```
import tkinter as tk
root = tk.Tk()
v = tk.IntVar()
v.set(1)  # initializing the choice, i.e. Python
languages = [ ("Python",1), ("Perl",2), ("Java",3), ("C++",4),  ("C",5)]
def ShowChoice():
   print(v.get())
tk.Label(root,
        text="""Choose your favourite programming language:""",
        justify = tk.LEFT,
        padx = 20).pack()
for val, language in enumerate(languages):
   tk.Radiobutton(root,
                  text=language,
                  padx = 20,
                  variable=v,
                  command=ShowChoice,
                  value=val+1).pack(anchor=tk.W)
root.mainloop()
```

Indicator

Instead of having radio buttons with circular holes containing white space, we can have radio buttons with the complete text in a box. We can do this by setting the indicatoron (stands for "indicator on") option to 0, which means that there will be no separate radio button indicator. The default is 1.

```
import tkinter as tk
root = tk.Tk()
v = tk.IntVar()
v.set(1)  # initializing the choice, i.e. Python
languages = [ ("Python",1), ("Perl",2), ("Java",3), ("C++",4),  ("C",5)]
def ShowChoice():
   print(v.get())
tk.Label(root,
        text="""Choose your favourite programming language:""",
        justify = tk.LEFT,
        padx = 20).pack()
for val, language in enumerate(languages):
   tk.Radiobutton(root,
                  text=language,
                  indicatoron = 0,
                  width = 20,
                  padx = 20,
                  variable=v,
                  command=ShowChoice,
                  value=val+1).pack(anchor=tk.W)
root.mainloop()
```

Checkboxes

- Checkboxes, also known as tickboxes or tick boxes or check boxes, are widgets that permit the user to make multiple selections from a number of different options. This is different to a radio button, where the user can make only one choice.

```
from tkinter import *
master = Tk()
var1 = IntVar()
Checkbutton(master, text="male",
variable=var1).grid(row=0, sticky=W)
var2 = IntVar()
Checkbutton(master, text="female",
variable=var2).grid(row=1, sticky=W)
mainloop()
```

We can improve this example a little bit. First we add a Label to it. Furthermore we add two Buttons, one to leave the application and the other one to view the values var1 and var2.

```
from tkinter import *
master = Tk()
def var_states():
   print("male: %d,\nfemale: %d" % (var1.get(), var2.get()))
Label(master, text="Your sex:").grid(row=0, sticky=W)
var1 = IntVar()
Checkbutton(master, text="male", variable=var1).grid(row=1, sticky=W)
var2 = IntVar()
Checkbutton(master, text="female", variable=var2).grid(row=2,
sticky=W)
Button(master, text='Quit', command=master.quit).grid(row=3, sticky=W,
pady=4)
Button(master, text='Show', command=var_states).grid(row=4, sticky=W,
pady=4)
mainloop()
```

Entry Widgets

- Entry widgets are the basic widgets of Tkinter used to get input, i.e. text strings, from the user of an application. This widget allows the user to enter a single line of text.

```
import tkinter as tk
master = tk.Tk()
tk.Label(master, text="First Name").grid(row=0)
tk.Label(master, text="Last Name").grid(row=1)
e1 = tk.Entry(master)
e2 = tk.Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
master.mainloop()
```

```
import tkinter as tk
def show_entry_fields():
   print("First Name: %s\nLast Name: %s" % (e1.get(), e2.get()))
master = tk.Tk()
tk.Label(master,
        text="First Name").grid(row=0)
tk.Label(master,
        text="Last Name").grid(row=1)
e1 = tk.Entry(master)
e2 = tk.Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
tk.Button(master,
         text='Quit',
         command=master.quit).grid(row=3,
                                   column=0,
                                   sticky=tk.W,
                                   pady=4)
tk.Button(master,
         text='Show', command=show_entry_fields).grid(row=3,
                                                      column=1,
                                                      sticky=tk.W,
                                                      pady=4)
tk.mainloop()
```

Dialogues and Message Boxes

```
import tkinter as tk
from tkinter import messagebox as mb
def answer():
    mb.showerror("Answer", "Sorry, no answer available")
def callback():
    if mb.askyesno('Verify', 'Really quit?'):
        mb.showwarning('Yes', 'Not yet implemented')
    else:
        mb.showinfo('No', 'Quit has been cancelled')
tk.Button(text='Quit', command=callback).pack(fill=tk.X)
tk.Button(text='Answer', command=answer).pack(fill=tk.X)
tk.mainloop()
```