1. Introduction to GUI Programming:

WHAT IS GUI?

GUI stands for **Graphical User Interface**.A graphics-based operating system interface that uses icons, menus and a mouse (to click on the icon or pull down the menus) to manage interaction with the system.A comprehensive **GUI environment** includes **four components**:

1. **Graphics library**
2. **User interface toolkit**
3. **User interface style guide**
4. **Consistent applications.**

1. Graphic Library: The graphics library provides a high-level graphics programming interface.

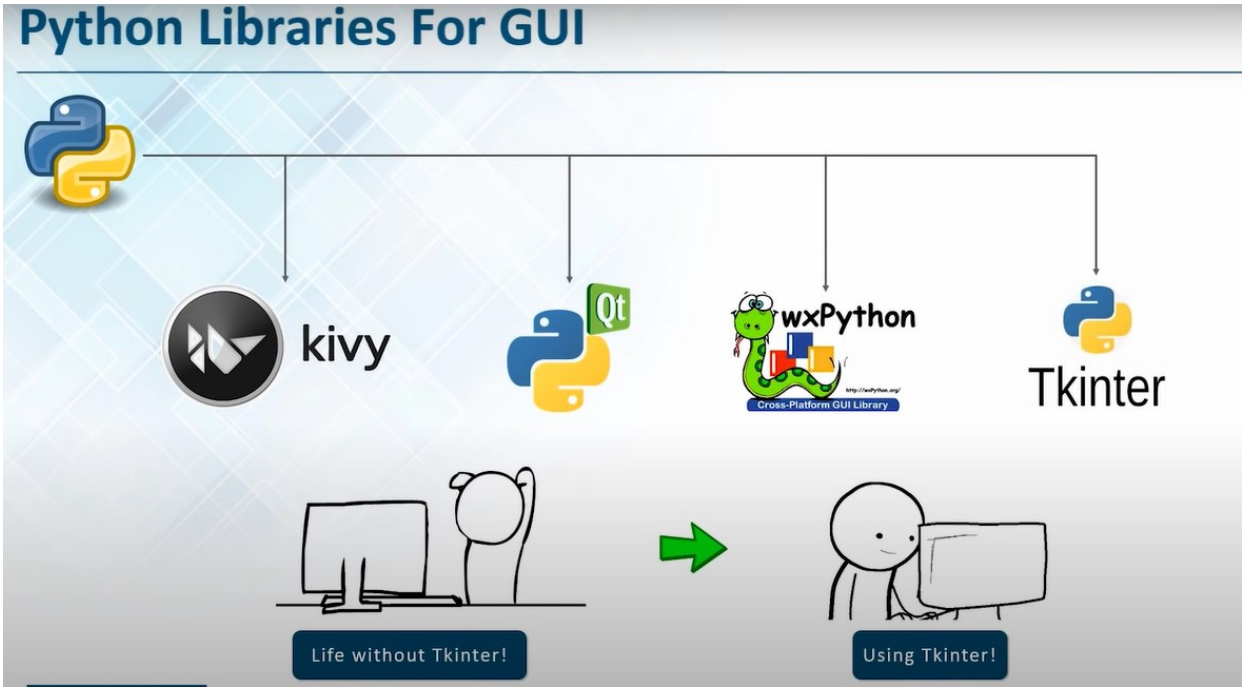
2. User Interface Toolkit: The user interface toolkit, built on top of the graphics library, provides application programs with mechanisms for creating and managing the dialogue elements of the windows, icons, menus, pointers and scroll bars (WIMPS) interface.

3. USer Interface Style Guide: The user interface style guide specifies how applications should employ the dialogue elements to present a consistent, easy-to-use environment (i.e., "look and feel") to the user.

4. Consistent Application: Application program conformance with a single user interface style is the primary determinant of ease of learning and use, and thus, of application effectiveness and user productivity.

PYTHON LIBRARIES FOR GUI:

1. **KIVY**
2. **QT**
3. **WXPYTHON**
4. **TKINTER**



INTRODUCTION TO TKINTER:

- Python has a lot of GUI frameworks, but **Tkinter** is the only framework **that's built into the Python standard library**. Tkinter has several strengths. **It's cross-platform**, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

- Although Tkinter is considered the de facto Python GUI framework, it's not without criticism. One notable **criticism** is that **GUIs built with Tkinter look outdated**. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

- Tkinter is **lightweight and relatively painless** to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top **priority is to quickly build** something that's functional and cross-platform.

FUNDAMENTALS OF TKINTER:

1. **IMPORT THE TKINTER MODULE**
2. **CREATE THE GUI APPLICATION MAIN WINDOW**
3. **ADD WIDGETS**
4. **ENTER THE MAIN EVENT LOOP**

WIDGETS:

DEFINITION: Tkinter provides us with a variety of common GUI elements which we can use to build out interface – such as buttons, menus and various kind of entry fields and display areas. We call these elements Widgets.

- In general, Widget is an element of Graphical User Interface (GUI) that displays/illustrates information or gives a way for the user to interact with the OS. In Tkinter , Widgets are objects ; instances of classes that represent buttons, frames, and so on.

- **Each separate widget is a Python object.**

Widgets are basic building blocks of GUI programming and are used to display information or get input from the user.Some are as follows:

1. | **BUTTON:|** The Button widget is used to display buttons in your application.|

1. **CANVAS:** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

1. **CHECK BUTTON:** The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

1. **ENTRY:** The Entry widget is used to display a single-line text field for accepting values from a user.

1. **FRAME:** The Frame widget is used as a container widget to organize other widgets.

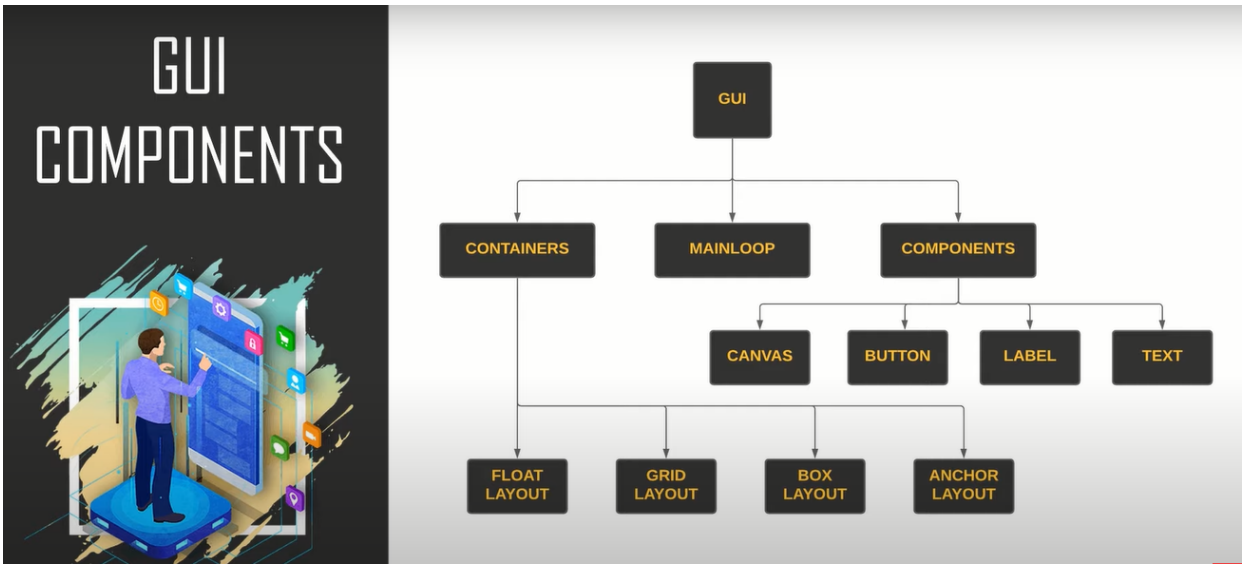
1. **LABEL:** The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

1. **LISTBOX:** The Listbox widget is used to provide a list of options to a user.

1. **MENU BUTTON:** The Menubutton widget is used to display menus in your application.

1. **MENU:** The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
    - **Message:** The Message widget is used to display multiline text fields for accepting values from a user.
    - **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.
    - **Scale:** The Scale widget is used to provide a slider widget.
    - **ScrollBar:** The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
    - **Text:** The Text widget is used to display text in multiple lines.
    - **Toplevel:** The Toplevel widget is used to provide a separate window container.
    - **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.
    - **PanedWindow:** A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
    - **LabelFrame:** A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
    - **tkMessageBox:** This module is used to display message boxes in your applications.

GUI COMPONENTS: These are the components of GUI:



In [ ]:
```
from tkinter import *
```

In [ ]:
```
#Creation Of Windows

from tkinter import Tk
root=Tk()
root.mainloop()

## Tk() is the class which is used to create the root windows of the application.
```

In [ ]:
```
#  To set the title of the Tkinter window

from tkinter import Tk
root=Tk()
root.title("Mohit Sir is best")
root.mainloop()
```

In [ ]:
```
#To check the screen size

from tkinter import Tk
root=Tk()
screen_width=root.winfo_screenwidth()
screen_height=root.winfo_screenheight()
print("Screen Width:",screen_width)
print("Screen Height:",screen_height)
```

In [ ]:
```
#To make windows full screen

from tkinter import Tk
root=Tk()
root.attributes('-fullscreen',True)
root.mainloop()
```

In [1]:
```
#To resize the windows

from tkinter import Tk
root=Tk()
root.geometry("420x500+200+200")  #Make windows 420X500 and place at position(200,200)
root.mainloop()
```

In [ ]:
```
#To create label as a child root of the window

from tkinter import Tk
root=Tk()
msg=Label(root,text='GUI programming with Python using Tkinter')
msg.pack()  #The pack() method is used on ech widget to position it inside its parent.
root.mainloop()
```

In [ ]:
```
#To change the font size of the text
```

```
from tkinter import Tk
root=Tk()
msg=Message(root,text='GUI programming with Python using Tkinter')
msg.config(font=('algerian',33,'italic bold underline'))
msg.pack()
root.mainloop()
```

In [ ]:

```
#To display two different labels with colored background

from tkinter import Tk,Label,Y,RIGHT
root=Tk()
Label1=Label(root,text='Powerful People', background='red')
Label2=Label(root,text='Makes Places Powerful',background='green')
Label1.pack(fill=X,padx=10,ipady=25, side=RIGHT)
Label2.pack(fill=X,padx=20,ipady=40, side=RIGHT)
root.mainloop()
```

Some of the PAcking Options:

- **Fill:** Widget expands to take up any extra space (X,Y or BOTH)
- **Padx/Pady:** Outter Padding
- **ipadx/ipady:** Inner Padding
- **Side:** Which side to stack from.Default is TOP (to bottom)

In [ ]:

```
#To display an image

# Import required libraries
from tkinter import *
from PIL import ImageTk, Image

# Create an instance of tkinter window
win = Tk()

# Define the geometry of the window
win.geometry("700x500")

frame = Frame(win, width=600, height=400)
frame.pack()
frame.place(anchor='center', relx=0.5, rely=0.5)

# Create an object of tkinter ImageTk
img = ImageTk.PhotoImage(Image.open("Rico.png"))

# Create a Label Widget to display the text or Image
label = Label(frame, image = img)
label.pack()

win.mainloop()
```

In [ ]:

```
#Add two buttons and print a message when it is clicked

from tkinter import Tk,Button
root=Tk()
exitButton=Button(root,text='EXIT',command=root.destroy)
```

```
exitButton.pack()

def My_callback():
    print("You Clicked the Message Button")
msg_button=Button(root, text='Click Here',command=My_callback)
msg_button.pack()
root.mainloop()
```

In [ ]:

```
#To print a colored text on a colored background GUI window

from tkinter import Tk,Label
root=Tk()
label=Label(root,text='Powerful People Comes From Powerful Places!!!!')
label.pack()
label.config(foreground='red',background='blue',text="Updated Text")
root.mainloop()
```

In [ ]:

```
#To display Menu on the Menu Bar

from tkinter import Tk,Menu
root=Tk()
#Create Menu Bar
menu_bar=Menu(root)

#Create Sub-Menu

fileMenu=Menu(menu_bar,tearoff=0)

#Add commands to Sub-Menu
fileMenu.add_command(label="Stop",command=root.destroy)
fileMenu.add_command(label="Kill",command=root.destroy)
fileMenu.add_command(label="Exit",command=root.destroy)

#Add the file drop down sub-menu in the main menu bar

menu_bar.add_cascade(label="File",menu=fileMenu)
root.config(menu=menu_bar)
root.mainloop()
```

In [ ]:

```
#To print message on the IDLE screen

from tkinter import Tk,Entry,Button,INSERT
root=Tk()

#Create an Entry Box
entry=Entry(root)
entry.pack()

#Print the contents of entry box in Console

def Printmsg():
    print(entry.get())

#Create a button that when clicked will orint the contents of the entry box

button=Button(root, text='Print Content',command=Printmsg)
```

```
button.pack()
root.mainloop()
```

In [ ]:

```
#To display text on console when a label is pressed.

from tkinter import Tk,Label
root=Tk()
label=Label(root,text='Printing Label...')
label.pack()

def my_callback():
    print("XYZ")
label.bind("<Button-1>",lambda e:my_callback)
root.mainloop()
```

In [1]:

```
#To display POP-UP dialog box

from tkinter import messagebox
title='Customer Feedback'
text='You Like Our Services?'
reply=messagebox.askquestion(title,text)
if reply=='yes':
    print("TYSM")
else:
    print("SOrry")
```

```
SOrry
```

In [ ]:

```
print(dir(tkinter))
```

In [ ]:

```
(help(Y))
```