**Graphical User Interface**

The user can interact with an application through graphics or an image is called GUI [Graphical User Interface]. Here the user need not remember any commands. User can perform task just by clicking on relevant images.

**Advantages:**

1. It is user friendly.
2. It adds attraction and beauty to any application by adding pictures, colors, menus, animation.
3. It is possible to simulate the real life objects using GUI.
4. GUI helps to create graphical components like push button, radio button, check box, text box, menus etc.,

**GUI in Python:**

Python offers **tkinter**module to create graphical programs. The **tkinter**represents ‘toolkit interface’ for GUI. This is an interface for Python programmers that enable them to the classes of TK module of TCL/TK [Tool Command Language]. TCL language use TK [Tool Kit] language to generate graphics.

**General steps involved in basic GUI programs:**

**1. Create a root Window:** The root window is the top level window that provides rectangular space on the screen where we can display text, color, images, etc.

 First import tkinter module

from tkinter import \*

 root=Tk() # create root window object.

**2. Create a Canvas / Frame:** Canvas and frame are child windows in the root window.

C=Canvas (root, bg=‘blue’, height=500, width=400, cursor=‘pencil’)

Cursors are – circle, hand1, hand2, heart, pencil, plus, mouse, star, watch etc…

**3. Create Widgets – Components:**

For Canvas – create line, circle, rectangle, any geometric objects

For Frame – buttons, text box, list, check box etc…

**4. Create an Event Handler**

Write the function for corresponding event.

**WAP to draw line, oval, polygon, rectangle and text**

from tkinter import \*

root=Tk()

root.title ("My Window")

#root.geometry ("400x300")

c=Canvas (root, bg="blue", height=700, width=1200, cursor='pencil')

id=c.create\_line (50, 50,200,50,200,150,width=4,fill="white")

id=c.create\_oval(100,100,400,300,width=2,fill="yellow", activefill="blue")

id=c.create\_polygon(10,10,200,200,300,200,width=3,fill="green",outline="red",activefill="Pink")

id=c.create\_rectangle(500,200,700,600,width=2,fill="gray",outline="yellow",activefill="green")

fnt=('Times',40,'bold italic underline')

id=c.create\_text(500,100,text="Python Graphics", font=fnt, fill="White", activefill="red")

c.pack()

root.mainloop()

**Frame:** A frame is similar to canvas, but it can hold components of forms. To create a frame, we can create an object of Frame class as:

**F= Frame (root, height=400, width=500, bg=“yellow”, cursor=“cross”**

Here, ‘F’ is an object of class Frame, the options height and width represents the area of frame in pixels, ‘bg’ represents the back ground color to be displayed and ‘cursor’ indicates the type of the cursor to be displayed in the frame.

Once the frame is created, it should be added to the root window using the **pack ()** method.

**Widgets / form elements:**

Widgets is a GUI component that is displayed on the screen and can perform a task as designed by the user. The widgets are:

**1. Button 2. Label 3. Text box 4. Message 5. Checkbox 6. List box 7. Option button 8. Scroll bars 9. Menus**

**Steps to create a Widgets**

**1. Create a widget object**

B=button (f, text=“Submit”)

**2. Define the event performed by the widget**

def buttonClick (self):

print (“You have clicked submit button”)

**3. Clicking event should be linked with the callback handler**

B**.**bind (‘<Button-1>’, buttonClick)

**4. Call event loop**

root.mainloop ()

**1. Button:** A push button is a component that performs some action when clicked. These buttons are created as objects of Button class as,

**b=Button (f, text="Ok", width=15, height=3, bg="yellow", fg="blue")**

Here, ‘**b’** is the object of Button class, ‘**f’** represents the frame for which the button is created as a child. The ‘**text’** option represents the text to be display on the button, **width** and **height**

represents the size of the button, ‘**bg’** represents the background color and ‘fg’ represents the fore ground color of the button,

**b.bind ("<Button-1>", buttonClick)**

We link the mouse left button with the buttonClick ( ) method using bind () method.

**2. Text:** A text widget is same as label. But text widget has several options and can display multiple lines of text in different colors and fonts. It is possible to insert text into a text widget, modify it or delete it. We can also display images in the text widget. A text is created as an object of Text class as,

**t=Text (f, width=20, height=3, font= ('Times', 20, 'italic'), fg='blue', bg="yellow", wrap=WORD)**

Here, ‘t’is the object of Text class, ‘**f’** represents the frame for which the text is created as a child.

The **width** and **height** represents the size of the text box, ‘**bg’** represents the background color and ‘fg’ represents the fore ground color of the text, ‘**font‘ represents** a tuple that contains font

name, size and style. The ‘**wrap’** represents the text information can be align with in the text box.

**3. Label:** A label represents the constant text that is displayed in the frame or container. A label can display one or more lines of text that cannot be modified. A label is created as an object of Label class as,

**l=Label (f, text="Label Demo", width=25, height=3, font= ('Times', 14, 'bold'), fg='blue', bg='yellow')**

Here, ‘**l’** is the object of Label class, ‘**f’** represents the frame for which the button is created as a child. The ‘**text’** option represents the text to be display on the Label box, **width** and **height** represents the size of the label, ‘**bg’** represents the background color and ‘fg’ represents the fore ground color of the label, ‘**font ‘** represents a tuple that contains font name, size and style.

**4. Check button:** Check buttons are also known as check boxes are useful for the user to select one or more options from available group of options. Check buttons are displayed in the form of square shaped boxes. When check box is selected, a tick mark is displayed on the check box. Check box is created as an object of Checkbutton class as,

**c1=Checkbutton (f, text="Python", bg='yellow', fg='red', font= ('Times', 20,'italic'))**

Here, ‘**c1’** is the object of Checkbutton class, ‘**f’** represents the frame for which the check button is created as a child. The ‘**text’** option represents the text to be display on the check box, ‘**bg’**

Represents the background color and ‘fg’ represents the fore ground color of the check box, ‘**font‘ represents** a tuple that contains font name, size and style.

**5. Radio button:** A radio button is similar to a check box, but it is used to select only one option from a group of available options. A radio button is displayed in the form of round shaped button. The user cannot select more than one option. When a radio

button is selected, there appears a dot in the radio button. We can create a radio button as an object of the Radiobutton class as,

**r1=Radiobutton (f, text="male", bg='green', fg='red', font= ('Times', 20, 'italic'), value=0)**

Here, ‘**r1’** is the object of Radiobutton class, ‘**f’** represents the frame for which the radio button is created as a child. The ‘**text’** option represents the text to be display on the radio button, ‘**bg’**

represents the background color and ‘fg’ represents the fore ground color of the radio button, ‘**font ‘** represents a tuple that contains font name, size and style, ‘**value’** represents a value that is set to this object when the radio button is clicked..

**Programming Examples**

**WAP to illustrate Button, Label, Text, Checkbox and Radio button**

**from** tkinter **import** \*

**def** buttonClick (self):

**print** (**"You have cliked Ok button"**)

root=Tk()

root.title (**"Button Example"**)

f=Frame (root, bg=**"blue"**, height=700, width=1200, cursor=**'cross'**)

f.propagate (0)

f.pack ()

b=Button (f, text=**"Ok"**, width=15,height=3, bg=**"yellow"**, fg=**"blue"**)

b.pack ()

b.bind (**"<Button-1>"**, buttonClick)

l=Label (f, text=**" Label Demo"**, width=25, height=3,

font= (**'Times'**, 14, **'bold'**), fg=**'blue'**, bg=**'yellow'**)

l.pack ()

t=Text (f, width=20, height=3, font= (**'Times'**, 20, **'italic'**), fg=**'blue'**, bg=**"yellow"**, wrap=WORD)

t.insert (END,**" Text Demo"**)

t.pack (side=LEFT)

c1=Checkbutton (f, text=**"Python"**, bg=**'yellow'**, fg=**'red'**,

font= (**'Times'**, 20, **'italic'**))

c1.pack (side=LEFT)

c2=Checkbutton (f, text=**"Networking"**, bg=**'yellow'**, fg=**'red'**,

font= (**'Times'**, 20, **'italic'**))

c2.pack (side=LEFT)

c3=Checkbutton (f, text=**"Java"**, bg=**'yellow'**, fg=**'red'**,

font= (**'Times'**, 20, **'italic'**))

c3.pack (side=LEFT)

r1=Radiobutton (f, text=**"male"**, bg=**'green'**, fg=**'red'**,

font= (**'Times'**, 20, **'italic'**), value=0)

r1.pack (side=LEFT)

r2=Radiobutton (f, text=**"female"**, bg=**'green'**, fg=**'red'**,

font= (**'Times'**, 20, **'italic'**), value=1)

r2.pack (side=LEFT)

root.mainloop ()

**WAP to draw BAR chart/graph**

**import** matplotlib.pyplot **as** plt

left = [1, 2, 3, 4, 5]

height = [10, 24, 0, 4, 50]

tick\_label = [**'one'**, **'two'**, **'three'**, **'four'**, **'five'**]

plt.bar (left, height, tick\_label = tick\_label,

width = 0.8, color = [**'red'**, **'green'**])

plt.xlabel (**'x - axis'**)

plt.ylabel (**'y - axis'**)

plt.title (**'My bar chart!'**)

plt.show()

**WAP to draw PIE chart/graph**

**import** matplotlib.pyplot **as** plt

activities = [**'routine'**, **'sleep'**, **'work'**, **'play'**]

slices = [3, 7, 8, 6]

colors = [**'r'**, **'y'**, **'g'**, **'b'**]

plt.pie(slices, labels = activities, colors=colors,

startangle=90, shadow = True, explode = (0, 0.1, 0, 0),

radius = 1.2, autopct = **'%1.2f%%'**)

plt.legend()

plt.show()