Programming I Lab Manual
GUI Programing Using Tkinter

Content in this module **page no**

Introduction to Tkinter

The purpose of this manual is to introduce you to the basics of GUI programming in Python, using Tkinter. There are several GUI interfaces available in Python:

Tkinter is the Python interface to the Tk GUI toolkit. WxPython is an open-source Python interface for wxWindows. JPython is a Python port for Java which gives Python scripts access to Java class libraries. Many others are also available. We will use Tkinter, due to the fact that it is the de facto standard Python GUI library.

What is Tk?

If Tkinter is the Python interface to the Tk GUI toolkit, what is Tk?

Tk started life as Tcl extension (1991). It is now written in C. Tk is a high-level windowing toolkit. You can interface with it directly using C or other languages.

Tk interfaces are also available in Python, Ruby, Perl, Tcl, and probably other languages. What Tk itself is interfacing with depends on your system:

- I Mac: Tk provides interfaces to the MacOS windowing system.
- I Windows: Tk provides interfaces to the Microsoft windowing system.
- I other platforms: Tk 8.X attempts to look like the Motif window manager, but without using Motif libraries. Prior to that it interfaced with the X window system.

Let it suffice to say that Tk provides a high-level means of accessing your system's windowing infrastructure.

Example Code One(creating an empty simple window)

```
# firstTkinter.py
#import the module with all the class(*)
from tkinter import *
#creat object of Tk class
top = Tk()
# Run the The 'mainloop' command isused to launch the window, and start the
#event loop.
top.mainloop()
```

Example Code Two(configured window)

Let's configure the default window with some parameters of the window

```
# firstTkinter2.py
from tkinter import *
top = Tk()
# Give the window a title.
top.title("My App")
# Give the window icon.
top.iconbitmap("m.ico")#relative path of the icon
# Change the minimum size.
top.minsize(400, 400)
# Change the background colour.
top.configure(bg = "green")
# Run the widget.
top.mainloop()
```

The Widget Classes

Tkinter's GUI classes define common GUI widgets such as buttons, labels, radio buttons, check buttons, entries, canvases, and others.

- **Button: -** A simple button, used to execute a command.
- **Canvas:-**Structured graphics, used to draw graphs and plots, create graphics editors, and implement custom widgets.
- **Checkbutton: -** Clicking a check button toggles between the values.
- **Entry:-**A text entry field, also called a text field or a text box.
- **Frame:-**A container widget for containing other widgets.
- **Label:-**Displays text or an image.
- **Menu:-**A menu pane, used to implement pull-down and popup menus.
- **Menubutton: -** A menu button, used to implement pull-down menus.
- **Message: -** Displays a text. Similar to the label widget, but can automatically wrap text to a given width or aspect ratio.
- **Radiobutton:-**Clicking a radio button sets the variable to that value, and clears all other radio buttons associated with the same variable.
- **Text :-**Formatted text display. Allows you to display and edit text with various styles and attributes. Also supports embedded images and windows.

Example Code One

```
#simple GUI with one label and button
from tkinter import *
window = Tk()  # Create a window
label = Label(window, text = "Welcome to Python")  # Create a label
button = Button(window, text = "Click Me")  # Create a button
label.pack()  # Place the label in the window
button.pack()  # Place the button in the window
window.mainloop()  # Create an event loop
```

Example Code Two

```
#adding diffrent wigets
from tkinter import *
# Create a window
window = Tk()
# Set a title
window.title("Widgets Demo")
# Create and add a frame to window
frame1 = Frame(window)
frame1.pack()
# Add a check button, and a radio button to frame1
cbtBold = Checkbutton(frame1, text = "Bold")
rbRed = Radiobutton(frame1, text = "Red", bg = "red", value = 1)
rbYellow = Radiobutton(frame1, text = "Yellow", bg = "yellow", value = 2)
cbtBold.grid(row = 1, column = 1)
rbRed.grid(row = 1, column = 2)
rbYellow.grid(row = 1, column = 3)
# Create and add a frame to window
frame2 = Frame(window)
frame2.pack()
# Add a label, an entry, a button, and a message to frame2
label = Label(frame2, text = "Enter your name: ")
entryName = Entry(frame2 )
btGetName = Button(frame2, text = "Get Name")
message = Message(frame2, text = "It is a widgets demo")
label.grid(row = 1, column = 1)
entryName.grid(row = 1, column = 2)
btGetName.grid(row = 1, column = 3)
message.grid(row = 1, column = 4)
# Add text
text = Text(window) # Create and add text to the window
text.pack()
text.insert(END, "Tip\nThe best way to learn Tkinter is to read ")
text.insert(END,  "these carefully designed examples and use them ")
text.insert(END, "to create your applications.")
window.mainloop() # Create an event loop
```

Example Code Three

```
#CanvasDemo
from tkinter import * # Import all definitions from tkinter
window = Tk() # Create a window
window.title("Canvas Demo") # Set title
# Place canvas in the window
canvas = Canvas(window, width = 200, height = 100, bg = "yellow")
canvas.pack()
# Place buttons in frame
frame = Frame(window)
frame.pack()
btRectangle = Button(frame, text = "Rectangle")
btOval = Button(frame, text = "Oval")
window.mainloop() # Create an event loop
```

Layout management in Tkinter

In this part of the Tkinter programming tutorial, we will introduce layout managers.

When we design the GUI of our application, we decide what widgets we will use and how we will organise those widgets in the application. To organise our widgets, we use specialised non visible objects called layout managers.

There are two kinds of widgets: containers and their children. The containers group their children into suitable layouts.

Tkinter has three built-in layout managers. The `pack`, `grid`, and `place` managers. The `pack` geometry manager organises widgets in horizontal and vertical boxes. The `grid` geometry managers places widgets in a two dimensional grid. Finally, the `place` geometry manager places widgets on their containers using absolute positioning.

The Grid manager

Example Code One

```
#grid demo

from tkinter import * # Import all definitions from tkinter

#create your own class

class GridManagerDemo:

   window = Tk() # Create a window

   window.title("Grid Manager Demo") # Set title

   message = Message(window, text =  "This Message widget occupies three rows and two columns")

   message.grid(row = 1, column = 1, columnspan = 2)

   Label(window, text = "First Name:").grid(row = 1, column = 3)

   Entry(window).grid(row = 1, column = 4, padx=3 , pady = 5)

   Label(window, text = "Last Name:").grid(row = 2, column = 3)

   Entry(window).grid(row = 2, column = 4)

   Button(window, text = "Get Name").grid(row = 3, padx = 5, pady = 5, column = 4, )

   window.mainloop() # Create an event loop

GridManagerDemo() # Create GUI
```

The Pack manager

Example Code one

```
#pack demo

from tkinter import * # Import all definitions from tkinter

class PackManagerDemo:

  def __init__(self):

    window = Tk() # Create a window

    window.title("Pack Manager Demo 1") # Set title

    Label(window, text = "Blue", bg = "blue").pack()

    Label(window, text = "Red", bg = "red").pack()

    Label(window, text = "Green", bg = "green").pack()

    window.mainloop() # Create an event loop

PackManagerDemo() # Create GUI
```

Example Code Two

```
from Tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack(fill=X)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(fill=X)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(fill=X)

mainloop()
```

Example Code Three

```
from Tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack(side=LEFT)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(side=LEFT)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(side=LEFT)

mainloop()
```

The Place manager

Example Code one

```
#place demo

from tkinter import * # Import all definitions from tkinter

class PlaceManagerDemo:

  def __init__(self):

    window = Tk() # Create a window

    window.title("Place Manager Demo") # Set title

    Label(window, text = "Blue", bg = "blue").place(x = 20, y = 20)

    Label(window, text = "Red", bg = "red").place(x = 50, y = 50)

    Label(window, text = "Green", bg = "green").place(x = 80, y = 80)

    window.mainloop() # Create an event loop

 PlaceManagerDemo() # Create GUI
```

Menus

You can use Tkinter to create menus, popup menus, and toolbars.Tkinter provides a comprehensive solution for building graphical user interfaces. This section introduces menus, popup menus, and toolbars.

Menus make selection easier and are widely used in windows. You can use the Menu class to create a menu bar and a menu, and use the add_command method to add items to the menu.

Example Code One

```
from tkinter import *

window = Tk()

window.title("Menu Demo")

# Create a menu bar

menubar = Menu(window)

# Display the menu bar

window.config(menu = menubar)

# Create a pull-down menu, and add it to the menu bar

operationMenu = Menu(menubar, tearoff = 0)

menubar.add_cascade(label = "Operation", menu = operationMenu)

operationMenu.add_command(label = "Add")

operationMenu.add_command(label = "Subtract")

operationMenu.add_separator()

operationMenu.add_command(label = "Multiply")

operationMenu.add_command(label = "Divide")

# Create more pull-down menus
```

```
exitmenu = Menu(menubar, tearoff = 0)

menubar.add_cascade(label = "Exit", menu = exitmenu)

exitmenu.add_command(label = "Quit", command = window.quit)

window.mainloop() # Create an event loop
```

Scrollbars

A Scrollbar widget can be used to scroll the contents in a Text, Canvas, or Listbox widget vertically or horizontally.

Example One

```
from tkinter import * # Import all definitions from tkinter

class ScrollText:

    def __init__(self):

        window = Tk() # Create a window

        window.title("Scroll Text Demo") # Set title

        frame1 = Frame(window)

        frame1.pack()

        scrollbar = Scrollbar(frame1)

        scrollbar.pack(side = RIGHT, fill = Y)

        text = Text(frame1, width = 40, height = 10, wrap = WORD,yscrollcommand=scrollbar.set)

        text.pack()

        scrollbar.config(command = text.yview)

        window.mainloop() # Create an event loop
```

```
ScrollText() # Create GUI
```

Event Handling

Simply putting GUI elements will not give sense until we write event handling code for them.

Let's demonstrate from simple to more complex

Example Code One

```
#simple GUI with class of mouse button click event handling

from tkinter import * # Import all definitions from tkinter

class ProcessButtonEvent:

    def __init__(self):

        window = Tk() # Create a window

        btOK = Button(window, text= "OK", fg = "red", command = self.processOK)

        btCancel = Button(window, text="Cancel",bg="yellow", command = self.processCancel)

        btOK.pack() # Place the OK button in the window

        btCancel.pack() # Place the Cancel button in the window

        window.mainloop()  # Create an event loop

    def processOK(self):

        print("OK button is clicked")

    def processCancel(self):

       print("Cancel button is clicked")
```

ProcessButtonEvent() # Create an object to invoke _ _init_ _ method

Example Code Two

```
#simple GUI with class of mouse button click event handling

from tkinter import * # Import all definitions from tkinter

class ProcessButtonEvent:

  def __init__(self):

      window = Tk() # Create a window

      btinfo = Button(window, text= "Information", command = self.info)

      btOK = Button(window, text= "Warrning", command = self.warninig)

      btCancel = Button(window, text="Error",command = self.err)

      btyn = Button(window, text="Yes/No",command = self.yesno)

      btoc= Button(window, text="Ok/Cancel",command = self.okcan)

      btinfo.pack()

      btOK.pack() # Place the OK button in the window

      btCancel.pack() # Place the Cancel button in the window

      btyn.pack()

      btoc.pack()

      window.mainloop()  # Create an event loop

  def info(self):

      messagebox.showinfo("showinfo", "This is an info msg")
```

```
    def warninig(self):
        messagebox.showwarning("showwarning", "This is a warning")
    def err(self):
        messagebox.showerror("showerror", "This is an error")
    def yesno(self):
        messagebox.askyesno("askyesno", "Continue?")
    def okcan(self):
        messagebox.askokcancel("askokcancel", "OK?")
ProcessButtonEvent() # Create an object to invoke _ _init_ _ method
```

Example Code Three

```
#dialogbox demo
from tkinter import *
def start():
    name = simpledialog.askstring("askstring", "Enter your name")
    age = simpledialog.askinteger("askinteger", "Enter your age")
    messagebox.showinfo("showinfo","Name:"+str(name)+"\n"+"Age: "+str(age))
window=Tk()
bt= Button(window, text="Start",command = start)
bt.pack()
window.mainloop()
```

Example Code Four

```
#open and save file dialog

from tkinter import *

root=Tk()

def opend():

root.filename=filedialog.askopenfilename(filetypes=(("howmm","*.hc"),("Allfiles","*.*")))

  print(root.filename)

def saved():

  root.filename=filedialog.asksaveasfile(mode='w',defaultextension=".txt")


button1 = Button(root, text = "Open File Dialog" ,command=opend)  # Create a button

button2 = Button(root, text = "Save File Dialog",command=saved)  # Create a button

button1.pack()  # Place the button in the window

button2.pack()  # Place the button in the window

root.mainloop()  # Create an event loop
```

Example Code Five

```
#simple calculator

from tkinter import *

def add():

   e3.insert(10,str(int(e1.get())+int(e2.get())))

def sub():

   e3.insert(10,str(int(e1.get())-int(e2.get())))

def div():

   e3.insert(10,str(int(e1.get())//int(e2.get())))

def mul():

   e3.insert(10,str(int(e1.get())*int(e2.get())))

def clr():

   e1.delete (0, END)

   e2.delete (0, END)

   e3.delete (0, END)

master = Tk()

master.minsize(400, 400)

Label(master, text="First Number").grid(row=0)

Label(master, text="Second Number").grid(row=1)
```

```
Label(master, text="The Result").grid(row=2)


e1 = Entry(master)

e2 = Entry(master)

e3 = Entry(master)


e1.grid(row=0, column=1)

e2.grid(row=1, column=1)

e3.grid(row=2, column=1)


Button(master, text='Add', command=add).grid(row=3, column=0)

Button(master, text='Subtract', command=sub).grid(row=3, column=1)

Button(master, text='Divide', command=div).grid(row=3, column=2)

Button(master, text='Multiplay', command=mul).grid(row=3, column=3)

Button(master, text='Clear', command=clr).grid(row=3, column=4)

mainloop( )
```

Exercise

1. Develop simple calculator

2. Develop simple text editor