Introduction

Python provides various options for developing GUIs. One such is Tkinter. It is Python interface to the Tk GUI toolkit which is available in Python. Other such options are wxPython, PyQt, JPython, etc.

* Standard GUI library for Python.
* Provides a fast and easy way to create GUI applications.

Steps to create GUI Application using Tkinter:

1. Importing the module
2. Create GUI main window
3. Adding widgets
4. Enter main event loop to take actions

Creating a window

import tkinter

from tkinter import \*

win = Tk()

win.geometry("200x200")

win.mainloop()

* geometry function is used to give dimensions for the window size.

Buttons

* A type of widget

import tkinter

from tkinter import \*

win = Tk()

win.geometry("200x200")

b=Button(win, text="button")

b.pack()

win.mainloop()

* Button() is used to create a button.
* The function should have the parameter of master in order to specify in which window it should be displayed.
* The text attribute is used to display what we need to show on the button
* Command attribute is used to make the button carry out any function of our choice.

Widget Placing Techniques

A widget can be placed in 3 different ways:

b.pack()

pack() function places the widget just as it is in the code. You cannot modify its position or alignment.

b.grid(row=5, column=6)

grid() function places the widget in the specified row and column. This method considers the whole window as rows and columns and thus the widget is placed in the cell formed.

b.place(x=200,y=600)

place() function places the widget in the specified pixels. This is the most accurate method to place a widget since it places the widget according to x and y axes pixels.

Canvas

* Rectangular area to draw shapes and figures

win = Tk()

win.geometry("200x200")

c=Canvas(win, height=200, width=650, background='red')

c.pack()

* Canvas() function is used to create a new canvas in the master window.
* Height and width are to be specified.
* Background/bg attribute helps in giving background colour to the canvas.

Standard Items supported in Canvas:

1. Arc

coord = 10, 50, 210, 240 *#coordinates of axes*

arc = c.create\_arc(coord, start=0, extent=120, fill='green')

- The create\_arc() function creates the arc within the given coordinates.

- Start and extent are the angles to be specified for the circular section.

1. Image

filename = PhotoImage(file = "35-x-45.jpg")

image = c.create\_image(50, 50, anchor=NE, image=filename)

1. Line

line = c.create\_line(10,20,300,400,fill='black')

* Create\_line() is used to create a line.
* The coordinates are to be provided

1. Polygon

pol = c.create\_polygon(5,8,300,30,190,190,230,410,fill='yellow')

1. Oval

oval= c.create\_oval(50,60,150,280,fill='brown')

Check button and radio button

g=IntVar()

p=IntVar()

m=IntVar()

check1 = Checkbutton(win, text='Games', offvalue=0, onvalue=1, height=2, width=4, variable=g)

check1.pack()

check2= Checkbutton(win, text='Painting', offvalue=0, onvalue=1, height=2, width=5, variable=p)

check2.pack()

check3= Checkbutton(win, text='Music', offvalue=0, onvalue=1, height=2, width=4, variable=m)

check3.pack()

* Checkbuton() function is used to create check buttons.
* Offvalue and onvalue can be 0 or 1. They are to be specified.
* Variables are used to store the value the user gives through the checkbutton.

g=IntVar()

radio1 = Radiobutton(win, text='Games', variable=g, value=1)

radio1.pack()

radio2 = Radiobutton(win, text='Play', variable=g, value=2)

radio2.pack()

radio3 = Radiobutton(win, text='Music', variable=g, value=3)

radio3.pack()

* Radiobutton() is used to create radio buttons.
* Since one will be able to select only some restricted values (mostly one option), there is only need of one variable to store the value. Values differ according to the radio button.
* Providing same value to all the radio buttons allow you to select all the radio buttons with the same value even if we select one.

Label

fn = Label(win, text="Enter first number ")

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

* Label() function is used to create labels/texts to show up on the window.

Entry

x1 = StringVar()

x2 = StringVar()

ef = Entry(win, textvariable=x1)

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

es = Entry(win, textvariable=x2)

es.grid(row=2,column=2)

* Entry() function is used to create a space to enter the input.
* Variables are given through textvariable.

Calculator

import tkinter

from tkinter import \*

from functools import partial

def sums(sm, x1,x2):

    num1 = x1.get()

    num2 = x2.get()

    add = int(num1) + int(num2)

    sm.config(text='The sum is %d' %add)

    return

def subtract(dm, x1,x2):

    num1 = x1.get()

    num2 = x2.get()

    subtract = int(num1) - int(num2)

    dm.config(text='Difference of numbers is %d' %subtract)

    return

win = Tk()

win.geometry("200x200")

fn = Label(win, text="Enter first number ")

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

sn = Label(win, text="Enter second number ")

sn.grid(row=2,column=0)

sm = Label(win)

sm.grid(row=6,column=0)

dm = Label(win)

dm.grid(row=7,column=0)

x1 = StringVar()

x2 = StringVar()

sums = partial(sums, sm, x1, x2)

subtract = partial(subtract,dm,x1,x2)

ef = Entry(win, textvariable=x1)

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

es = Entry(win, textvariable=x2)

es.grid(row=2,column=2)

button = Button(win, text='Add', command=sums)

button.grid(row=3, column=0)

button = Button(win, text='Subtract', command=subtract)

button.grid(row=4, column=0)

win.mainloop()

Frame

f1 = Frame(win)

f1.pack(side=TOP)

* Frame() is used to create frames. Frames are to be aligned using side attribute in pack.
* Any widgets to be kept should be given the name of the frame as master.

Listbox

lb = Listbox(win)

lb.insert(1,'List')

lb.insert(2,'Tuple')

lb.insert(3,'Dictionary')

lb.pack()

* Listbox() creates a list of items.
* Insert() helps to insert items into the listbox.

Toplevel

win = Tk()

win.title('one')

top = Toplevel()

top.title('two')

* Toplevel widgets work as windows that are directly managed by the window manager
* They do not necessarily have a parent widget on top of them
* Your application can use any number of top-level windows
* Title() is used to give a title to the window.

Message box

from tkinter import messagebox

win = Tk()

def hello():

    messagebox.showinfo("Tk","Welcome!")

b=Button(win, text='Click Me!', command=hello)

b.pack()

Menu and menu buttons

mb = Menubutton(win, text='File')

mb.grid()

mb.menu = Menu(mb)

mb['menu'] = mb.menu

x1 = IntVar()

x2 = IntVar()

mb.menu.add\_checkbutton(label='open', variable=x1)

mb.menu.add\_checkbutton(label='close', variable=x2)

mb.pack()

* Menu button is the part of a drop-down menu that stays on the screen all the time.
* Associated with a menu widget that can display the choices for that menubutton when the user clicks on it.
* To associate the menu button with a set of choices, set this option to the Menu object containing those choices.
* Menu() is to allow to create all kinds of menus that can be used by our applications.

mb = Menu(win)

fm = Menu(mb)

fm.add\_command(label='Close', command=nothing)

fm.add\_separator()

fm.add\_command(label='Exit', command=win.quit)

mb.add\_cascade(label="File",menu=fm)

* Add\_command() adds a menu command to the menu.
* Add\_separator() adds separator in between commands.
* Add\_cascade() adds the menu to the menu bar.

Scale, spin box and scroll

sc = Scale(win)

sc.pack()

* Scale sets a horizontal bar from 0 to 100 on the window.

sb = Spinbox(win, from\_=1900 ,to= 2100)

sb.pack()

* Spinbox() creates a list of values in the defined limits where a user can select a particular value by increasing or decreasing within the specified limits.

sb = Scrollbar(win)

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

* Scrollbar() creates a scroll bar according to the number of items on the window.

Paned Window

pw = PanedWindow()

pw.pack(fill=BOTH, expand=1)

lt = Entry(pw, bd=5)

pw.add(lt)

pw2 = PanedWindow(pw, orient=VERTICAL)

pw.add(pw2)

top = Scale(pw2, orient=HORIZONTAL)

pw2.add(top)

bn = Button(pw2, text='HELLO')

pw2.add(bn)

mainloop()

* PanedWindow() allows to create paned windows (windows with partition).
* Orient helps to fix the direction of the paned windows.
* Each widget in a paned window should be added to it using add().