**Name: Session:**

**Programming I**

**Lab Exercise 12/18/2019**

**Building GUI’s with TKinter**

In this activity we will be building graphical user interfaces using TKinter. The Tkinter module (“Tk interface”) is the standard Python interface to the Tk GUI toolkit from [Scriptics](http://www.scriptics.com/) (formerly developed by Sun Labs).

We will start with creating a simple GUI application that has two labels, two entry boxes, and a checkbutton (Tkinter version of a checkbox).

1. Open a new IDLE window and type in the following code and run it.

from tkinter import \*

root = Tk()

root.mainloop()

1. That is pretty boring so let’s put something in the window. Open a new IDLE window and type in the following code.

from tkinter import \*

root = Tk()

label\_1 = Label(root, text = "Name")

label\_2 = Label(root, text = "Password")

entry\_1 = Entry(root)

entry\_2 = Entry(root)

label\_1.grid(row = 0, sticky = E)

label\_2.grid(row = 1, sticky = E)

entry\_1.grid(row = 0, column = 1)

entry\_2.grid(row = 1, column = 1)

cb = Checkbutton(root, text = "Keep me logged in")

cb.grid(columnspan = 2)

root.mainloop()

Notice the following:

1. You must import Tkinter
2. Your base window is called root. Root is an object that is an empty window.
3. label\_1, label\_2, entry\_1, and cb all go into root
4. All objects use grid layout.
5. Now let’s see how we can put a button in the window. Enter and run the following code:

from tkinter import \*

root = Tk()

def printName():

print ("My name is Joe")

button\_1 = Button(root, text = "Print my name", command = printName)

button\_1.pack()

root.mainloop()

Note that pack() is a function that puts things in your window. If you do not give it any parameters, it will just pack the button in where it can fit. The button size is determined by the text that must fit in it. Later on we will see how to put images on your buttons. When the button is clicked, it will execute the event handler specified by the command parameter of the Button constructor.

1. Now let’s see another method of invoking the event handler known as “binding” a function to an event. Open a new IDLE window, type in and run the following code:

from tkinter import \*

root = Tk()

def printName(event):

print ("My name is Joe")

button\_1 = Button(root, text = "Print my name")

button\_1.bind("<Button-1>", printName)

button\_1.pack()

root.mainloop()

Note that the bind function has two parameters, “<Button-1>” and printName. The first is a reference to the left mouse button and the second is the event handler.

1. What if we just wanted to detect mouse clicks. One way we can do that is to create a surface that can detect them. For this we will use a Frame that will go on the window. Tkinter can detect when we click on a Frame just as a Button. Open a new IDLE window and enter and run the following code.

from tkinter import \*

root = Tk()

def leftClick(event):

print ("Left")

def middleClick(event):

print ("Middle")

def rightClick(event):

print ("Right")

frame = Frame(root, width = 300, height = 250)

frame.bind("<Button-1>", leftClick)

frame.bind("<Button-2>", middleClick)

frame.bind("<Button-3>", rightClick)

frame.pack()

root.mainloop()

Note that we will use all three mouse buttons to call three different event handlers.

1. Now let us see if we can get Tkinter to work using the object oriented paradigm. We will create a class called MyButtons that has a constructor and one method; printMessage. Open a new IDLE window and type and run the following code.

from tkinter import \*

#class definition

class MyButtons:

def \_\_init\_\_(self, master):

frame = Frame(master)

frame.pack()

self.printButton = Button(frame, text = "Print something", command = self.printMessage)

self.printButton.pack(side = LEFT)

self.quitButton = Button(frame, text = "Quit", command = master.destroy)

self.quitButton.pack(side = LEFT)

def printMessage(self):

print ("Wow this actually works")

#main program

root = Tk()

b = MyButtons(root)

root.mainloop()

Note: root is known to the class as “master”

Note: Quit button calls the destroy method of the window which is built-in

Note: Both buttons are packed LEFT. The second one (Quit) will pack in LEFT of what space is available.

1. Buttons are great but there are other things that we can put in a GUI. Now let’s put in a drop-down menu. Open a new IDLE window and type and run the following code.

from tkinter import \*

#define event handlers

def printOpen():

print ('Open ....')

def printClose():

print ('Close ....')

def printCopy():

print ('Copy ....')

def printPaste():

print ('Paste ....')

root = Tk()

#create the base menu object

mnu = Menu(root)

root.config(menu = mnu)

#create the File cascading menu

fileMenu = Menu(mnu)

mnu.add\_cascade(label = 'File', menu = fileMenu)

fileMenu.add\_command(label = 'Open', command = printOpen)

fileMenu.add\_command(label = 'Close', command = printClose)

fileMenu.add\_separator()

fileMenu.add\_command(label = 'Exit', command = root.destroy)

#create the Edit cascading menu

editMenu = Menu(mnu)

mnu.add\_cascade(label = 'Edit', menu = editMenu)

editMenu.add\_command(label = 'Copy', command = printCopy)

editMenu.add\_command(label = 'Paste', command = printPaste)

root.mainloop()

1. Now let’s add a Toolbar. Open a new IDLE window, type in and run the following code:

from tkinter import \*

#define event handlers

def printOpen():

print ('Open ....')

def printClose():

print ('Close ....')

def printCopy():

print ('Copy ....')

defprintPaste():

print ('Paste ....')

root = Tk()

#Create base menu

mnu = Menu(root)

root.config(menu = mnu)

#Create File cascading menu

fileMenu = Menu(mnu)

mnu.add\_cascade(label = 'File', menu = fileMenu)

fileMenu.add\_command(label = 'Open', command = printOpen)

fileMenu.add\_command(label = 'Close', command = printClose)

fileMenu.add\_separator()

fileMenu.add\_command(label = 'Exit', command = root.destroy)

#Create File cascading menu

editMenu = Menu(mnu)

mnu.add\_cascade(label = 'Edit', menu = editMenu)

editMenu.add\_command(label = 'Copy', command = printCopy)

editMenu.add\_command(label = 'Paste', command = printPaste)

#Create toolbar – buttons in a frame

toolbar = Frame(root, bg = 'blue')

button\_1 = Button(toolbar, text = 'Open', command = printOpen)

button\_1.pack(side = LEFT, padx = 2, pady = 2)

button\_2 = Button(toolbar, text = 'Close', command = printClose)

button\_2.pack(side = LEFT, padx = 2, pady = 2)

button\_3 = Button(toolbar, text = 'Copy', command = printCopy)

button\_3.pack(side = LEFT, padx = 2, pady = 2)

button\_4 = Button(toolbar, text = 'Paste', command = printPaste)

button\_4.pack(side = LEFT, padx = 2, pady = 2)

button\_5 = Button(toolbar, text = 'Exit', command = root.destroy)

button\_5.pack(side = LEFT, padx = 2, pady = 2)

toolbar.pack(side = TOP, fill = X)

root.mainloop()

1. Now let us place a status bar by adding the following code to the previous program:

#Status bar

status = Label(root, text = 'Preparing to do nothing', bd = 1, relief = SUNKEN, anchor = W)

status.pack(side = BOTTOM, fill = X)

1. In addition to placing widgets in our window, we can also place a canvas that we can draw on. Open a new IDLE window, enter and run the following code:

from tkinter import \*

root = Tk()

canvas = Canvas(root, width = 600, height = 400)

canvas.pack()

blackline = canvas.create\_line(0,0,600,200)

redline = canvas.create\_line(0,400, 600,200, fill = 'red')

greenbox = canvas.create\_rectangle(100, 100, 200, 300, fill = 'green')

#uncomment the following to delete things.

##canvas.delete(blackline)

##canvas.delete(redline)

##canvas.delete(ALL)

root.mainloop()

1. We can also place images in our window. I will show you two methods; the first one if you have GIF format and the second if you have PNG format. PNG is my personal preference as it supports more colors and better resolutions. GIF was invented by Compuserve and was one of the first web image formats. Open a new IDLE window, enter and run this code:

#Method 1 for GIF

from tkinter import \*

root = Tk()

photo = PhotoImage(file = "image1.gif")

label = Label(root, image = photo)

label.pack()

root.mainloop()

Note: If you wish to use other file formats, you will need to install Pillow with includes the Python Imaging Library (PIL)

#Method 2 for PNG

from tkinter import \*

from PIL import ImageTk

root = Tk()

photo = ImageTk.PhotoImage(file = "image1.png")

label = Label(root, image = photo)

label.pack()

root.mainloop()

1. Now that we can add images to our Tkinter application, let us redo our Toolbar using images (This uses method 1 for GIF but you are welcome to try method 2 for PNG:

from tkinter import \*

#event handlers

def printOpen():

print ('Open ....')

def printClose():

print ('Close ....')

def printCopy():

print ('Copy ....')

def printPaste():

print ('Paste ....')

#Create menus

root = Tk()

#Create icons for buttons

icon1 = PhotoImage(file = "image1.gif")

icon2 = PhotoImage(file = "image2.gif")

icon3 = PhotoImage(file = "image3.gif")

icon4 = PhotoImage(file = "image4.gif")

icon5 = PhotoImage(file = "image5.gif")

#Create a base level menu

mnu = Menu(root)

root.config(menu = mnu)

#Add a File cascading menu

fileMenu = Menu(mnu)

mnu.add\_cascade(label = 'File', menu = fileMenu)

fileMenu.add\_command(label = 'Open', command = printOpen)

fileMenu.add\_command(label = 'Close', command = printClose)

fileMenu.add\_separator()

fileMenu.add\_command(label = 'Exit', command = root.destroy)

#Add an Edit cascading menu

editMenu = Menu(mnu)

mnu.add\_cascade(label = 'Edit', menu = editMenu)

editMenu.add\_command(label = 'Copy', command = printCopy)

editMenu.add\_command(label = 'Paste', command = printPaste)

#Create toolbar with buttons with images

toolbar = Frame(root, bg = 'blue')

button\_1 = Button(toolbar, image=icon1, command = printOpen)

button\_1.pack(side = LEFT, padx = 2, pady = 2)

button\_2 = Button(toolbar, image=icon2, command = printClose)

button\_2.pack(side = LEFT, padx = 2, pady = 2)

button\_3 = Button(toolbar, image=icon3, command = printCopy)

button\_3.pack(side = LEFT, padx = 2, pady = 2)

button\_4 = Button(toolbar, image=icon4, command = printPaste)

button\_4.pack(side = LEFT, padx = 2, pady = 2)

button\_5 = Button(toolbar, image=icon5, command = root.destroy)

button\_5.pack(side = LEFT, padx = 2, pady = 2)

toolbar.pack(side = TOP, fill = X)

#Status bar

status = Label(root,text = 'Preparing to do nothing', bd = 1, relief = SUNKEN, anchor = W)

status.pack(side = BOTTOM, fill = X)

root.mainloop()

1. Now let’s go back to where we started. We will create a functional login screen. Create a new IDLE window, enter and run the following code:

from tkinter import \*

passwords = {'bgates':'money', 'jdoe':'password', 'jadams':'second', 'gwashington':'first'}

def validate():

username = entry\_1.get()

password = entry\_2.get()

if password == passwords[username]:

print ('Login successful')

else:

print ('Access denied')

root = Tk()

label\_1 = Label(root, text = "Name")

label\_2 = Label(root, text = "Password")

entry\_1 = Entry(root)

entry\_2 = Entry(root)

button\_1 = Button(root, text = "Login", command = validate)

label\_1.grid(row = 0, sticky = E)

label\_2.grid(row = 1, sticky = E)

entry\_1.grid(row = 0, column = 1)

entry\_2.grid(row = 1, column = 1)

button\_1.grid(row = 2, column = 2)

c = Checkbutton(root, text = "Keep me logged in")

c.grid(columnspan = 2)

root.mainloop()

1. Your turn. Modify the login screen with two major improvements. Your program should not allow the user to see the password being entered (Hint: the Entry constructor has a show parameter). You program should not crash if you do not have a valid user.

Hint: passwords.keys() returns a list of all keys (user names) in the passwords dictionary

1. Your turn. Create a Tkinter application that will allow the user to enter 2 numbers; start and end and prints to the console the sum of the cubes of those numbers. You will need two labels, two entry boxes and a button.

**Print out the last two programs source code, attach to this assignment and turn in.**