Practical Learning #9-1
Python GUI with events

Building Simple Calculator

Create a python file name PL-9_LastnameFirstname.

Import everything from tkinter

```
from tkinter import *
```

Here, we are creating our class, Window, and inheriting from the Frame class. Frame is a class from the tkinter module.

Then we define the settings upon initialization. This is the master widget.

```
class Window(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
```

The above is really all we need to do to get a window instance started.

Let's create the root window.

```
root = Tk()
```

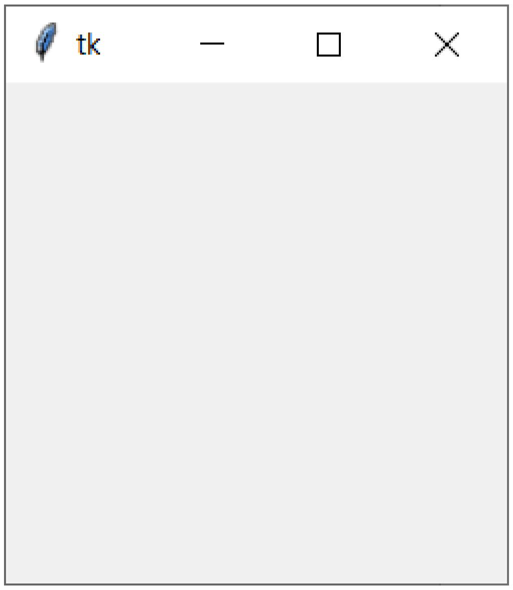
Then, create an instance.

```
app = Window(root)
```

Show it and begin the mainloop.

```
root.mainloop()
```

Output:



Adding Label, Entry and buttons

Here's our new code:

```
from tkinter import *

class Window(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
        self.init_window()

    #Creation of init_window
    def init_window(self):
        #changing the title of our master widget
        self.master.title("GUI")


        #creating label and entry
        lblFirstnum = Label(self,text="First Number: ")
        lblSecnum = Label(self,text="Second Number:")
        txtFirst = Entry(self,width=20)
        txtSec = Entry(self,width=20)

        #creating a button instance
        btnAdd = Button(self,text="ADD",width=7)
        btnQuit = Button(self,text="QUIT",width=7)

        #organize widget and place then on the window
        lblFirstnum.grid(row=0,column=0,pady=3)
        txtFirst.grid(row=0,column=1,columnspan=2)
        lblSecnum.grid(row=1,column=0,pady=3)
        txtSec.grid(row=1,column=1,columnspan=2)
        btnAdd.grid(row=2,column=1,pady=3)
        btnQuit.grid(row=2,column=2)

        #place the widget on the root window
        self.pack()
root = Tk()

#size of the window
root.geometry("300x90")
app = Window(root)
root.mainloop()
```

The ***self.init_window()*** – This is where we specify rules to our window. In our example, we give the window a title, which adds the title "GUI".

We add other widgets such as label, entry and buttons.

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

Entry – The Entry widget is used to display a single-line text field for accepting values from a user.

three main operations that you can perform with Entry widgets:

1. **Retrieving text** with .get()
2. **Deleting text** with .delete()
3. **Inserting text** with .insert()

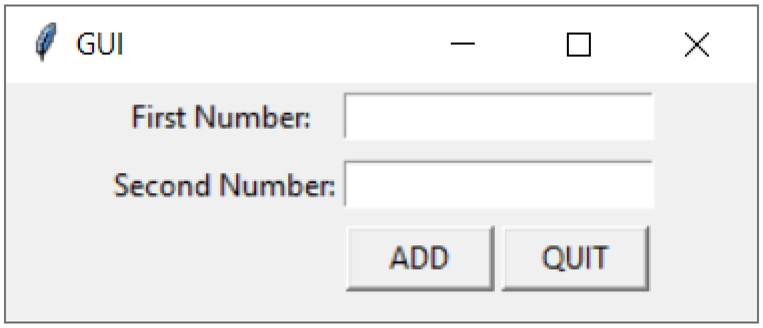
Button – A button that can contain text and can perform an action when clicked

We organize the widgets using the ***grid()*** method.

Then we place the widget on the root window using the ***place()*** method.

Outside our class, we also change the size our window to 300x90 using the ***geometry()*** manager.

Output:



Making your Application Interactive

When you create a Tkinter application, you must call window.mainloop() to start the **event loop**. During the event loop, your application checks if an event has occurred. If so, then some code can be executed in response.

The event loop is provided for you with Tkinter, so you don't have to write any code that checks for events yourself. However, you do have to write the code that will be executed in response to an event. In Tkinter, you write functions called **event handlers** for the events that you use in your application.

Note: An **event** is any action that occurs during the event loop that might trigger some behavior in the application, such as when a key or mouse button is pressed.

When an event occurs, an **event object** is emitted, which means that an instance of a class representing the event is instantiated. You don't need to worry about creating these classes yourself. Tkinter will create instances of event classes for you automatically.

Using command

Every Button widget has a command attribute that you can assign to a function. Whenever the button is pressed, the function is executed.

Here's our new code:

```
from tkinter import *

class Window(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
        self.init_window()

    #Creation of init_window
    def init_window(self):
        #changing the title of our master widget
        self.master.title("GUI")

        #creating label and entry
        lblFirstnum = Label(self,text="First Number: ")
        lblSecnum = Label(self,text="Second Number:")
        self.lblResult = Label(self,relief=RIDGE,width=30)
        self.txtFirst = Entry(self,width=20)
        self.txtSec = Entry(self,width=20)

        #creating a button instance
        btnAdd = Button(self,text="ADD",width=7,command=self.getSum)
        btnQuit = Button(self,text="QUIT",width=7, command=self.client_exit)
```

Add a new label for result

Since, were going to answer these widgets in our method, we need to add the "self" variable.

Every button widget's command attribute is assigned to a function. Whenever the button is pressed, the function is executed.
btnAdd – getSum ***btnQuit*** – client_exit

```
        #organize widget and place then on the window
        lblFirstnum.grid(row=0,column=0,pady=3)
        self.txtFirst.grid(row=0,column=1,columnspan=2)
        lblSecnum.grid(row=1,column=0,pady=3)
        self.txtSec.grid(row=1,column=1,columnspan=2)
        btnAdd.grid(row=2,column=1,pady=3)
        btnQuit.grid(row=2,column=2)
        self.lblResult.grid(row=3,column=0,columnspan=3)
```

Add the lblResult on our window.

```
        #place the widget on the root window
        self.pack()

    def client_exit(self):
        exit()
```

Function for ***btnQuit***.

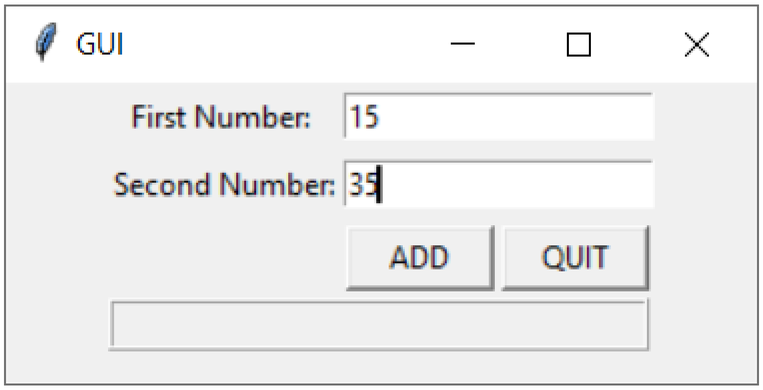
```
    def getSum(self):
        firstNum = self.txtFirst.get()
        secNum = self.txtSec.get()
        a = int(firstNum)+int(secNum)
        self.lblResult.config(text="The answer is {}".format(a))

root = Tk()
#size of the window
root.geometry("300x120")
app = Window(root)
root.mainloop()
```

Function for ***btnAdd***.
The ***.get()*** function was used to get the data from the Entry widgets.

Comput for the sum store to variable a.

.config() method was used to update/configure the text value or the label.

Output:



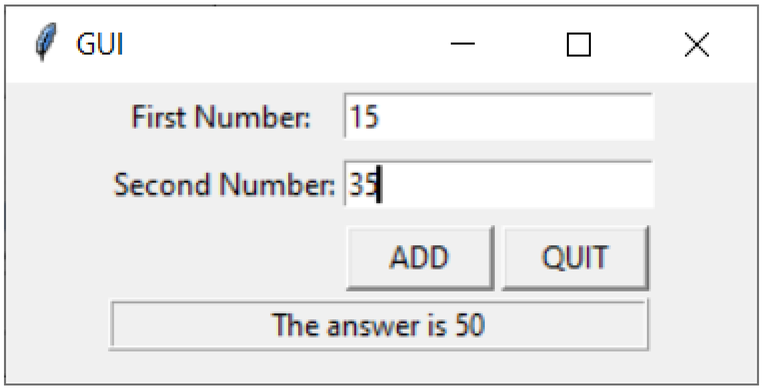
GUI

First Number: 15

Second Number: 35

ADD QUIT

When button ADD is pressed:



GUI

First Number: 15

Second Number: 35

ADD QUIT

The answer is 50