PREV
Lesson 36. Testing and debugging your programs

NEXT
Lesson 38. Capstone project: game of tag

Lesson 37. A library for graphical user interfaces

After reading lesson 37, you'll be able to

- Describe a graphical user interface
- Use a library for a graphical user interface to write a program

Every program you've written has interacted with the user in a text-based way. You've been displaying text on the screen and getting text input from the user. Although you can write interesting programs this way, the user is missing a more visual experience.

Consider this

Think about a program you use in your everyday life: a browser to go on the internet. What kinds of interactions do you have with the browser?

Answer: You open the browser window, click buttons, scroll, select text, and close the browser window.

Many programming languages have libraries that help programmers write visual applications. These applications use familiar interfaces between the user and the program: buttons, text boxes, drawing canvases, icons, and many more.

37.1. A LIBRARY FOR GRAPHICAL USER INTERFACES

A library for a graphical user interface (GUI) is a set of classes and methods that know how to interface between the user and the operating system to display graphical control elements, called *widgets*. Widgets are meant to enhance the user experience through interaction: buttons, scrollbars, menus, windows, drawing canvases, progress bars, or dialog boxes are a few examples.

Python comes with a standard library for GUIs, called tkinter, whose documentation is available at https://docs.python.org/3.5/library/tkinter.html#module-tkinter.

Developing a GUI application usually requires three steps. Each step is demonstrated in this lesson. As with any other program, development also involves setting up unit tests and debugging. The three steps are as follows:

- Setting up the window by determining its size, location, and title.
- Adding widgets, which are interactive "things" like buttons or menus, among many others.
- Choosing behaviors for widgets to handle events such as clicking a button or selecting a menu item. Behaviors are implemented by writing event handlers in the form of functions that tell the program which actions to take when the user interacts with the specific widget.

Quick check 37.1

What are some actions that you could do on each of the following?

1

A button

2

A scrollbar

3

4

A canvas

37.2. SETTING UP A PROGRAM USING THE TKINTER LIBRARY

All GUI programs typically run in a window. The window can be customized by changing its title, size, and background color. The following listing shows how to create a window of size 800 x 200 pixels, with a title of “My first GUI” and a background color of gray.

Listing 37.1. Creating a window

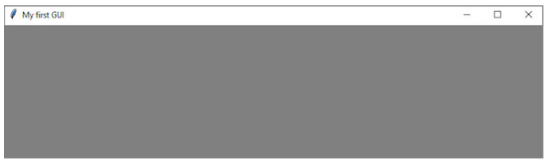
```
import tkinter                                   1

window = tkinter.Tk()                            2
window.geometry("800x200")                       3
window.title("My first GUI")                     4
window.configure(background="grey")              5
window.mainloop()                                6
```

- ***1*** **Imports the tkinter library**
- ***2*** **Creates a new object and binds it to a variable named window**
- ***3*** **Changes the size of the window**
- ***4*** **Adds a title to the window**
- ***5*** **Changes the background color of the window**
- ***6*** **Starts the program**

After you run this program, a new window will show up on your computer screen. If you don’t see it, it might be hiding behind another window you have open, so look on your taskbar for a new icon. You can terminate your program by closing the window.

Figure 37.1 shows what the window looks like on the Windows operating system. The window may look different if you’re using Linux or a Mac operating system.

Figure 37.1. An 800 x 200 pixel window with the title “My first GUI” and a background color of gray



Quick check 37.2

Write a program for each of the following:

1

Makes a 500 x 200 window with the title “go go go” and the background color green

2

Makes a 100 x 900 window with the title “Tall One” and the background color red

3

Makes two 100 x 100 windows with no title, but one has the background color white and the other black

37.3. ADDING WIDGETS

A blank window isn’t interesting. The user has nothing to click! After you create the window, you can start adding widgets. You’ll create a program that has three buttons, one text box, one progress bar, and one label.

To add a widget, you need two lines of code: one to create the widget and one to put it on the window. The following code shows the two lines by creating one button and adding it to the window object from listing 37.1. The first line creates the button and binds it to a variable named `btn`. The second line adds it (with pack) to the window:

```
btn = tkinter.Button(window)
btn.pack()
```

The next listing shows you how to add three buttons, one text box, and one label, assuming that you've already created a window, as in listing 37.1.

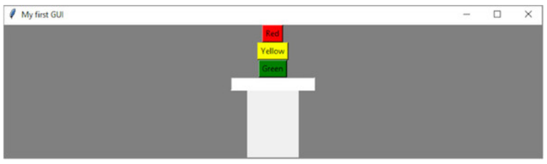
Listing 37.2. Adding widgets to the window

```
import tkinter
window = tkinter.Tk()
window.geometry("800x200")
window.title("My first GUI")
window.configure(background="grey")

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

yellow = tkinter.Button(window, text="Yellow", bg="ye
yellow.pack()

green = tkinter.Button(window, text="Green", bg="gree
green.pack()

textbox = tkinter.Entry(window)
textbox.pack()

colorlabel = tkinter.Label(window, height="10", width
colorlabel.pack()

window.mainloop()
```

- ***1*** **Creates a new button with a red background color and with the text "Red" on it**
- ***2*** **Adds the button with those properties to the window**
- ***3*** **Creates and adds a box in which you can enter text**
- ***4*** **Creates and adds a label whose height is 10**

When you run this program, you get a window that looks like the one in figure 37.2.

Figure 37.2. Window after adding three buttons, one text box, and one label. The top button is red and is labeled Red, the middle button is yellow and is labeled Yellow, and the bottom button is green and is labeled Green.



You can add many widgets to your programs. The ones that come with the standard tkinter library are listed in table 37.1.

Table 37.1. Widgets available in tkinter

Widget name	Description	Widget name	Description
Button	Shows a button	Menubutton	Shows menus
Canvas	Draws shapes	OptionMenu	Shows a pop-up menu
Checkbutton	Shows options via checkboxes (can select more than one option)	PanedWindow	Contains window panes that can be resized
Entry	A text field the user can type text in	Radiobutton	Shows options via radio buttons (can select only one option)
Frame	Container to put other widgets in	Scale	Shows a slider
Label	Shows single line of text or an image	Scrollbar	Adds scrollbars to other widgets
LabelFrame	Container to add space	Spinbox	Like an Entry, but can select only from some text
Listbox	Shows options via a list	Text	Shows multiple lines of text
Menu	Shows commands (contained inside Menubutton)	TopLevel	Allows for a separate window container

Write a line that creates each of the following:

- An orange button with the text Click here
- Two radio buttons
- A checkbutton

37.4. ADDING EVENT HANDLERS

At this point, you've created GUI windows and added widgets to them. The last step is to write code that tells the program what to do when a user interacts with the widgets. The code must somehow link the widget to the action.

When you create a widget, you give it the name of a command you want to run. The command is a function in the same program. The following listing shows an example of a code snippet that changes a window's background color when a button widget is clicked.

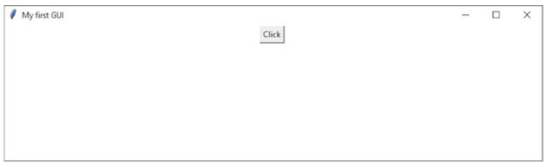
Listing 37.3. Event handler for a button click

```
import tkinter

def change_color():
    window.configure(background="white")

window = tkinter.Tk()
window.geometry("800x200")
window.title("My first GUI")
window.configure(background="grey")

white = tkinter.Button(window, text="Click", command=
white.pack()

window.mainloop()
```

- **1 Function representing the event to happen**
- **2 The function changes the background color of the window.**
- **3 Button with an associated action, by assigning the function name to the command parameter**

Figure 37.3 shows what the screen looks like after the button is clicked. The window is originally gray, but after you click the button, it's white. Clicking the button again doesn't change the color back to gray; it stays white.

Figure 37.3. The window background color changes to white from gray after clicking the button.



You can do more interesting things with event handlers. You can apply everything you've learned so far in the book when you write GUIs. For the final example, you'll write code for a countdown timer. You'll see how to read information from other widgets, use loops in event handlers, and even use another library.

Listing 37.4 shows a program that reads a number the user enters in a text box and then displays a countdown from that number to 0, with the number changing every second. There are four widgets in the program:

- A label with instructions for the user
- A text box for the user to put in a number
- A button to start the countdown
- A label to show the changing numbers

The button is the only widget that has an event handler associated with it. The function for that event will do the following:

- Change the color of the label to white
- Get the number from the text box and convert it to an integer
- Use the number from the text box in a loop, starting from that value and ending at 0

The bulk of the work is being done inside the loop. It uses the loop variable, `i`, to change the text of the label. Notice that you're giving a variable name to the text parameter of the label, whose value changes every time through the loop. Then, it calls an update method to refresh the window and show the changes. Finally, it uses the `sleep` method from the `time` library to pause execution for one second. Without the `sleep` method, the countdown

Listing 37.4. Program that reads a text box and counts down that many seconds

```
import tkinter
import time

def countdown():
    countlabel.configure(background="white")
    howlong = int(textbox.get())
    for i in range(howlong,0,-1):
        countlabel.configure(text=i)
        window.update()
        time.sleep(1)
    countlabel.configure(text="DONE!")

window = tkinter.Tk()
window.geometry("800x600")
window.title("My first GUI")
window.configure(background="grey")

lbl = tkinter.Label(window, text="How many seconds to
lbl.pack()
textbox = tkinter.Entry(window)
textbox.pack()
count = tkinter.Button(window, text="Countdown!", com
count.pack()
countlabel = tkinter.Label(window, height="10", width
countlabel.pack()

window.mainloop()
```

- ***1*** **Event handler function**
- ***2*** **Changes the label's color to white**
- ***3*** **Gets the value from the text box and converts it to an int**
- ***4*** **Loops starting from the number in the text box until 0**
- ***5*** **Changes the text on the label to the value of the loop variable**
- ***6*** **Updates the window to show the updated value on the label**
- ***7*** **Waits one second using the time library**
- ***8*** **Changes the text of the label to done after reaching 0**
- ***9*** **One label with instructions for the user**
- ***10*** **Text box for the user to enter a number**
- ***11*** **Button to start the countdown, with the function written in the first line as the event command**
- ***12*** **A label on which to print the countdown**

With the ability to set up a window, add widgets, and create event handlers, you can write many visually appealing and uniquely interactive programs.

Quick check 37.4

Q1:

Write code that creates a button. When clicked, the button randomly chooses the color red, green, or blue, and changes the background color of the window to the chosen color.

SUMMARY

In this lesson, my goal was to teach you how to use a graphical user interface library. The library contains classes and methods that help programmers manipulate graphical elements of the operating system. Here are the major takeaways:

- Your GUI program works inside a window.
- In the window, you can add graphical elements, called widgets.
- You can add functions that perform tasks when a user interacts with the widget.

Let's see if you got this...

Q37.1

Write a program that stores names, phone numbers, and emails of people in a phonebook. The window should have three text boxes for the name, phone, and email. Then it should have one button to add a contact, and one button to show all contacts. Finally, it should have a label. When the user clicks the Add button, the program reads the text boxes and stores the information. When the user clicks the Show button, it

PREV
Lesson 36. Testing and debugging your programs

NEXT
Lesson 38. Capstone project: game of tag