Python Programming

- Presentation By Uplatz
- Contact us: https://training.uplatz.com
- Email: info@uplatz.com
- Phone: +44 7836 212635

GUI PROGRAMMING

Learning outcomes:

What is a *GUI PROGRAMMING*?

Tkinter Programming

Tkinter Widgets

Building Your First Python GUI program with *Tkinter*

What is a GUI PROGRAMMING?

A **graphical user interface** (GUI) allows a user to interact with a computer program using a pointing device that manipulates small pictures on a computer screen.
Python has a huge number of GUI frameworks (or toolkits) available for it, from TkInter (traditionally bundled with Python, using Tk) to a number of other cross-platform solutions, as well as bindings to platform-specific (also known as "native") technologies.

What is a GUI PROGRAMMING?

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below:

Tkinter: Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.

wxPython: This is an open-source Python interface for wxWindows **http://wxpython.org**.

JPython: JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine **http://www.jython.org**.

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter has several strengths. It’s **cross-platform**, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they’re run.

Tkinter Programming

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps:

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. We present these widgets as well as a brief description in the following table:

Operator	Description
Button	The Button widget is used to display buttons in your application.
Label	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

Tkinter Widgets

Operator	Description
Checkbutton	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
Entry	The Entry widget is used to display a single-line text field for accepting values from a user.
Canvas	The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
Frame	The Frame widget is used as a container widget to organize other widgets.
Listbox	The Listbox widget is used to provide a list of options to a user.

Tkinter Widgets

Operator	Description
Menu	The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
Menubutton	The Menubutton widget is used to display menus in your application.
Message	The Message widget is used to display multiline text fields for accepting values from a user.
Radiobutton	The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
Scale	The Scale widget is used to provide a slider widget.

Tkinter Widgets

Operator	Description
Spinbox	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
Text	The Text widget is used to display text in multiple lines.
Toplevel	The Toplevel widget is used to provide a separate window container .
LabelFrame	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

Building Your First Python GUI program with *Tkinter*

The foundational element of a Tkinter GUI is the **window**. Windows are the containers in which all other GUI elements live. These other GUI elements, such as text boxes, labels, and buttons, are known as **widgets**. Widgets are contained inside of windows.

First, create a window that contains a single widget. Start up a new Python shell session and follow along!

Building Your First Python GUI program with *Tkinter*

With your Python shell open, the first thing you need to do is import the Python GUI Tkinter module:

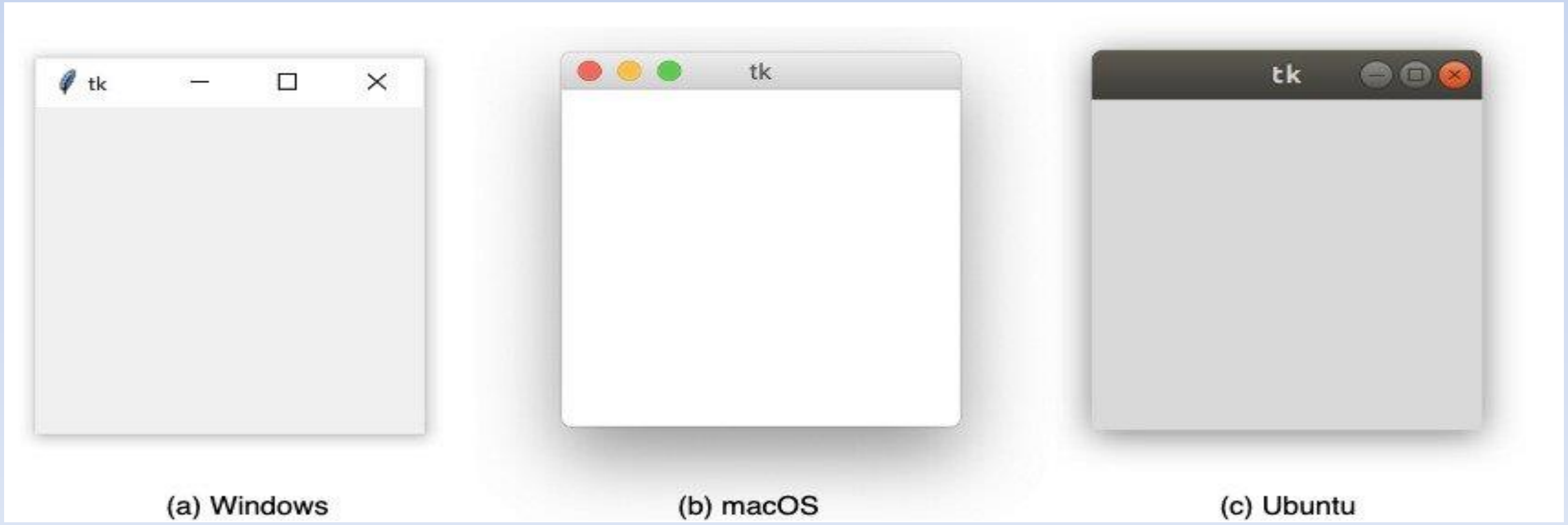
```
>>>import tkinter as tk
```

A **window** is an instance of Tkinter’s Tk class. Go ahead and create a new window and assign it to the variable window:

```
>>> window=tk.Tk()
```

Building Your First Python GUI program with *Tkinter*

When you execute the above code, a new window pops up on your screen. How it looks depends on your operating system:



(a) Windows (b) macOS (c) Ubuntu

Building Your First Python GUI program with *Tkinter*

Adding a Widget:

Now that you have a window, you can add a widget. Use the tk.Label class to add some text to a window. Create a Label widget with the text "Hello, Tkinter" and assign it to a variable called greeting:

```
>>> greeting = tk.Label(text="Hello, Tkinter")
```

Building Your First Python GUI program with *Tkinter*

Adding a Widget:

The window you created earlier doesn’t change. You just created a Label widget, but you haven’t added it to the window yet. There are several ways to add widgets to a window. Right now, you can use the Label widget’s .pack() method:

```
>>> greeting.pack()
```

Building Your First Python GUI program with *Tkinter*

Adding a Widget:

When you .pack() a widget into a window, Tkinter sizes the window as small as it can while still fully encompassing the widget. Now execute the following:

```
>>> window.mainloop()
```

window.mainloop() tells Python to run the Tkinter **event loop**. This method listens for events, such as button clicks or keypresses, and **blocks** any code that comes after it from running until the window it's called on is closed.

Building Your First Python GUI program with *Tkinter*

You can control Label text and background colors using the foreground and background parameters:

```
lbl = tk.Label( text="Hello, Tkinter",
foreground=“red", background=“blue"  )
```

You can also control the width and height of a label with the width and height parameters:

```
lbl = tk.Label( text="Hello, Tkinter", fg=“red",
bg=“blue“, width=10, height=10  )
```

Building Your First Python GUI program with *Tkinter*

Displaying Clickable Buttons With Button Widgets: **Button** widgets are used to display **clickable buttons**. They can be configured to call a function whenever they’re clicked.

For example, the following code creates a Button with a blue background and yellow text. It also sets the width and height to 25 and 5 text units, respectively:

```
button = tk.Button( text="Click me!", width=25,
height=5, bg="blue", fg="yellow", )
```

Building Your First Python GUI program with *Tkinter*

Getting User Input With Entry Widgets: When you need to get a little bit of text from a user, like a name or an email address, use an **Entry** widget. They display a **small text box** that the user can type some text into. Creating and styling an Entry widget works pretty much exactly like label and button widgets. For example, the following code creates a widget with a blue background, some yellow text, and a width of 50 text units:

```
entry = tk.Entry(fg="yellow", bg="blue", width=50)
```

Building Your First Python GUI program with *Tkinter*

Getting User Input With Entry Widgets: There are three main operations that you can perform with Entry widgets:

Retrieving text with .get()

Deleting text with .delete()

Inserting text with .insert()

The best way to get an understanding of Entry widgets is to create one and interact with it.

Building Your First Python GUI program with *Tkinter*

Getting User Input With Entry Widgets:

```
import tkinter as tk
window = tk.Tk()
label = tk.Label(text="Name")
entry = tk.Entry()
label.pack()
entry.pack()
```

Click inside the Entry widget with your mouse and type “Python“.

Building Your First Python GUI program with *Tkinter*

Getting User Input With Entry Widgets:

Now you’ve got some text entered into the Entry widget, but that text hasn’t been sent to your program yet. You can use .get() to retrieve the text and assign it to a variable called name:

```
>>> name = entry.get()
>>> name
'Python‘
```

You should write these codes in python shell

Building Your First Python GUI program with *Tkinter*

Getting User Input With Entry Widgets:

You can .delete() text as well. This method takes an integer argument that tells Python which character to remove. For example, the code block below shows how .delete(0) deletes the first character from the Entry:

```
>>> entry.delete(0)
```

The text remaining in the widget is now "ython“

```
>>> entry.delete(0,2)
```

The text remaining in the widget is now "hon“

Building Your First Python GUI program with *Tkinter*

Getting User Input With Entry Widgets:

You can also .insert() text into an Entry widget

```
>>> entry.insert(0, “Hello")
```

The first argument tells .insert() where to insert the text. If there is no text in the Entry, then the new text will always be inserted at the beginning of the widget, no matter what value you pass as the first argument.

Building Your First Python GUI program with *Tkinter*

Frame Widgets:

The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way . It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets.

```
import tkinter as tk
window = tk.Tk()
frame1 = tk.Frame(master=window, width=100, height=100, bg="red")
frame1.pack()
frame2 = tk.Frame(master=window, width=50, height=50, bg="yellow")
frame2.pack
window.mainloop()
```

Building Your First Python GUI program with *Tkinter*

The .place() Geometry Manager:

You can use **.place()** to **control the precise location** that a widget should occupy in a window or Frame.

```
import tkinter as tk
window = tk.Tk()
label1 = tk.Label(text="I'm at (0, 0)", bg="red")
label1.place(x=0, y=0)
label2 = tk.Label(text="I'm at (75, 75)", bg="yellow")
label2.place(x=75, y=75)
window.mainloop()
```

Building Your First Python GUI program with *Tkinter*

Canvas:

The Canvas is a rectangular area intended for drawing pictures or other complex layouts. You can place graphics, text, widgets or frames on a Canvas.

```
import tkinter as tk
top = tk.Tk()
C = tk.Canvas(top, bg="blue", height=250, width=300)
coord = 10, 50, 240, 210
arc = C.create_arc(coord, start=0, extent=150,
fill="red")
C.pack()
top.mainloop()
```

Uplatz

Building Your First Python GUI program with *Tkinter*

Listbox:

The Listbox widget is used to display a list of items from which a user can select a number of items.

```
import tkinter as tk
window = tk.Tk()
Lb1 = tk.Listbox(window)
Lb1.insert(1, "Python")
Lb1.insert(2, "Perl")
Lb1.insert(3, "C")
Lb1.pack()
window.mainloop()
```

Building Your First Python GUI program with *Tkinter*

Menu:

The Menu widget is used to create various types of menus (top level, pull down, and pop up) in the python application.

Let’s see the example.

Thank you

Uplatz