Tkinter Button

Summary: in this tutorial, you'll learn about the Tkinter Button widget and how to use it to create various kinds of buttons.

Introduction to Tkinter button widget

Button widgets represent a clickable item in the applications. Typically, you use a text or an image to display the action that will be performed when clicked.

Buttons can display text in a single font. However, the text can span multiple lines. On top of that, you can make one of the characters underline to mark a keyboard shortcut.

To invoke a function (https://www.pythontutorial.net/python-basics/python-functions/) or a method of a class (https://www.pythontutorial.net/python-oop/python-class/) automatically when the button is clicked, you assign its `command` option to the function or method. This is called the command binding in Tkinter (https://www.pythontutorial.net/tkinter/tkinter-command/) .

To create a button, you use the `ttk.Button` constructor as follows:

```
button = ttk.Button(container, **option)
```

A button has many options. However, the typical ones are like this:

```
button = ttk.Button(container, text, command)
```

In this syntax:

- The `container` is the parent component on which you place the button.
- The `text` is the label of the button.
- The `command` specifies a callback function that will be called automatically when the button clicked.

Command callback

The `command` option associates the button's action with a function or a method of a class. When you click or press the button, it'll automatically invoke a callback function.

To assign a callback to the `command` option, you can use a lambda expression:

```
def callback():
    # do something


ttk.Button(
   root,
   text="Demo Button",
   command=callback
)
```

If the function contains one expression, you use a lamba expression:

```
ttk.Button(
   root,
   text="Demo Button",
   command=lambda_expression
)
```

Button states

The default state of a button is `normal`. In the `normal` state, the button will respond to the mouse events and keyboard presses by invoking the callback function assigned to its command option.

The button can also have the `disabled` state. In the `disabled` state, a button is greyed out and doesn't respond to the mouse events and keyboard presses.

To control the state of a button, you use the `state()` method:

```
# set the disabled flag
button.state(['disabled'])

# remove the disabled flag
button.state(['!disabled'])
```

Tkinter button examples

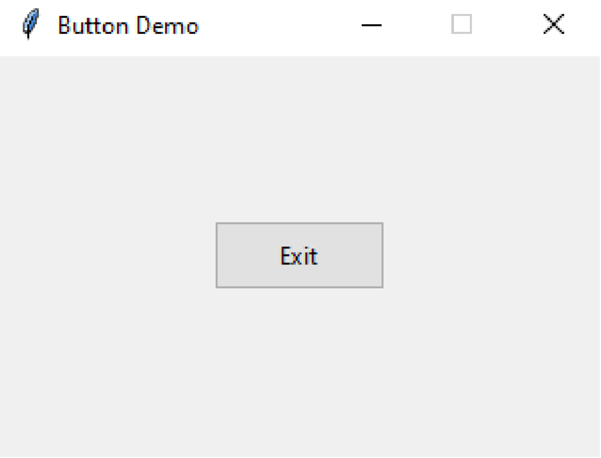
Let's take some examples of using button widgets.

1) Simple Tkinter button example

The following program shows how to display an `Exit` button. When you click it, the program is terminated.

```
import tkinter as tk
from tkinter import ttk

# root window
root = tk.Tk()
root.geometry('300x200')
root.resizable(False, False)
root.title('Button Demo')

# exit button
exit_button = ttk.Button(
    root,
    text='Exit',
    command=lambda: root.quit()
)

exit_button.pack(
    ipadx=5,
    ipady=5,
    expand=True
)

root.mainloop()
```

Output:



How it works.

The following creates the `Exit` button:

```
    exit_button = ttk.Button(
        root,
        text='Exit',
        command=lambda: root.quit()
    )
```

The command of the button is assigned to a lambda expression (https://www.pythontutorial.net/python-basics/python-lambda-expressions/) that closes the root window.

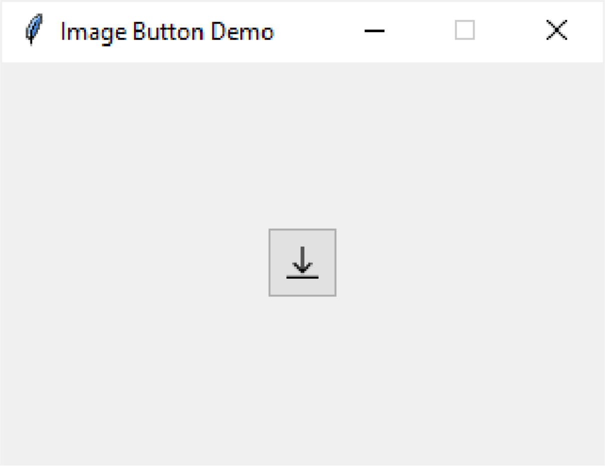
2) Tkinter image button example

The following program shows how to display an image button. To practice this example, you need to download the following image first:

Just right-click and save it into a folder that is accessible from the following program e.g., `assets` folder:

```
import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo


# root window
root = tk.Tk()
root.geometry('300x200')
root.resizable(False, False)
root.title('Image Button Demo')


# download button
def download_clicked():
    showinfo(
        title='Information',
        message='Download button clicked!'
    )


download_icon = tk.PhotoImage(file='./assets/download.png')
download_button = ttk.Button(
    root,
    image=download_icon,
    command=download_clicked
)

download_button.pack(
    ipadx=5,
    ipady=5,
    expand=True
)


root.mainloop()
```

Output:



How it works.

- First, create a new instance of the `tk.PhotoImage` class that references the image file `'./assets/download.png'` .
- Second, create the `ttk.Button` whose image option is assigned to the image.
- Third, assign a function to the `command` option. When you click the button, it'll call the `download_clicked` function that displays a message box.

3) Displaying an image button

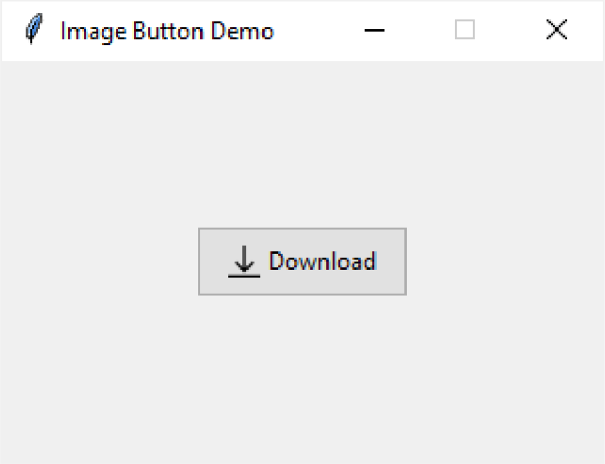
To display both text and image on a button, you need to use the `compound` option. If you don't, the button will display the text only, not the image.

The following shows how to display both text and image on a button:

```
import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo
```

```
# root window
root = tk.Tk()
root.geometry('300x200')
root.resizable(False, False)
root.title('Image Button Demo')


# download button handler
def download_clicked():
    showinfo(
        title='Information',
        message='Download button clicked!'
    )


download_icon = tk.PhotoImage(file='./assets/download.png')

download_button = ttk.Button(
    root,
    image=download_icon,
    text='Download',
    compound=tk.LEFT,
    command=download_clicked
)

download_button.pack(
    ipadx=5,
    ipady=5,
    expand=True
)


root.mainloop()
```

Output:



Summary

- Use the `ttk.Button()` class to create a button.
- Assign a lambda expression (https://www.pythontutorial.net/python-basics/python-lambda-expressions/) or a function (https://www.pythontutorial.net/python-basics/python-functions/) to the `command` option to respond to the button click event.
- Assign the `tk.PhotoImage()` to the `image` property to display an image on the button.
- Use the `compound` option if you want to display both text and image on a button.