P Python TUTORIAL

How to Change the Appearances of Widgets Dynamically Using Ttk Style map() Method

Summary: in this tutorial, you'll learn how to use the ttk Style `map()` method to dynamically change the appearance of a widget based on its specific state.

Typically, a ttk widget (https://www.pythontutorial.net/tkinter/tkinter-ttk/) allows you to change its appearance based on a specific state.

The following table shows a list of widget states and their meanings:

State	Meaning
`active`	The mouse is currently within the widget.
`alternate`	Ttk reserved this state for application use.
`background`	The widget is on a window that is not the foreground window. The foreground window is a window that is getting user inputs. This state is only relevant to Windows and macOS.
`disabled`	The widget won't respond to any actions.

State	Meaning
`focus`	The widget currently has focus.
`invalid`	The value of the widget is currently invalid.
`pressed`	The widget is currently being clicked or pressed e.g. when a Button widget is pressed.
`readonly`	The readonly widget prevents you from changing its current value e.g., a read-only `Entry` widget won't allow you to change its text contents.
`selected`	The widget is selected e.g. when radio buttons are checked.

To change the appearance of a widget dynamically, you can use the `map()` method of the Style object:

```
style.map(style_name, query)
```

The `map()` method accepts the first argument as the name of the style e.g., `TButton` and `TLabel` .

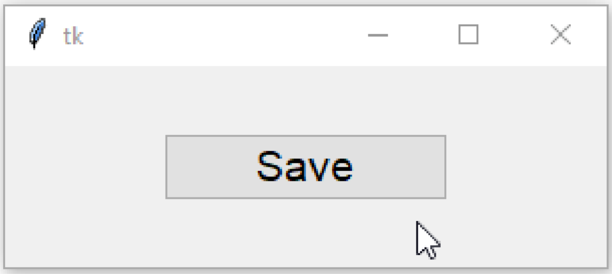
The argument query is a list of keyword arguments where each key is a style option and values are lists of tuples of `(state,value)` .

For example, the following code dynamically changes the foreground color of a button widget:

```
import tkinter as tk
from tkinter import ttk


class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.geometry('300x100')

        button = ttk.Button(self, text='Save')
        button.pack(expand=True)

        style = ttk.Style(self)
        style.configure('TButton', font=('Helvetica', 16))
        style.map('TButton',
                foreground=[('pressed', 'blue'),
                            ('active', 'red')])

        print(style.layout('TButton'))


if __name__ == "__main__":
    app = App()
    app.mainloop()
```

In this example, when you move focus to the button, its text color changes to red. And when you click or press the button, its text color turns to blue.

Output:



Summary

- Use the `style.map()` method to dynamically change the appearance of a widget based on its specific state.