Using Classes And Objects To Create GUIs | Python Tkinter GUI Tutorial In Hindi #25

Using Classes And Objects To Create GUIs | Python Tkinter GUI Tutorial In Hindi #25

Python is often treated purely as a scripting language, but it is fundamentally an OOP language. With OOP, the user states the program's structure, and the user's classes literally return "objects," which is why it is called "object" oriented. The objects serve as "instances" of your classes. For writing any lengthy GUI program, it's better to use OOP (classes and objects).

Code is described below:

```
from tkinter import *


class GUI(Tk):
    def __init__(self):
        super().__init__()
        self.geometry("744x377")

    def status(self):
        self.var = StringVar()
        self.var.set("Ready")
        self.statusbar = Label(self, textvar=self.var, relief=SUNKEN, anchor="w")
        self.statusbar.pack(side=BOTTOM, fill=X)

    def click(self):
        print("Button clicked")

    def createbutton(self, inptext):
        Button(text=inptext, command=self.click).pack()


if __name__ == '__main__':
    window = GUI()
    window.status()
    window.createbutton("Click me")
    window.mainloop()
```

- Importing *Tkinter* is the same as importing any other module in the Python code. Note that the module's name in Python 2.x is '*Tkinter,*' and in Python 3.x, it is '*tkinter*'.

```
from tkinter import *
```

- A class GUI(Tk) is created, which is inherited from Tk (Tkinter). We have to make a constructor within this class using 'def __init__', and the "self" option is used in this constructor rather than root.
- We have to call the superclass constructor within this constructor using super().__init__().
- We set the geometry of the GUI using the geometry() function. As in the example: the width is 744 pixels and height is 377 pixels, so we can write the function as *geometry(744x377).*

```
class GUI(Tk):
    def __init__(self):
        super().__init__()
        self.geometry("744x377")
```

- Another function is defined using 'def' ('status' is the function name), and self is passed within this function.
- A variable var is taken as StringVar(), and that is set to "Ready."
- A status bar is created using the Label() widget, and the var is taken as the textvar of the statusbar.
- The statusbar is packed using pack() method at the BOTTOM filled in the X dimension of the window.

```
def status(self):
        self.var = StringVar()
        self.var.set("Ready")
        self.statusbar = Label(self, textvar=self.var, relief=SUNKEN, anchor="w")
        self.statusbar.pack(side=BOTTOM, fill=X)
```

- A function click() is defined using 'def', which will print "Button clicked" on the terminal whenever the function will be called.

```
def click(self):
        print("Button clicked")
```

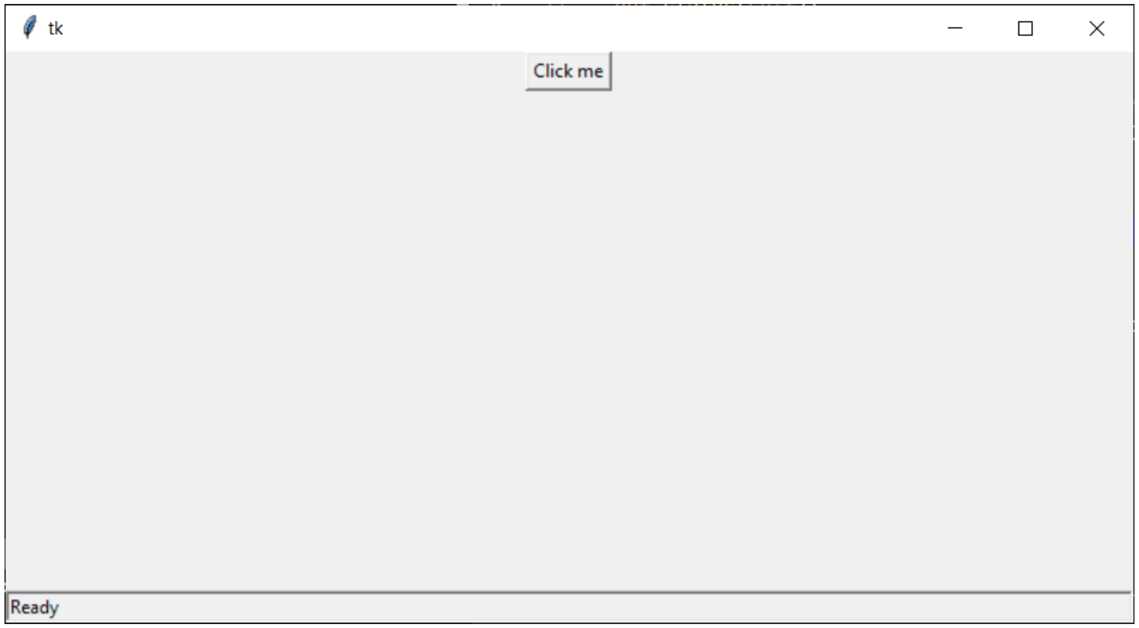
- We extend our little script by defining one more function, "createbutton." Within this function, a button is created using the Button () widget. We bind the Button to the *click() function*. So, every time this Button is clicked, the text "Button clicked" will be printed on the terminal from which we had called the script. Then we need to use the pack() method to pack it.

```
def createbutton(self, inptext):
        Button(text=inptext, command=self.click).pack()
```

- The main function is created as __name__== '__main__'. We have to make an object named "window" (equivalent to root) of the GUI() class. Similarly, we have to call other functions as well.
- There is a method known by the name *mainloop()*, which is used when your application is ready to run. This is an infinite loop used to run the application, wait for an event, and process the event as long as the window is not closed.

```
if __name__ == '__main__':
    window = GUI()
    window.status()
    window.createbutton("Click me")
    window.mainloop()
```

Output: The output of the code (or the GUI window) is given below:



Code as described/written in the video

Copy

```
from tkinter import *


class GUI(Tk):
    def __init__(self):
        super().__init__()
        self.geometry("744x377")

    def status(self):
        self.var = StringVar()
        self.var.set("Ready")
        self.statusbar = Label(self, textvar=self.var, relief=SUNKEN, anchor="w")
        self.statusbar.pack(side=BOTTOM, fill=X)

    def click(self):
        print("Button clicked")

    def createbutton(self, inptext):
        Button(text=inptext, command=self.click).pack()


if __name__ == '__main__':
    window = GUI()
    window.status()
    window.createbutton("Click me")
    window.mainloop()
```

Previous

Next

CodeWithHarry

Copyright © 2022 CodeWithHarry.com