Overview

This lesson explains what callbacks are in Tkinter and how they connect user actions to your code.

Important Information

What a Callback Is

A **callback** is a function that you write, but you do not call it directly. Instead, Tkinter calls it for you when something happens.

For example, when a button is clicked, Tkinter calls the function you attached to that button.

Why Callbacks Exist

In a GUI, your program spends most of its time waiting. The event loop waits for events, and when an event happens, it runs the correct callback.

This is different from a normal program where you decide exactly when each function runs. In a GUI, the user's actions help decide what code runs next.

A Button Command Callback

A button callback is set using the `command` option.

```
from tkinter import *
from tkinter import ttk

root = Tk()

def on_button_click():
    print("Button clicked!")

button = ttk.Button(root, text="Click Me", command=on_button_click)
button.grid(row=0, column=0, padx=10, pady=10)

root.mainloop()
```

When the user clicks the button, Tkinter creates a click event, the event loop processes it, and your callback runs.

Callbacks Must Be Fast

Callbacks run inside the event loop. If a callback takes too long, the event loop is blocked and the GUI freezes.

This is why long-running work should be broken into steps using `after`, or handled using a different approach that doesn't block the GUI.

A Callback That Schedules More Work

A powerful pattern is: do a small amount of work, then schedule the next step, then return immediately.

```
from tkinter import *

root = Tk()
count = 0

def count_up():
    global count
    count += 1
    print(count)

    if count < 10:
        root.after(200, count_up)  # schedule the next callback

root.after(200, count_up)
root.mainloop()
```

Here, the callback is short, so the event loop stays responsive.