Overview

This lesson introduces **tkinter**, Python's built-in library for creating graphical applications. You will learn what a **GUI** is, how humans interact with computers through GUIs, why both **tk** and **ttk** are used together in modern tkinter programs, and the three core pillars that every tkinter app is built on.

Important Information

What Is a GUI?

A **GUI (Graphical User Interface)** is a way for people to interact with a computer using visual elements instead of typed commands.

Examples of GUI elements include:

- Windows
- Buttons
- Text boxes
- Menus
- Sliders

Instead of typing instructions into a terminal, users **click**, **type**, and **drag**. This makes programs easier and more natural for humans to use.

GUIs and Human–Machine Interaction

Human–machine interaction is about how people communicate with computers. A GUI acts as a **bridge** between human thinking and computer logic:

- Humans think visually and spatially
- Computers process instructions step by step
- A GUI translates user actions (like clicking a button) into code the computer can understand

tkinter allows Python programs to create this bridge.

What Is tkinter?

tkinter is Python's standard GUI library. It lets you:

- Create windows
- Add buttons, labels, and input fields
- Respond to user actions like clicks and typing

tkinter is included with most Python installations, which makes it beginner-friendly.

tk vs ttk (and Why You Use Both)

tkinter is built on top of an older toolkit called **tk**. Over time, a newer system called **ttk (themed tk)** was added.

- **tk**
  - The original widget system
  - Simple and reliable
  - Some widgets look outdated
- **ttk**
  - Modern, themed widgets
  - Better appearance
  - Adapts to the operating system's look (Windows, macOS, Linux)

Why Import Both?

In most real tkinter programs, you import **tk** and **ttk** together:

- tk handles the main window and core functionality
- ttk provides modern widgets like buttons, labels, and input fields

Using both simplifies development because:

- You get access to **all widgets**
- Your app looks more modern
- You don't need to worry about which system does what

This is why you will often see both imported at the start of a tkinter program.

The 3 Core Pillars of tkinter Development

Every tkinter application is built on three fundamental ideas.

1. Widgets

Widgets are the building blocks of a GUI.

Common widgets include:

- Labels (display text)
- Buttons (trigger actions)
- Entry fields (user input)
- Checkboxes and dropdowns

If you can see it or interact with it, it is a widget.

2. Geometry (Layout)

Geometry controls **where widgets go** inside a window.

tkinter uses layout managers to:

- Position widgets
- Control spacing
- Adjust when the window resizes

Without geometry management, widgets would not appear in usable locations.

3. The Event Loop

The **event loop** is what keeps a GUI running.

It:

- Waits for user actions (clicks, typing, resizing)
- Responds when something happens
- Keeps the window open

Without the event loop:

- The window would appear and immediately close
- Buttons would not work
- The program would not respond to the user

All interactive GUI programs rely on an event loop.

Set Up

Installing tkinter, tk, and ttk

In most standard Python installations:

- tkinter is already installed
- tk and ttk come with it

Importing tkinter

To import both `tk` and `ttk`, use the following code:

```
from tkinter import *
from tkinter import ttk
```

Creating A Named Window

Before we can add widgets to our tkinter app, we need to ensure that a window is created as a base. It is usually best practice to also name the window.

Our window will be called our `root`. You can technically create a window with any variable name, but the standard practice is to call it `root`.

```
root = Tk()
```

We can also give the window a name to make it clearer to users what the intent of the application is.

```
root.title("This Is The Title Of The Window")
```

Add a Mainloop

We won't discuss in depth at this point what adding the `mainloop()` does. Right now adding the `mainloop()` will make it so that the window stays open when the program runs.

```
root.mainloop()
```

For the lessons on widgets, make sure the `mainloop()` is at the end of your code.

Once you have a window and `tkinter` has been imported, you are ready to start adding widgets to create a GUI!