Module 4-1: Graphical User Interfaces

**Basics, Events, Widgets**

There are many frameworks that can be used to create graphical user interfaces. We will be focusing on using Tkinter! Others are: PyQT, Kivy, Jython, WxPython, PyGUI.

Tkinter is open-source, famous for its simplicity, and usually comes bundled with Python.

**Event-Driven Programming**

User actions like clicks and button typing are called **events**

Making network connections and background tasks completing are examples of **events** that are **not controlled by the user**

**Event Handler:** A function that performs an action in response to an **event.** And we **bind handlers to events** when programming.

**TKinter Elements**

**We generally construct a tree of widgets for our GUI. Each widget will have a parent widget leading all the way up to the root window of our application.**

**Start by importing Tkinter:**

**Creating A Window:**

To create widgets under a parent widget simply use things such as:

There are 2 layout management libraries that are used to manage the GUI widgets within a GUI window.

* The Pack Library: The pack() method organizes widgets in a block, either horizontally or vertically:
  + **label.pack()** or **button.pack()** for example
* The Grid Library: The **grid()** method arranges widgets in a grid-like structure with rows and columns:
  + **label.grid(row = 0, column = 0)** or **button.grid(row = 1, column = 0)**

**Event Handling:**



**Running The Application:**

To run the Tkinter event loop, you run the **mainloop() method** on the root window. This method **listens for events** and **updates the GUI** accordingly.

**Graphical User Interfaces: Geometry, Toolkits, and Conventions**

**Geometry**

Not that kind of Geometry! Geometry in Python refers to the way in which management and arrangement of widgets occur.

There are 3 main Geometry Managers. The other 2 have been covered earlier and will be reiterated below. The newer one is the Place Geometry Manager:

* Place Geometry Manager: The place() method allows you to specify the exact position and size of the widget within the window using pixel coordinates. It is more precise but more complex.
* The Pack Library: The pack() method organizes widgets in a block, either horizontally or vertically: **label.pack()** or **button.pack()** for example
* The Grid Library: The **grid()** method arranges widgets in a grid-like structure with rows and columns: **label.grid(row = 0, column = 0)** or **button.grid(row = 1, column = 0)**

**Toolkits**

Tkinter boasts many libraries or modules that help extend the functionality of Tkinter. They proved other widgets, themes, and utilities.

These toolkits can be installed using Python’s package manager, such as pip.

Popular Tkinter Toolkits:

* **ttk (Themed Tkinter ):** Contains widgets that are more modern and consistent across platforms.
* **Pillow:** Not a Tkinter module or toolkit but, Pillow provides a wide range of image processing capabilities, including opening, manipulating, and saving different image file formats. It is often used with Tkinter.
* **ttkthemes:** more tkinter stuff
* **Pygubu Designer:** Pygubu is a RAD (Rapid Application Development) tool for building GUI applications with Tkinter. It allows you to design your GUI visually using a drag-and-drop interface and automatically generates the corresponding Python code.
* **TkinterDnD:** TkinterDnD is an extension module that adds support for drag-and-drop functionality to Tkinter applications. It allows you to drag items (such as files or text) between widgets within your application or between your application and other applications.
* **tksheet**: tksheet is a Python library that provides a spreadsheet-like table widget for Tkinter. It allows you to display and manipulate tabular data in a grid format, similar to a spreadsheet application.