6 Key Board Navigation

1

# What Is Keyboard Navigation?

Keyboard navigation refers to the ability to control and interact with a uer interface using keys like **Tab**, **Arrow keys**, **Enter**, and **custom shortcuts**. In desktop **graphical user interfaces** (GUIs) like tkinter, this means:

* **Tabbing through widgets** (like buttons, entries, checkboxes)
* **Triggering actions** with keys (e.g., Enter to submit, Esc to cancel)
* **Custom hotkeys** for power users (e.g., Ctrl+S to save)

This is *essential* for accessibility, especially for users who rely on screen readers or cannot use a mouse.

# How It Works in tkinter

Tkinter supports keyboard navigation through **focus management** and **event bindings**.

## Focus Management

1. Widgets like **Entry, Button, Text,** etc., can receive focus.
2. Use **widget.focus\_set()** to manually set focus.
3. Use **Tab** to move forward and **Shift+Tab** to move backward through focusable widgets.

Binding Keyboard Events

You can bind keys to actions using **.bind():**

Simple code to capture a Keypress. Now don’t go looking for your key presses in a label, because when you use print, you are printing only to the terminal. But if you press a key on your keyboard and look at the terminal, you will see the key that you pressed.

Create a python file **keyboard\_navigation.py** in Visual Studio

import tkinter as tk

def on\_key(event):

    print(f"You pressed: {event.keysym}")

root = tk.Tk()

root.bind("<Key>", on\_key)  # Global key capture

root.mainloop()

Run the program and click on the window first. That way the app understands that you are talking to it. Now start pressing Keys and look at your terminal.

2

You can also bind specific keys. Here we are targeting the **Enter** Key and the **Escape** Key:

import tkinter as tk

def on\_key(event):

    print(f"You pressed: {event.keysym}")

root = tk.Tk()

root.bind("<Key>", on\_key)  # Global key capture

root.bind("<Return>", lambda e: print("Enter pressed"))

root.bind("<Escape>", lambda e: print("Escape pressed"))

root.mainloop()

Now click on the window again after running it, and then hit either the **Enter** or the **Escape** key.

3

You can tie this in to a Widget. Here we are using an entry box, and hitting the ctrl-s will give you a message

import tkinter as tk

def on\_key(event):

    print(f"You pressed: {event.keysym}")

root = tk.Tk()

root.bind("<Key>", on\_key)  # Global key capture

root.bind("<Return>", lambda e: print("Enter pressed"))

root.bind("<Escape>", lambda e: print("Escape pressed"))

entry = tk.Entry(root)

entry.config(background="#f0d38c", foreground="black")

root.geometry("300x200")

entry.pack()

entry.bind("<Control-s>", lambda e: print("Save triggered"))

root.mainloop()

Notice that you can hit the **left ctrl** key or the **right ctrl** key, and it will tell you which one you hit. It also will tell you that you hit the **s** key. This will be after seeing the message (Save Triggered). The **ctrl-s** is a common shortcut key for the save command.

4

# Menu shortcut keys

You can even set up keyboard shortcuts for menus

import tkinter as tk

def on\_key(event):

    print(f"You pressed: {event.keysym}")

root = tk.Tk()

root.bind("<Key>", on\_key)  # Global key capture

root.bind("<Return>", lambda e: print("Enter pressed"))

root.bind("<Escape>", lambda e: print("Escape pressed"))

def save\_function(event=None):

    print("Save triggered")

menu = tk.Menu(root)

file\_menu = tk.Menu(menu, tearoff=0)

file\_menu.add\_command(label="Save", accelerator="Ctrl+S", command=save\_function)

menu.add\_cascade(label="File", menu=file\_menu)

root.config(menu=menu)

root.bind("<Control-s>", save\_function)

Our save button isn’t really going to save anything, but it does demonstrate how we add a keyboard trigger event.

5