Lecture 28: GUI using Tkinter

Comp 102

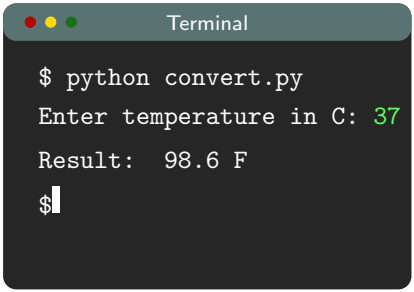
Forman Christian University

All semester, your programs lived in a terminal.

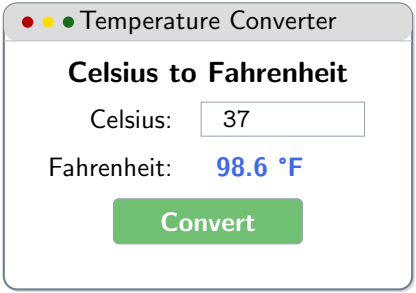
All semester, your programs lived in a terminal.

Today, they get a window.

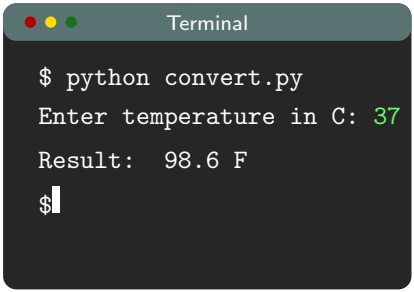
Same program. Different experience.



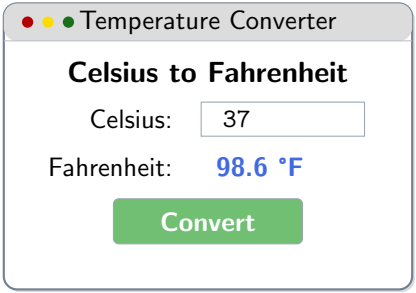
VS.



Same program. Different experience.



VS.



Which one would you rather use **every day**?

GUI = **G**raphical **U**ser **I**nterface

GUI = **G**raphical **U**ser **I**nterface

Three ingredients:

Widgets	**Layout**	**Events**
buttons, labels, fields	where they go	what happens on click

By the end of today, you'll build a **real app**.

Part 1: The Window

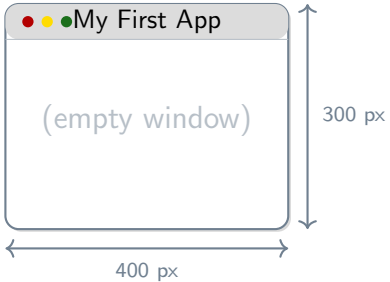
Every App Starts with a Window

```
1 import tkinter as tk
2
3 root = tk.Tk()
4 root.title("My First App")
5 root.geometry("400x300")
6
7 root.mainloop()
```

Every App Starts with a Window

```
1  import tkinter as tk
2
3  root = tk.Tk()
4  root.title("My First App")
5  root.geometry("400x300")
6
7  root.mainloop()
```

`tkinter` is a **standard library** module.
No install needed!



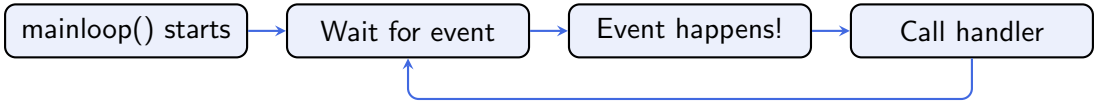
The Event Loop — `mainloop()`

Without `mainloop()`, the window **flashes and dies**.

The Event Loop — `mainloop()`

Without `mainloop()`, the window **flashes and dies**.

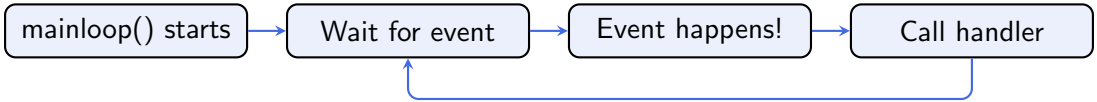
It's an infinite loop that **waits for events**:



The Event Loop — `mainloop()`

Without `mainloop()`, the window **flashes and dies**.

It's an infinite loop that **waits for events**:



Same idea as `turtle.done()` from last class!

A waiter standing ready — does nothing until you raise your hand.

You Try!

Create a window with:

- Your name as the **title**
- Size "500x400"

Run it. A blank window should appear.
Close it with the ✖ button.

Part 2: Labels & Buttons

Widgets = Furniture for Your Window

displays text

clickable action

text input

Widgets = Furniture for Your Window

displays text | clickable action | text input

Every widget needs two things:

1. **Create** it: `tk.Label(root, text="...")`
2. **Place** it: `label.grid(row=0, column=0)`

Widgets = Furniture for Your Window

displays text | clickable action | text input

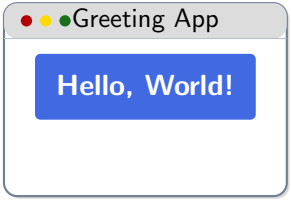
Every widget needs two things:

1. **Create** it: `tk.Label(root, text="...")`
2. **Place** it: `label.grid(row=0, column=0)`

Forget step 2? Widget exists but is **invisible**!

Labels — Python Speaks to the User

```
1  import tkinter as tk
2
3  root = tk.Tk()
4  root.title("Greeting App")
5
6  label = tk.Label(root,
7      text="Hello, World!",
8      font=("Arial", 24, "bold"),
9      fg="white",
10     bg="royalblue",
11     padx=20, pady=10)
12 label.grid(row=0, column=0)
13
14 root.mainloop()
```



Buttons — The User Speaks to Python

```
1 def on_click():
2     label.config(text="Button clicked!")
3
4 button = tk.Button(root,
5     text="Click Me!",
6     command=on_click)
7 button.grid(row=1, column=0)
```

Buttons — The User Speaks to Python

```
1 def on_click():
2     label.config(text="Button clicked!")
3
4 button = tk.Button(root,
5     text="Click Me!",
6     command=on_click)
7 button.grid(row=1, column=0)
```

`command=on_click` — the function to call when clicked.

`label.config(text=...)` — **update** the label after creation.

Warning: Parentheses Matter!

`command=on_click` ✓ Pass the function

"Here's my phone number."

`command=on_click()` ✗ Call it **immediately**

"I'm calling you right now."

Warning: Parentheses Matter!

`command=on_click` ✓ Pass the function

"Here's my phone number."

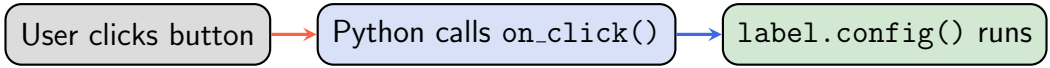
`command=on_click()` ✗ Call it **immediately**

"I'm calling you right now."

This is called a **callback**.

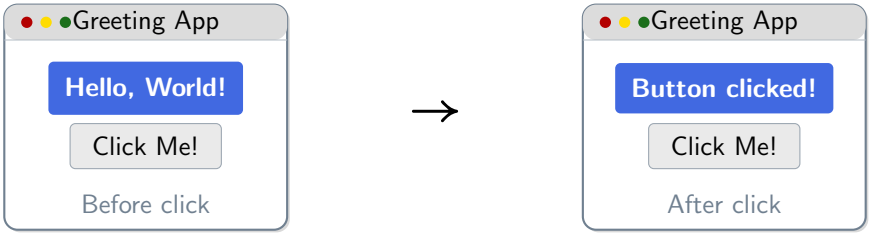
"Call this function **back** when the button is clicked."

How It All Connects



How It All Connects

User clicks button → Python calls `on_click()` → `label.config()` runs



You Try!

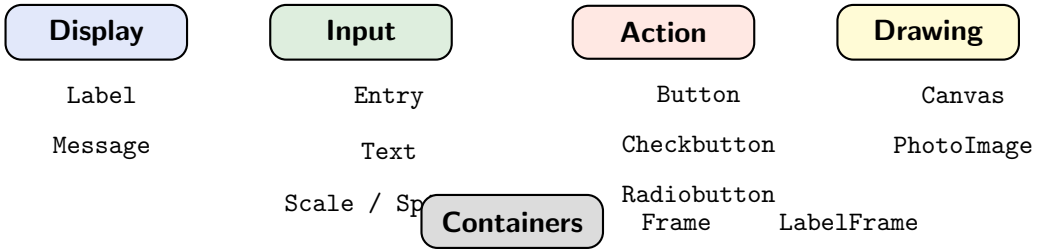
Build a window with:

1. A Label saying `"Hello!"`
2. A Button labeled `"Say Goodbye"`
3. Clicking the button changes the label to `"Goodbye!"`

Hint: Use `label.config(text="Goodbye!")` inside your callback function.

Part 3: The Widget Toolkit

Tkinter Has Many Widgets

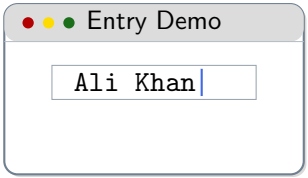


We've seen `Label` and `Button`.

Let's meet a few more...

Entry — Single-Line Text Input

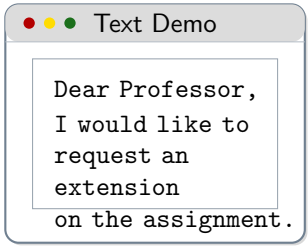
```
1  name = tk.Entry(root,
2      font=("Arial", 14),
3      width=20)
4  name.grid(row=0, column=0)
5
6  # Read what user typed:
7  text = name.get()
8
9  # Clear the entry:
10 name.delete(0, tk.END)
11
12 # Pre-fill with text:
13 name.insert(0, "Type here...")
```



`.get()` → `"Ali Khan"`
`.delete(0, END)` → clears
`.insert(0, "...")` → fills

Text — Multi-Line Text Area

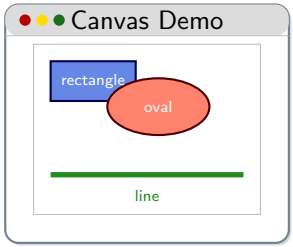
```
1  text_box = tk.Text(root,
2      height=5, width=30,
3      font=("Arial", 12))
4  text_box.grid(row=0, column=0)
5
6  # Read all content:
7  content = text_box.get(
8      "1.0", tk.END)
9
10 # Insert text:
11 text_box.insert(tk.END,
12     "Hello!\n")
13
14 # Clear everything:
```



Like a text editor inside your app!
"1.0" = line 1, character 0

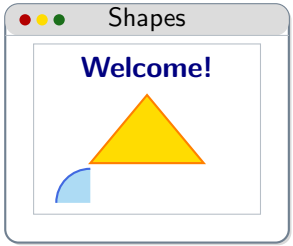
Canvas — Drawing Shapes

```
1  canvas = tk.Canvas(root,
2      width=300, height=200,
3      bg="white")
4  canvas.grid(row=0, column=0)
5
6  # Rectangle (x1,y1, x2,y2)
7  canvas.create_rectangle(
8      20, 20, 120, 80,
9      fill="royalblue",
10     outline="navy")
11
12 # Oval (bounding box)
13 canvas.create_oval(
14     150, 20, 280, 100,
```



Canvas — Text & More Shapes

```
# Text on canvas
canvas.create_text(150, 30,
    text="Welcome!",
    font=("Arial", 18, "bold"),
    fill="navy")

# Polygon (list of points)
canvas.create_polygon(
    150, 50, 200, 130,
    100, 130,
    fill="gold",
    outline="orange")

# Arc (partial oval)
```



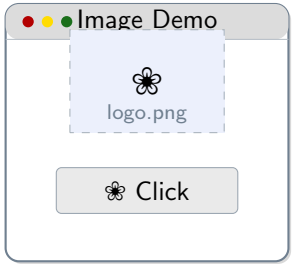
Canvas is like turtle — but inside a GUI window!

Displaying Images

```
# Load an image
photo = tk.PhotoImage(
    file="logo.png")

# Display in a Label
img_label = tk.Label(root,
    image=photo)
img_label.grid(row=0, column=0)

# Or on a Button
img_btn = tk.Button(root,
    image=photo,
    command=on_click)
img_btn.grid(row=1, column=0)
```

Displaying Images

```
# Load an image
photo = tk.PhotoImage(
    file="logo.png")

# Display in a Label
img_label = tk.Label(root,
    image=photo)
img_label.grid(row=0, column=0)

# Or on a Button
img_btn = tk.Button(root,
    image=photo,
    command=on_click)
img_btn.grid(row=1, column=0)
```



Supports: .png, .gif
For .jpg: use Pillow library

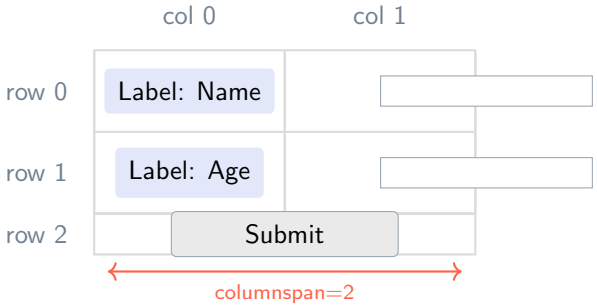
Canvas + Image Together

```
canvas = tk.Canvas(root, width=400, height=300)
canvas.grid(row=0, column=0)

# Load and place image on canvas
bg = tk.PhotoImage(file="background.png")
canvas.create_image(200, 150, image=bg)

# Draw shapes on top of the image
canvas.create_text(200, 30,
    text="My Drawing App",
    font=("Arial", 20, "bold"), fill="white")

canvas.create_oval(50, 50, 150, 150,
    fill="yellow", outline="orange", width=2)
```

Part 4: Layout & Arrangement

Grid = Spreadsheet for Widgets

`.grid(row=, column=)` places widgets in a table.



Grid in Code

```
1  tk.Label(root, text="Name:").grid(
2      row=0, column=0)
3  tk.Entry(root).grid(
4      row=0, column=1)
5
6  tk.Label(root, text="Age:").grid(
7      row=1, column=0)
8  tk.Entry(root).grid(
9      row=1, column=1)
10
11 tk.Button(root, text="Submit").grid(
12     row=2, column=0, columnspan=2)
```

Sticky — The Compass

`sticky` controls **alignment** inside the cell, using compass directions:

Sticky in Action

`sticky="w"`	Align **left**	labels
`sticky="e"`	Align **right**	right-justified text
`sticky="ew"`	**Stretch** horizontally	entries, buttons
`sticky="nsew"`	**Fill entire cell**	canvases, frames

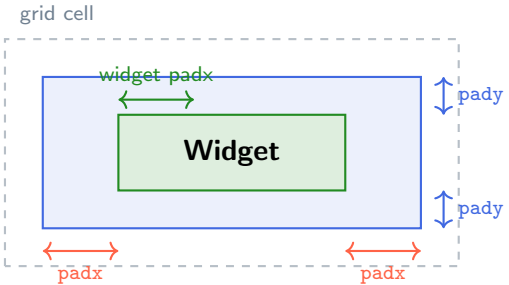
Sticky in Action

`sticky="w"`	Align **left**	labels
`sticky="e"`	Align **right**	right-justified text
`sticky="ew"`	**Stretch** horizontally	entries, buttons
`sticky="nsew"`	**Fill entire cell**	canvases, frames

Without `sticky`, widgets **center** in their cell.

Pro tip: Use `sticky="w"` on labels, `sticky="ew"` on entries for clean, professional-looking forms.

Padding — Breathing Room

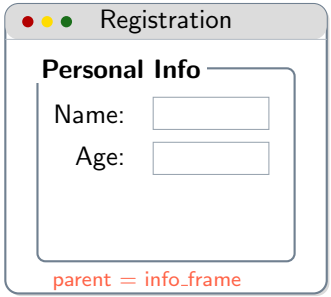


`.grid(padx=10, pady=5)` Space **outside** the widget

`Label(root, padx=10)` Space **inside** the widget

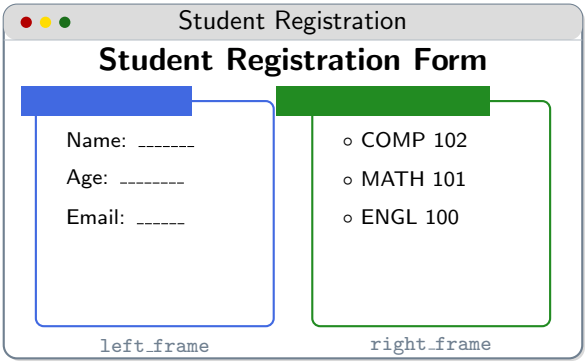
Frame — Grouping Widgets

```
1  # Create a frame container
2  info_frame = tk.LabelFrame(root
       ,
3       text="Personal Info",
4       padx=10, pady=10)
5  info_frame.grid(row=0, column
       =0,
6       padx=10, pady=5)
7
8  # Widgets go INSIDE the frame
9  tk.Label(info_frame,
10      text="Name:").grid(
11      row=0, column=0)
12 tk.Entry(info_frame).grid(
```



Frame Layout Pattern

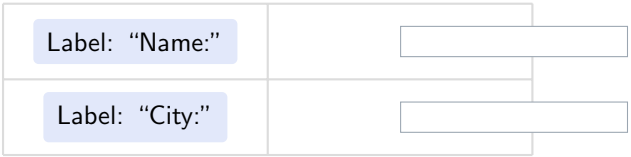
Use frames to **divide your window** into sections:



Each frame has its **own** grid — independent of the main window's grid!

You Try!

Create a form with grid:



Labels in column 0 with `sticky="e"`.
Entry fields in column 1 with `sticky="w"`.
Add `padx=5, pady=5` for spacing.

Part 5: Making It Interactive

StringVar — The Magic Glue

```
result_var = tk.StringVar()
result_var.set("Waiting...")

result_label = tk.Label(root,
    textvariable=result_var)
result_label.grid(row=1, column=0)
```

StringVar — The Magic Glue

```
result_var = tk.StringVar()
result_var.set("Waiting...")

result_label = tk.Label(root,
    textvariable=result_var)
result_label.grid(row=1, column=0)
```

StringVar is a special variable that **auto-updates** any widget connected to it.

`text="fixed"`	Static, never changes
`textvariable=my_var`	Dynamic, linked to a `StringVar`

Reading User Input

```
1 name_entry = tk.Entry(root)
2 name_entry.grid(row=0, column=1)
3
4 def greet():
5     name = name_entry.get()
6     result_var.set(f"Hello, {name}!")
7
8 tk.Button(root, text="Greet",
9     command=greet).grid(row=1, column=0)
```

Reading User Input

```
1  name_entry = tk.Entry(root)
2  name_entry.grid(row=0, column=1)
3
4  def greet():
5      name = name_entry.get()
6      result_var.set(f"Hello, {name}!")
7
8  tk.Button(root, text="Greet",
9      command=greet).grid(row=1, column=0)
```

.get() reads what the user typed.
.set() updates the connected label.

The Three-Step Pattern

(This is the core of **ALL** GUI programming)

The Three-Step Pattern

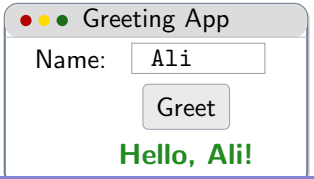
(This is the core of **ALL** GUI programming)

Web apps, mobile apps, desktop apps — **same pattern**.

You Try!

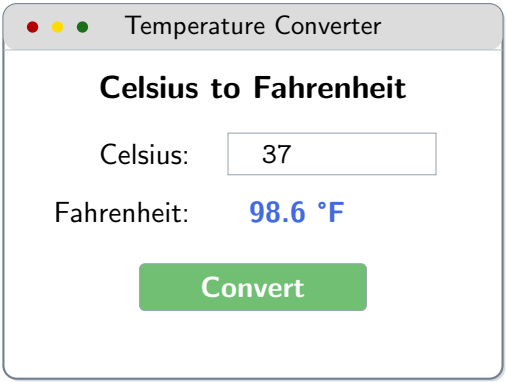
Build a **Greeting App**:

1. An Entry field for the user's name
2. A `"Greet"` button
3. A Label that shows `"Hello, [name]!"`



Part 6: Temperature Converter

Let's Build a Real App



Follow along in Thonny!

Step 1: Window Setup

```
1 import tkinter as tk
2
3 root = tk.Tk()
4 root.title("Temperature Converter")
5 root.geometry("350x200")
```

Run it — you should see an empty window.

Step 2: Header & Input

```
# Header
tk.Label(root, text="Celsius to Fahrenheit",
    font=("Arial", 16, "bold")
    ).grid(row=0, column=0, columnspan=2, pady=10)

# Input row
tk.Label(root, text="Celsius:",
    font=("Arial", 12)
    ).grid(row=1, column=0, sticky="e", padx=5)

celsius_entry = tk.Entry(root,
    font=("Arial", 12), width=10)
celsius_entry.grid(row=1, column=1, sticky="w", padx=5)
```

Step 3: Result Display

```
result_var = tk.StringVar()
result_var.set("---")

tk.Label(root, text="Fahrenheit:",
    font=("Arial", 12)
    ).grid(row=2, column=0,
           sticky="e", padx=5)

tk.Label(root, textvariable=result_var,
    font=("Arial", 12, "bold"), fg="blue"
    ).grid(row=2, column=1,
           sticky="w", padx=5)
```

`textvariable` links this label to our `StringVar`.

Step 4: The Brain

```
def convert():
    try:
        c = float(celsius_entry.get())
        f = c * 9/5 + 32
        result_var.set(f"{f:.1f} °F")
    except ValueError:
        result_var.set("Invalid input!")
```

Step 4: The Brain

```
def convert():
    try:
        c = float(celsius_entry.get())
        f = c * 9/5 + 32
        result_var.set(f"{f:.1f} °F")
    except ValueError:
        result_var.set("Invalid input!")
```

`try/except` — from Week 9!
`f"{f:.1f}"` — 1 decimal place, from Week 2.

Step 5: Connect & Launch

```
1 tk.Button(root, text="Convert",
2     font=("Arial", 12),
3     command=convert
4     ).grid(row=3, column=0,
5            columnspan=2, pady=10)
6
7 root.mainloop()
```

Step 5: Connect & Launch

```
1 tk.Button(root, text="Convert",
2     font=("Arial", 12),
3     command=convert
4     ).grid(row=3, column=0,
5            columnspan=2, pady=10)
6
7 root.mainloop()
```

Run it!

Type 100 → 212.0 °F

Type 0 → 32.0 °F

Type 37 → 98.6 °F

The Complete Program

```
1  import tkinter as tk
2
3  root = tk.Tk()
4  root.title("Temperature Converter")
5  root.geometry("350x200")
6
7  tk.Label(root, text="Celsius to Fahrenheit",
8      font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=2, pady=10)
9  tk.Label(root, text="Celsius:", font=("Arial", 12)).grid(row=1, column=0, sticky="e", padx=5)
10 celsius_entry = tk.Entry(root, font=("Arial", 12), width=10)
11 celsius_entry.grid(row=1, column=1, sticky="w", padx=5)
12 result_var = tk.StringVar()
13 result_var.set("---")
14 tk.Label(root, text="Fahrenheit:", font=("Arial", 12)).grid(row=2, column=0, sticky="e", padx=5)
15 tk.Label(root, textvariable=result_var, font=("Arial", 12, "bold"),
16     fg="blue").grid(row=2, column=1, sticky="w", padx=5)
17
18 def convert():
19     try:
20         c = float(celsius_entry.get())
21         f = c * 9/5 + 32
22         result_var.set(f"{f:.1f} °F")
23     except ValueError:
24         result_var.set("Invalid input!")
25
26 tk.Button(root, text="Convert", font=("Arial", 12),
27     command=convert).grid(row=3, column=0, columnspan=2, pady=10)
28 root.mainloop()
```

Part 7: Polish & Explore

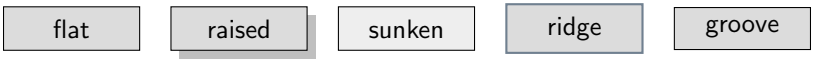
Styling Your App

```
tk.Button(root, text="Convert",
    font=("Arial", 12, "bold"),
    bg="#4CAF50",                # Green background
    fg="white",                  # White text
    activebackground="#45a049", # Darker on click
    relief="raised",             # 3D effect
    padx=20, pady=5)
```

Styling Your App

```
tk.Button(root, text="Convert",
    font=("Arial", 12, "bold"),
    bg="#4CAF50",                # Green background
    fg="white",                  # White text
    activebackground="#45a049", # Darker on click
    relief="raised",             # 3D effect
    padx=20, pady=5)
```

Relief styles:



More Widgets (Preview)

```
1  # Checkbutton
2  var = tk.BooleanVar()
3  tk.Checkbutton(root,
4      text="Remember me",
5      variable=var)
6
7  # Radiobutton
8  choice = tk.StringVar(value="C"
       )
9  tk.Radiobutton(root,
10     text="Celsius",
11     variable=choice, value="C")
12 tk.Radiobutton(root,
13     text="Fahrenheit",
```

Checkbutton:

☑ Remember me

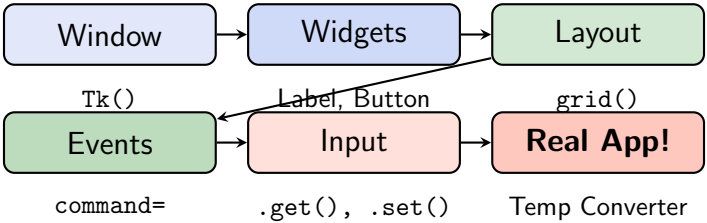
Radiobutton:

◉ Celsius

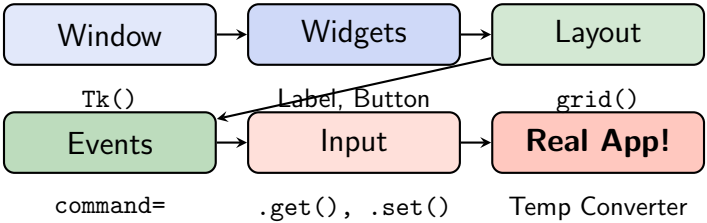
○ Fahrenheit

Summary

The Journey Today



The Journey Today



Every concept — variables, functions, `try/except`, strings — came from **earlier in the course**. Tkinter just adds a **face**.

The GUI Programming Recipe

1. Create the window — `tk.Tk()`
2. Create widgets — `Label, Button, Entry`
3. Place them — `.grid(row=, column=)`
4. Write callbacks — `def on_click(): ...`
5. Connect callbacks — `command=on_click`
6. Start the loop — `root.mainloop()`

The GUI Programming Recipe

1. Create the window — `tk.Tk()`
2. Create widgets — `Label, Button, Entry`
3. Place them — `.grid(row=, column=)`
4. Write callbacks — `def on_click(): ...`
5. Connect callbacks — `command=on_click`
6. Start the loop — `root.mainloop()`

VS Code, Chrome, Thonny — they all follow this **same pattern**.

Key Commands Cheat Sheet

Category	Code
Window	`tk.Tk()`, `.title()`, `.geometry()`, `.mainloop()`
Label	`tk.Label(root, text=, font=, fg=, bg=)`
Button	`tk.Button(root, text=, command=)`
Entry	`tk.Entry(root)`, `.get()`, `.insert()`, `.delete()`
Layout	`.grid(row=, column=, columnspan=, sticky=)`
Variables	`tk.StringVar()`, `.get()`, `.set()`
Update	`.config(key=val)`, `textvariable=`

The Big Idea

Programs aren't just **logic**.

They're **experiences**.

Today you learned to build experiences
that **anyone** can use — not just programmers.

Last class, the **turtle** was your artist.
Today, **you** became the architect.

Practice Ideas

1. **Reverse converter:** add F→C mode with Radiobuttons
2. **BMI calculator:** height & weight entries, compute BMI
3. **Counter app:** label + "+" and "−−" buttons
4. **Quiz app:** question label, 4 Radiobutton choices, "Check" button
5. **Color picker:** R, G, B entries, label shows that color

Questions?