

# Introduction to GUI and Event-Driven Programming

---

01219116 Computer Programming II

*Chaiporn Jaikaeo*

*Department of Computer Engineering  
Kasetsart University*

# Outline

---

- Python GUI frameworks
- Basics of Tkinter
- GUI widgets
- Widget styling
- Event loop and event-driven programming

# GUI vs. CLI vs. TUI

```
code $ ls /usr/bin/x*
/usr/bin/xar          /usr/bin/xed          /usr/bin/xpath
/usr/bin/xargs        /usr/bin/xgettext.pl  /usr/bin/xpath5.18
/usr/bin/xattr        /usr/bin/xgettext5.18.pl /usr/bin/xslt-config
/usr/bin/xattr-2.7    /usr/bin/xip          /usr/bin/xsltproc
/usr/bin/xcodeselect  /usr/bin/xjc          /usr/bin/xsubpp
/usr/bin/xcodesbuild  /usr/bin/xml2-config   /usr/bin/xsubpp5.18
/usr/bin/xcrun        /usr/bin/xml2man       /usr/bin/xxd
/usr/bin/xcscontrol   /usr/bin/xmlcatalog
/usr/bin/xcsdiagnose  /usr/bin/xmllint

code $ cal
      January 2021
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
code $
```

**Command-Line Interface (CLI)**



```
code -- htop -- 80x24

1  [|||||] 2.6% Load average: 2.02 2.03 1.80
2  [|||||] 10.7% Uptime: 5 days, 22:52:00
3  [|||||]
4  [|||||]
Mem [|||||] 3.70G/8.00G
Swp [|||||] 450M/1.00G

PID USER PRI NI VIRT RES S CPU% MEM% TIME+ Command
10250 cpj 24 0 18.7G 123M ? 4.9 1.5 16:46.07 /Applications/Visual St
10247 cpj 17 0 4514M 34968 ? 4.4 0.4 13:03.85 /Applications/Visual St
448 cpj 24 0 5057M 58156 ? 1.5 0.7 12:45.49 /Applications/Utilities
22185 cpj 17 0 4233M 10388 ? 1.0 0.1 0:00.07 /usr/sbin/screencapture
495 cpj 8 0 5294M 312M ? 1.0 3.8 2h30:07 /Applications/Google Ch
10242 cpj 17 0 9244M 59252 ? 0.7 0.7 2:30.06 /Applications/Visual St
375 cpj 17 0 4232M 11708 ? 0.6 0.1 14:50.10 /Library/PrivilegedHelp
261 cpj 17 0 4259M 22248 ? 0.6 0.3 11:32.38 /usr/sbin/universalacce
22184 cpj 24 0 4196M 2432 R 0.2 0.0 0:00.05 htop
526 cpj 17 0 8618M 40088 ? 0.2 0.5 7:36.63 /Applications/Google Ch
511 cpj 8 0 4514M 80504 ? 0.0 1.0 54:44.87 /Applications/Google Ch
374 cpj 17 0 4251M 14424 ? 0.0 0.2 1:05.10 /Library/Application Su
366 cpj 17 0 4299M 21504 ? 0.0 0.3 11:01.03 /Applications/Wacom Tab
20423 cpj 17 0 5590M 186M ? 0.0 2.3 1:02.55 /Applications/Mail.app/

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
```

**Text-based User Interface (TUI)**

# Python GUI Frameworks

---

- Tkinter
  - Most generic; comes with standard Python
- wxPython
  - Native OS look and feel
  - Platform-specific code may be required
- PyQt/PySide
  - More than GUI framework; supports networking, databases, etc.
  - Platform agnostic
  - Requires separate installation of Qt
- *Many more...*

# Tcl/Tk and Tkinter

---

- Tcl (Tool command language)
  - Designed in 1980s as a general-purpose, interpreted programming language
  - Commonly used for rapid prototyping
- Tk (toolkit) is a cross-platform windowing toolkit for Tcl
- Tkinter is a Python interface to Tcl/Tk



# Typical Tkinter Program (No Class)

```
import tkinter as tk
```

```
def say_hello():  
    print("Hello")
```

```
root = tk.Tk()  
root.title("My First GUI App")  
root.geometry("300x100")
```



root window creation

```
label = tk.Label(root, text="Welcome to my GUI application")  
label.pack()
```

widget  
creation

```
btn_hello = tk.Button(root, text="Hello", command=say_hello)  
btn_hello.pack()
```

```
btn_quit = tk.Button(root, text="Quit", command=root.destroy)  
btn_quit.pack()
```

```
root.mainloop()
```

running event loop

event handlers  
(callbacks)

# Typical Tkinter Program (with Class)

```
import tkinter as tk

class App(tk.Tk):

    def __init__(self):
        super().__init__()
        self.title("My First GUI App")
        self.geometry("300x100")
        self.label = tk.Label(self, text="Welcome to my GUI application")
        self.label.pack()
        self.btnHello = tk.Button(self, text="Hello", command=self.say_hello)
        self.btnHello.pack()
        self.btnQuit = tk.Button(self, text="Quit", command=self.destroy)
        self.btnQuit.pack()

    def say_hello(self):
        print("Hello")

    def run(self):
        self.mainloop()

if __name__ == "__main__":
    app = App()
    app.run()
```



# The Root Window

---

- A Tkinter application is initialized by creating one (and only one) root window
- A root window is created by instantiating an object of the Tk class

```
import tkinter as tk

root = tk.Tk()
root.title("My First App")
root.geometry("300x200")
```

- For OO style, we can subclass Tk and wrap everything inside our own class

```
import tkinter as tk

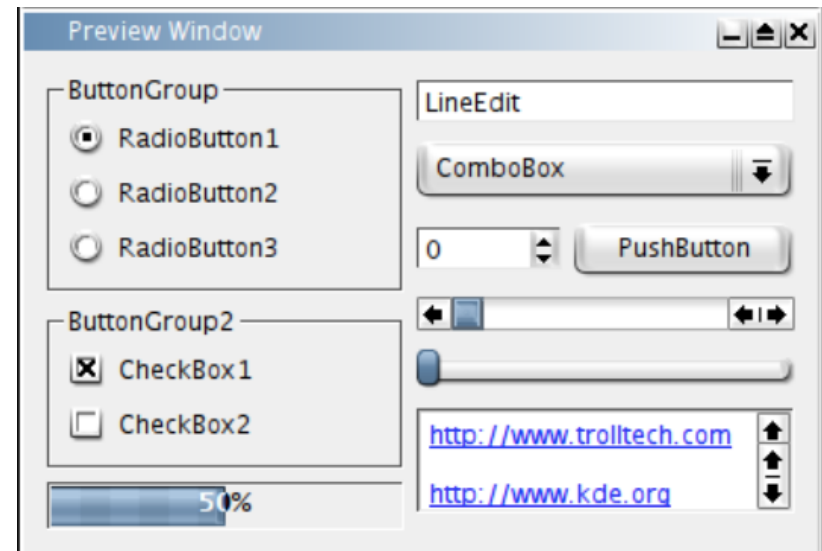
class Application(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("My First App")
        self.geometry("300x200")

app = Application()
```



# GUI Widgets

- A widget (or a control) is an element of interaction, appearing as a visible part of the application's GUI, such as a button or a scroll bar
- Each widget facilitates a specific type of user-computer interaction



[https://en.wikipedia.org/wiki/Graphical\\_widget](https://en.wikipedia.org/wiki/Graphical_widget)

# Creating a Widget

---

- A widget is created by instantiating an object of a widget class
- A list of attributes may also be specified to control the behavior and appearance of the widget
  - See <https://anzelg.github.io/rin2/book2/2405/docs/tkinter/std-attrs.html> for more comprehensive list of attributes

```
label1 = tk.Label(text="Name")  
label2 = tk.Label(text="School", foreground="blue")
```

- The widget will not show up on screen until one of its layout methods is called
  - we will use the **pack** layout for now

```
label1.pack()  
label2.pack()
```

# Tk Widgets

Widget	Purpose
Label	Display static text or an image
Button	Execute a specific task; a “do this now” command
Menu	Implement top-level, pulldown, and popup menus
Menubutton	Display popup or pulldown menu items when activated
OptionMenu	Create a popup menu, and a button to display it
Entry	Enter one line of text
Text	Display and edit formatted text, possibly with multiple lines
Checkbutton	Set on-off, True-False selections
Radiobutton	Allow one-of-many selections
Listbox	Choose one or more alternatives from a list
Scale	Select a numerical value by moving a “slider” along a scale
Spinbox	Allow to select from a numerical range or from a list

# Widget Styling

---

- A widget can be styled during its creation, or reconfigured later, by setting its appearance-related attributes

```
# style the widget during its creation
lbl = tk.Label(text="Huge Label", font=("Arial",48))

# reconfigure the widget using config() or configure() method
lbl.config(foreground="red")

# widget can also be used as a dict for reconfiguration
lbl["background"] = "yellow"
lbl["text"] = "Fancy Label"
```

# Tk Themed Widgets

- The `tkinter.ttk` module provides access to the Tk themed widget set
- Ttk widgets are themed, i.e., their appearances are controlled by a set of configurable styles

```
import tkinter as tk
import tkinter.ttk as ttk

root = tk.Tk()
root.title("Tk vs. Ttk Widgets")
root.geometry("400x100")

btn_tk = tk.Button(text="Tk Button")
btn_tk.pack()

btn_ttk = ttk.Button(text="Ttk Button")
btn_ttk.pack()

root.mainloop()
```



# Ttk Widgets

- Ttk implements all original Tk widgets, and provides additional widgets as follows

Widget	Purpose
Combobox	Combine a text field with a pop-down list of values
Progressbar	Show the status of a long-running operation
Separator	Display a horizontal or vertical separator bar
Sizegrip	Allow user to resize the entire application window
Treeview	Display a hierarchical collection of items

# Ttk Widget Styling

- Tk widget styling (their styles can be completely independent)

```
label1 = tk.Label(text="Label 1", fg="black", bg="white")
```

- Ttk widget styling (their styles are defined as part of a theme)
  - Define a new style called BW.TLabel, which is derived from the default Label style (called TLabel)

```
style = ttk.Style()
style.configure("BW.TLabel", foreground="black", background="white")

label1 = ttk.Label(text="Label 1", style="BW.TLabel")
```

- Make all Label widgets display blue background

```
style.configure("TLabel", background="blue")
```

- Make every widget use Arial font with size of 24pt

```
style.configure(".", font=("Arial",24))
```

# Label Widget

---

- Display static text or an image
- Typically provide no user interaction

macOS



Windows



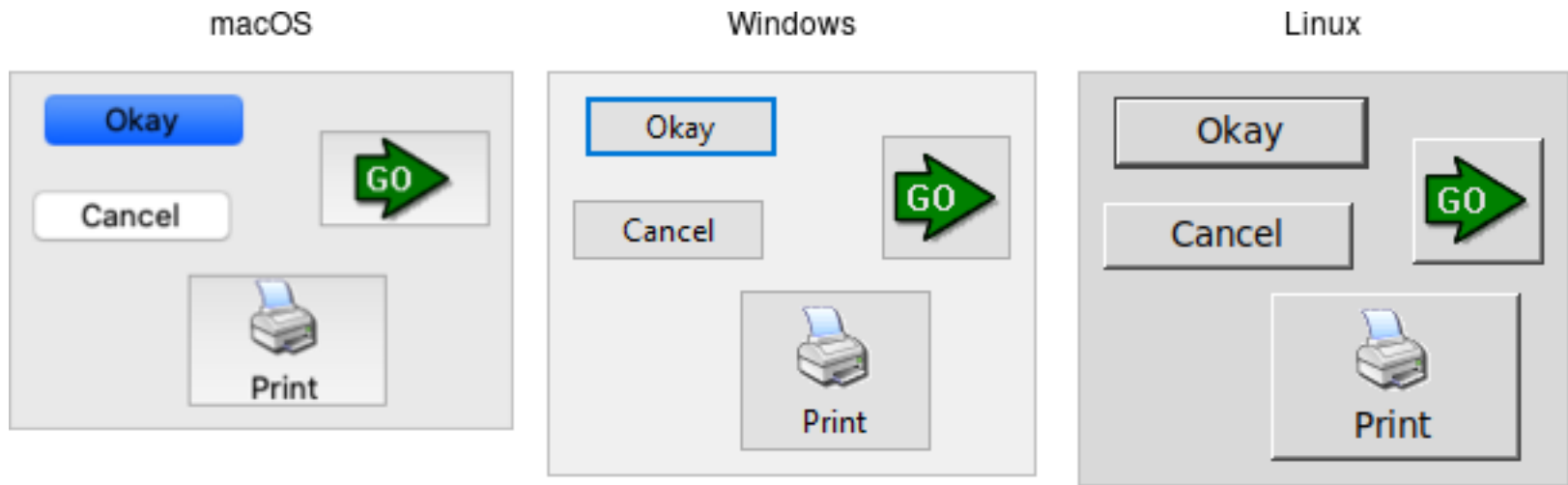
Linux





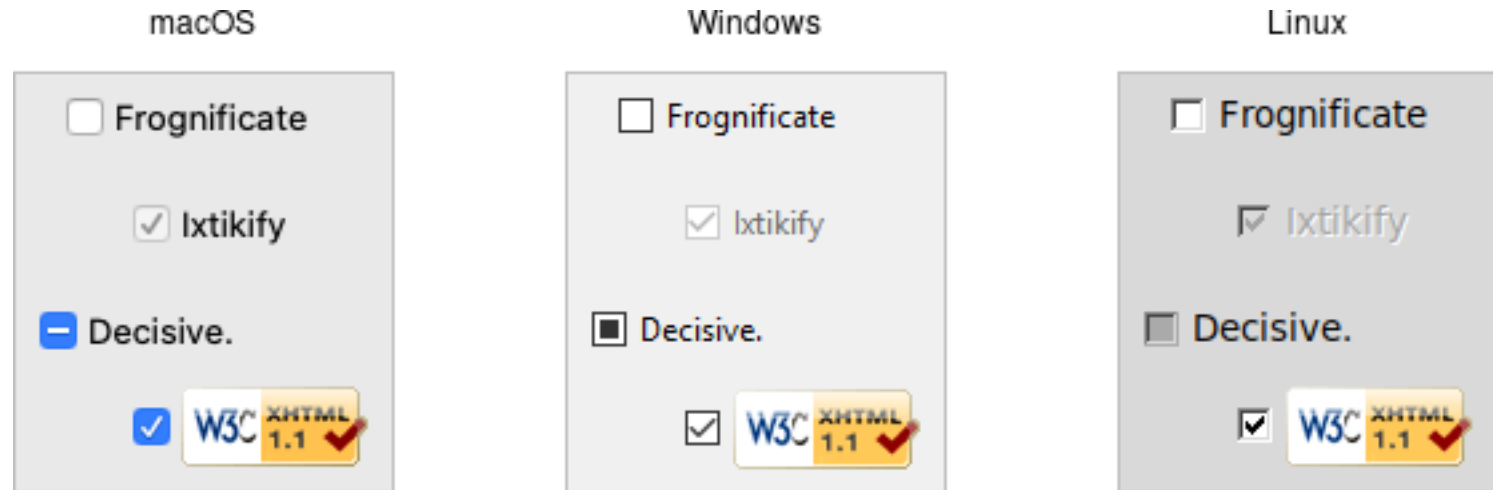
# Button Widget

- Execute a specific task; a “do this now” command



# Checkbutton Widget

- Set on-off, True-False selections



# Radiobutton Widget

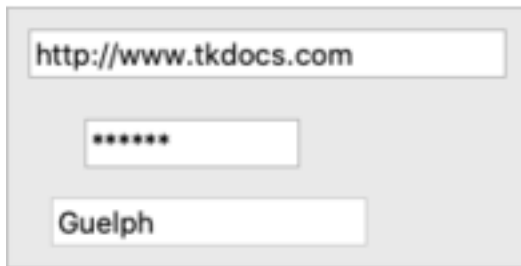
- Allow one-of-many selections



# Entry Widget

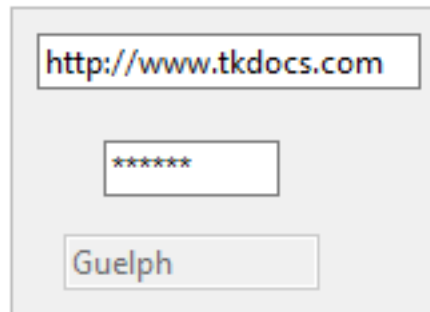
- Enter one line of text

macOS



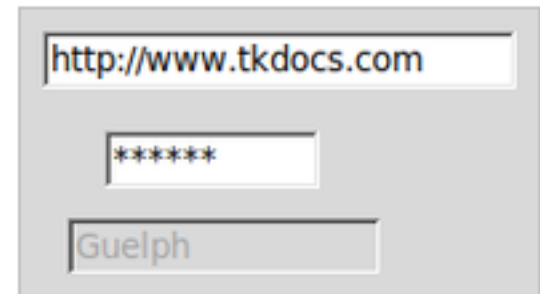
A screenshot of a macOS Tkinter window showing three entry widgets. The top widget contains the text 'http://www.tkdcs.com'. The middle widget contains six asterisks '\*\*\*\*\*'. The bottom widget contains the text 'Guelph'.

Windows



A screenshot of a Windows Tkinter window showing three entry widgets. The top widget contains the text 'http://www.tkdcs.com'. The middle widget contains six asterisks '\*\*\*\*\*'. The bottom widget contains the text 'Guelph'.

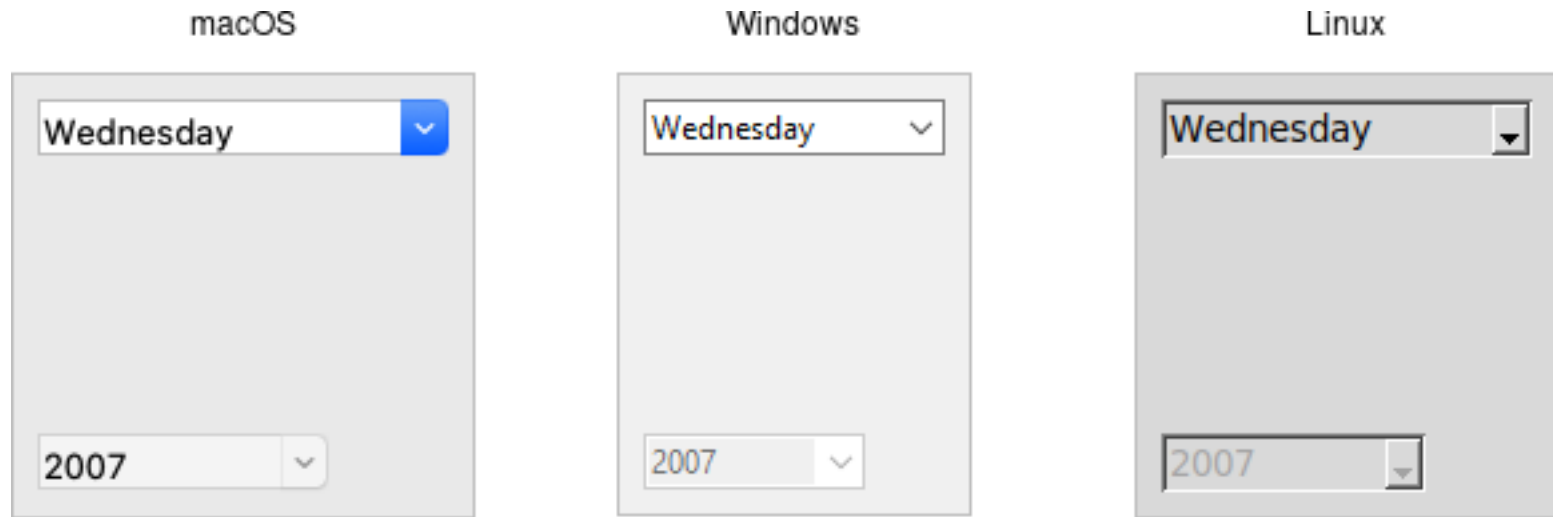
Linux



A screenshot of a Linux Tkinter window showing three entry widgets. The top widget contains the text 'http://www.tkdcs.com'. The middle widget contains six asterisks '\*\*\*\*\*'. The bottom widget contains the text 'Guelph'.

# Combobox Widget

- Combine a text field with a pop-down list of values

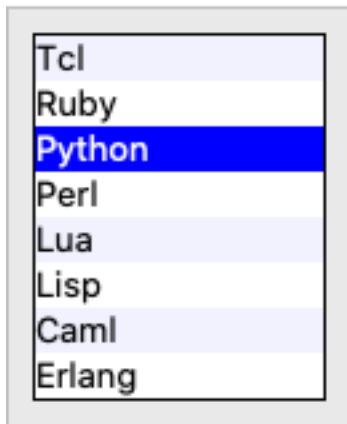


# Listbox Widget

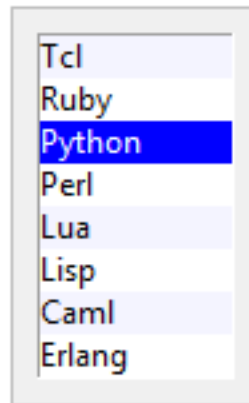
---

- Choose one or more alternatives from a list

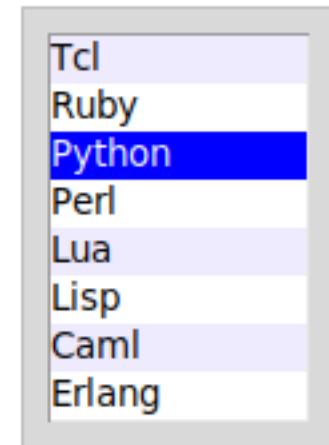
macOS



Windows



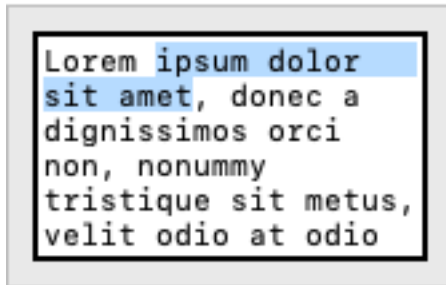
Linux



# Text Widget

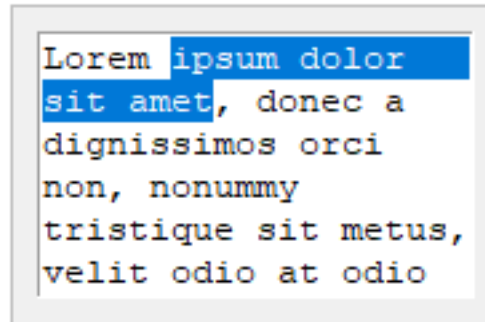
- Display and edit formatted text, possibly with multiple lines

macOS

A screenshot of a text widget on macOS. The text is displayed in a monospaced font. The first line, "Lorem ipsum dolor", is highlighted in light blue. The rest of the text is in a standard black font.

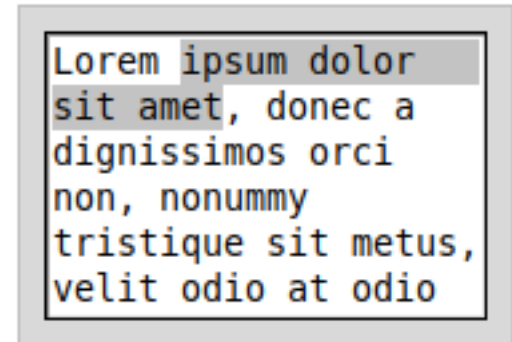
```
Lorem ipsum dolor  
sit amet, donec a  
dignissimos orci  
non, nonummy  
tristique sit metus,  
velit odio at odio
```

Windows

A screenshot of a text widget on Windows. The text is displayed in a monospaced font. The first line, "Lorem ipsum dolor", is highlighted in a dark blue color. The rest of the text is in a standard black font.

```
Lorem ipsum dolor  
sit amet, donec a  
dignissimos orci  
non, nonummy  
tristique sit metus,  
velit odio at odio
```

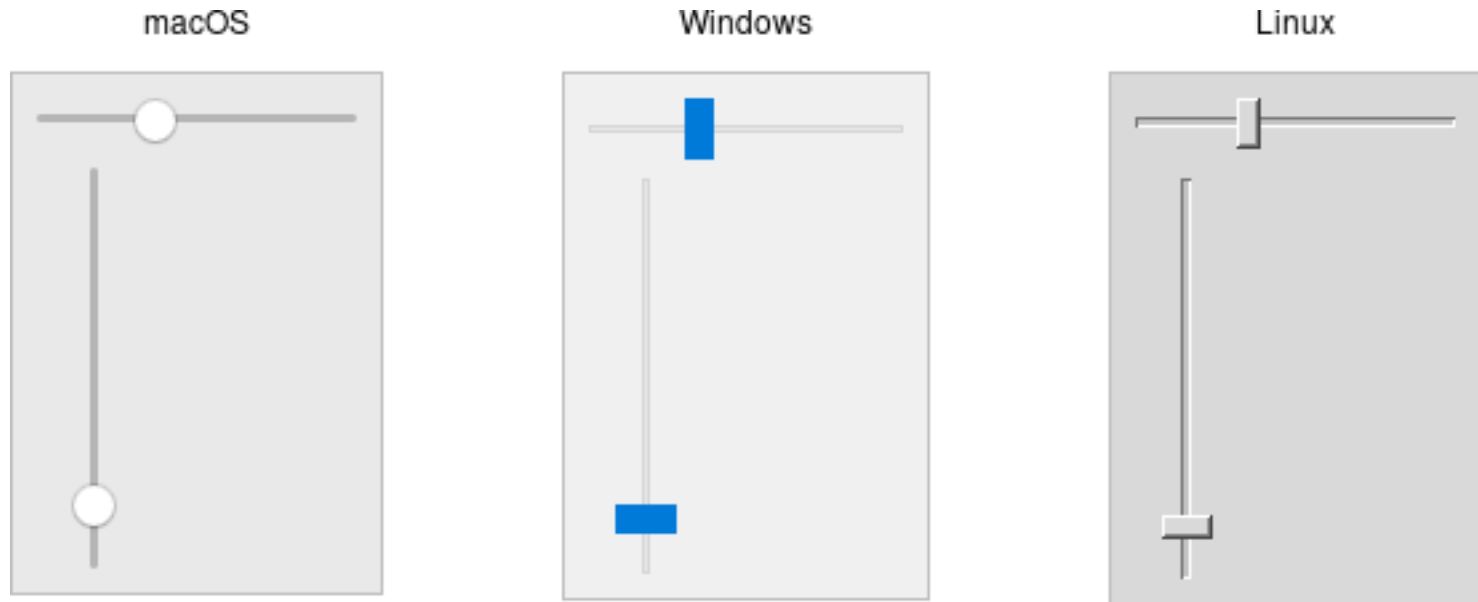
Linux

A screenshot of a text widget on Linux. The text is displayed in a monospaced font. The first line, "Lorem ipsum dolor", is highlighted in a light gray color. The rest of the text is in a standard black font.

```
Lorem ipsum dolor  
sit amet, donec a  
dignissimos orci  
non, nonummy  
tristique sit metus,  
velit odio at odio
```

# Scale Widget

- Select a numerical value by moving a “slider” along a scale

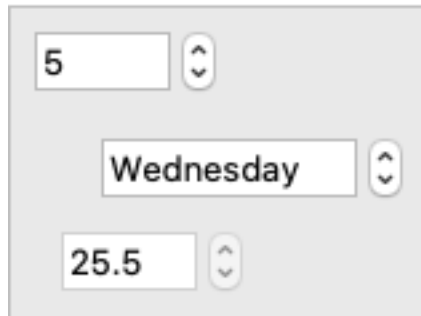




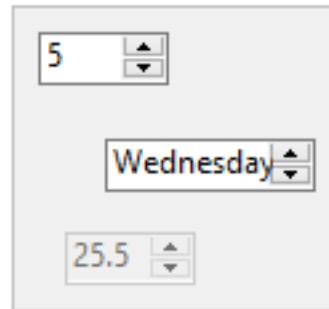
# Spinbox Widget

- Allow to select from a numerical range or from a list

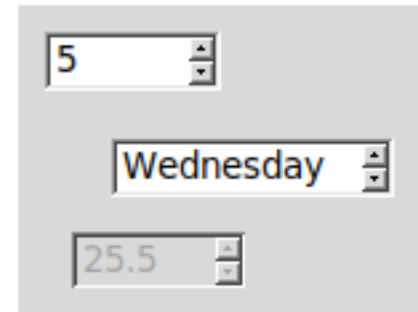
macOS



Windows

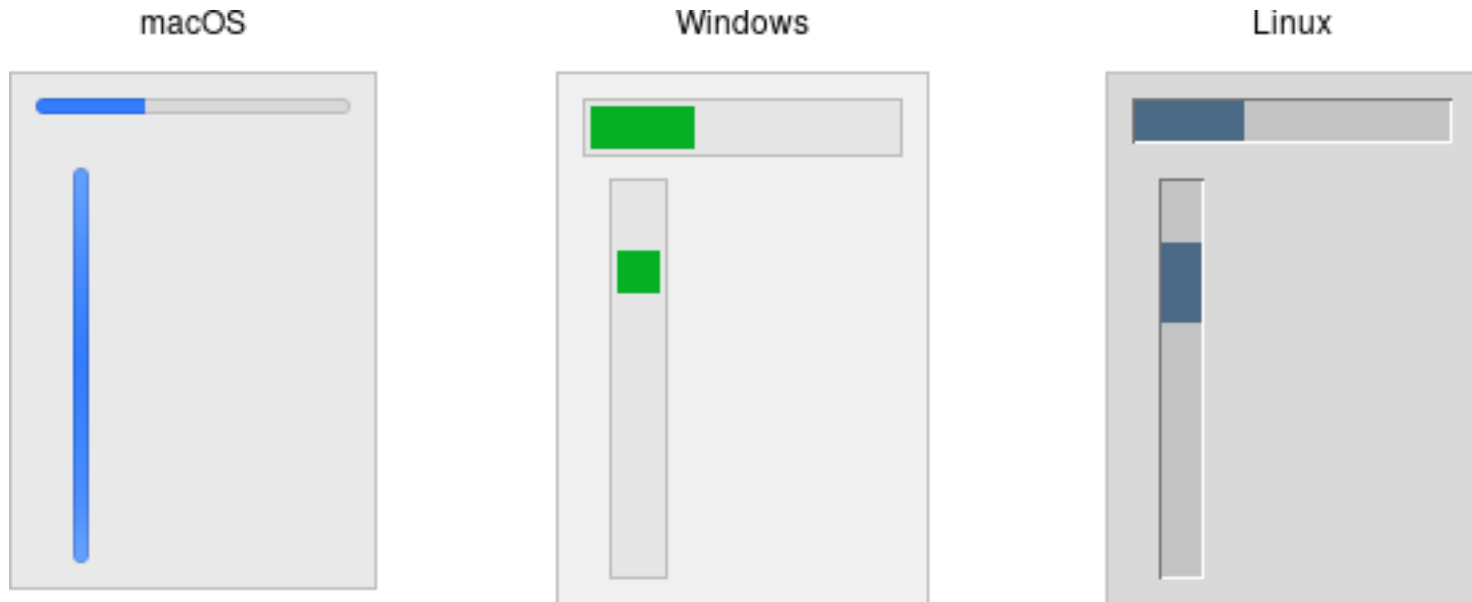


Linux



# Progressbar Widget

- Show the status of a long-running operation



# Treeview Widget

- Display a hierarchical collection of items

macOS

	Size	Modified
widgets	25KB	Yesterday
gallery	2KB	Two weeks ago
▶ resources	220KB	Three weeks ago
▼ tutorial	2.1MB	Ten minutes ago
canvas	18KB	Last week
tree	5KB	Ten minutes ago
text	12KB	Yesterday

Windows

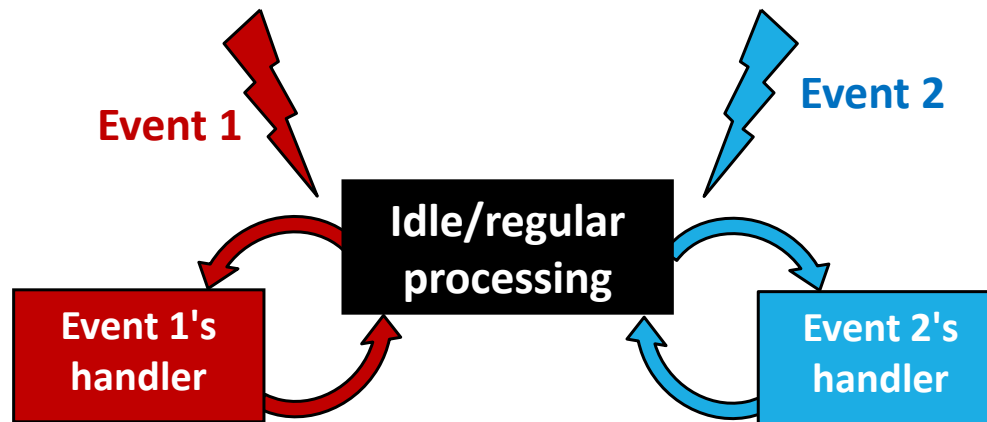
	Size	Modified
widgets	25KB	Yesterday
gallery	2KB	Two weeks ago
▢ resources	220KB	Three weeks ago
▢ tutorial	2.1MB	Ten minutes ago
canvas	18KB	Last week
tree	5KB	Ten minutes ago
text	12KB	Yesterday

Linux

	Size	Modified
widgets	25KB	Yesterday
gallery	2KB	Two weeks ago
@ resources	220KB	Three weeks ago
@ tutorial	2.1MB	Ten minutes ago
canvas	18KB	Last week
tree	5KB	Ten minutes ago
text	12KB	Yesterday

# Event-Driven Programming

- Also known as **reactive programming**
- Perform regular processing or be idle
- React to events when they happen immediately



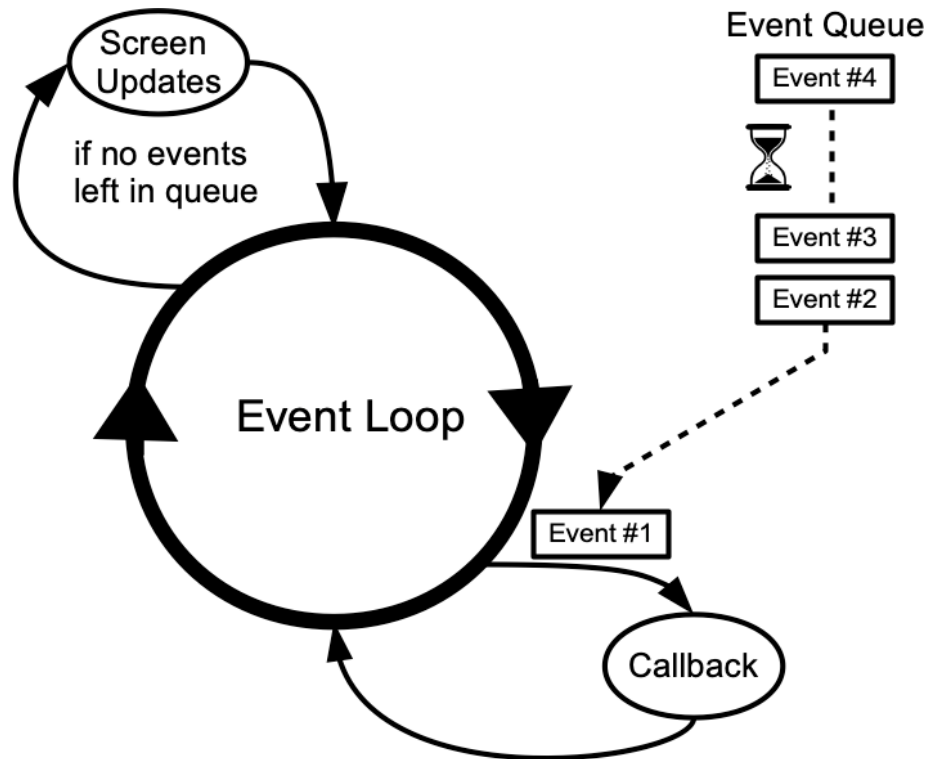
# Possible Event Sources

---

- GUI widgets
- Keyboard and mouse
- Timers
- Control variables
  
- Most event types can be found from
  - <https://anzelg.github.io/rin2/book2/2405/docs/tkinter/event-types.html>

# The Event Loop

- The event loop is started by calling the root window's `mainloop()` method
  - There can be only one event loop in a Tkinter app



# GUI Widget Events

---

- Certain GUI widgets generate events via user interaction
  - E.g., the Button widget accepts a callback as *command* argument

```
def onclick():  
    print("Button is clicked")  
  
btn = ttk.Button(text="Click Me", command=onclick)
```

- Changes of a widget's state such as active, inactive, resized, destroyed, also trigger events

# Keyboard and Mouse Events

---

- Keyboard events are triggered when user presses or releases a key
- Mouse events are triggered when
  - User moves the mouse pointer into, out of, or within a widget
  - User presses or releases a mouse button
- Some keyboard and mouse events' names can be found from:
  - [https://www.python-course.eu/tkinter\\_events\\_binds.php](https://www.python-course.eu/tkinter_events_binds.php)



# Keyboard and Mouse Event Examples

- Run this example and try moving the mouse inside the application or press some keys

```
import tkinter as tk
from tkinter import ttk

root = tk.Tk()
root.title("Event Demonstration")
root.geometry("500x200")

lbl = ttk.Label(text="Starting...", font=("Arial",36))

# make the label fill the container and also expand when the container is resized
lbl.pack(fill=tk.BOTH, expand=1)

lbl.bind("<Leave>", lambda e: lbl.config(text="Moved mouse outside"))
lbl.bind("<Enter>", lambda e: lbl.config(text="Moved mouse inside"))
lbl.bind("<ButtonPress-1>", lambda e: lbl.config(text="Clicked left mouse button"))
lbl.bind("<Double-1>", lambda e: lbl.config(text="Double clicked"))

root.bind("<Return>", lambda e: lbl.config(text="Enter key is pressed"))
root.bind("<Key-F1>", lambda e: lbl.config(text="Key F1 is pressed"))
root.bind("<KeyRelease-F1>", lambda e: lbl.config(text="Key F1 is released"))

root.mainloop()
```

# Timers

- All Tk widgets provide the `after()` method , whose signature is  
`widget.after(delay, callback=None)`
- This method calls the function `callback` after the given `delay` in milliseconds
  - If no function is given, it acts like `time.sleep()` (but in milliseconds instead of seconds)
- Example below waits for 2 seconds after the button is clicked, then changes the text to "Done"

```
def onclick():  
    btn.config(text="Wait for 2 seconds", state=tk.DISABLED)  
    btn.after(2000, lambda: btn.config(text="Done", state=tk.NORMAL))  
  
btn = ttk.Button(text="Start", command=onclick)
```

# Control Variable Classes

- Tkinter provides a number of control variable classes that can generate events

Construction	Data Type	Default Value
<code>tk.StringVar()</code>	str	""
<code>tk.IntVar()</code>	int	0
<code>tk.DoubleVar()</code>	double	0.0
<code>tk.BooleanVar()</code>	bool	False

- The `set()` method must be used to set variable's values
- One may subclass the **Variable** class for custom datatypes

# Control Variable Event Binding

- Use the `trace_add()` method to bind a read/write event to a callback

```
num = tk.IntVar()  
num.trace_add("write", lambda *args: print("New value = {}".format(num.get())))
```

- Specific widgets, such as Label and Entry, allow binding to control variables directly
  - A Label widget accepts a StringVar as *textvariable* argument and monitors its write events to update the label text
  - An Entry widget accepts a StringVar as *textvariable* argument and updates the variable when the text entry changes

```
text = tk.StringVar()  
lbl = ttk.Label(textvariable=text)  
entry = ttk.Entry(textvariable=text)
```

# Conclusion

---

- GUI provides interactive and friendly interface for users
- Tkinter is a simple GUI toolkit readily available in every Python distribution
- A GUI application consists of one root window containing a number of GUI widgets
- Events may be generated by user interacting with widgets as well as other sources such as variable updates and timers
- An event loop is responsible for processing events and updating GUI window

# References

---

- *Tkinter* – Python interface to Tcl/Tk
  - <https://docs.python.org/3/library/tkinter.html>
- Tk themed widgets
  - <https://docs.python.org/3/library/tkinter.ttk.html>
- Tkinter 8.5 reference: a GUI for Python
  - <https://anzelg.github.io/rin2/book2/2405/docs/tkinter/index.html>
- TkDocs Tutorial
  - <https://tkdocs.com/tutorial/index.html>